Designing an active recommender framework to support the development of reasoning mechanisms for smart cyber-physical systems

Tepjit, S.

DOI
10.4233/uuid:943cabcf-697f-4e82-8d51-480d0f171496

Publication date
2022

Document Version
Final published version

Citation (APA)
Tepjit, S. (2022). *Designing an active recommender framework to support the development of reasoning mechanisms for smart cyber-physical systems*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:943cabcf-697f-4e82-8d51-480d0f171496

Designing an active recommender framework to support the development of reasoning mechanisms for smart cyber-physical systems

Sirasak Tepjit

Designing an active recommender framework to support the development of reasoning mechanisms for smart cyber-physical systems

Proefschrift

ter verkrijging van de grad van doctor
aan de Technische Universiteit Delft,
op gezag van
de Rector Magnificus prof.ir. T.H.J.J van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
Woensdag, 6, April, 2022 om 12.30 uur

door
Sirasak Tepjit
Master of Science in Logistics and Supply Chain Management
University of Portsmouth, United Kingdom,
geboren te Yala, Thailand

This dissertation has been approved by the promotor:
Prof. Dr. I. Horváth

Composition of the Doctoral Committee:

Rector Magnificus	Chairman
Prof. Dr. I. Horváth	Delft University of Technology

Independent members:

Prof. Dr. Y. Zeng	Concordia University, Canada
Prof. Dr. E. Du Bois	University of Antwerp, Belgium
Prof. Dr. B. Tekinerdogan	Wageningen University, the Netherlands
Assoc. Prof. Dr. C. Anutariya	Asian University of Technology, Thailand
Prof. Dr. R.H.M. Goossens	Delft University of Technology
Prof. Dr. F.E.H.M. Smulders	Delft University of Technology

Reserve member:

Prof. Dr. J.P.L. Schoormans	Delft University of Technology

This Ph.D. research was funded by the Scholarship of the Royal Thai Government.

Designing an active recommender framework to support the development of reasoning mechanisms for smart cyber-physical systems

Keywords: smart cyber-physical systems, application-specific reasoning mechanisms, active recommender framework, context-sensitive recommendation, automated parking assist system

Printed by Proefschriftmaken
Proofread by Brent Jones
Cover design by Sirasak Tepjit
Design and layout by Prangnat Chininthorn

Ph.D. dissertation
Delft University of Technology, Delft, the Netherlands
ISBN: 978-9-46-384299-0

An electric copy of this dissertation is available at the repository of Delft University of Technology

Summary

Designing an active recommender framework to support the development of reasoning mechanisms for smart cyber-physical systems

Background of the research

Modern engineered systems are becoming more and more intellectualized. This trend of the current development of CPSs called for a new classification and identification of the various generations of CPSs. This promotion research focused on the 2G-CPSs (also referred to as 'smart cyber-physical systems', S-CPSs). The built-in computational intelligence makes them capable of building awareness, reasoning about the objectives and states of operations, planning adaptations, and providing services even in dynamically changing contexts. Typically, S-CPSs apply one specific reasoning strategy and mechanism for simple problems and a combination of reasoning strategies for compound problems. Designing complex reasoning mechanisms (RMs) is a complicated task requiring a high-level abstraction and asufficiently comprehensive logical model. Due to the proliferation of S-CPSs, there is a growing need for different task-orientated, application-specific reasoning mechanisms (ASRMs). They should be in synergic connections with each other according to the logic of knowledge that they process and reason with. Designing ASRMs for CPSs with smart capabilities is a new issue both for systems research and for system development.

The advancement of technologies and the growing demand for applications offer more and more opportunities for designing smart systems. However, any rapid change in the technologies creates difficulty for system designers. If they are not equipped with the latest technological and methodological knowledge, their innovation potential and competitiveness are reduced. The need to support designing reasoning mechanisms for S-CPSs by computer-aided design systems can be considered from two aspects, (i) technology aspect and (ii) human aspect. From the aspect of technology, S-CPSs should be

based on the integration of multiple novel technologies. The issue and challenging nature of technology combinations should be resolved. From a human perspective, designers should be (i) protected against knowledge obsolesces and deficits, (ii) defended against unknown technologies and unmanageable complexities,and (iii) supported in solving their design tasks efficiently and reliably.

Research problems

Our research concentrated on a problem recently emerged related to S-CPSs. The essence of the problem is that S-CPSs are based on application-specific reasoning mechanisms (ASRMs) that enable them to generate context-dependent solutions for various application problems. The promotional research was conceptualized and conducted according to the research hypothesis that a computer-aided design support tool can be conceived as an 'active recommender framework' (ARF) for a compositional design of ASRMs of S-CPSs. The ARF can be characterized by multiple system-level functional features, from which the interrelated process monitoring and decision support functionalities have been considered in the promotion research. The research problems were addressed on three levels: (i) the entire phenomenon of supporting the design of ASRMs by an ARF; (ii) the services of an active recommender framework, in particular in the context of S-CPSs and the methodology to support the development of ARF, and (iii) the application context in which the ARF was supposed to provide recommendation services to support solving procedural and knowledge related problems, as an intellectualized assistive system.

Research methodology and activities

The whole research project was methodologically framed by a logical flow of four research cycles. Each research cycle addressed different aspects of the ARF development. The selected application context was a specific part of the design process of an automated parking assist system, as the target ASRM. The research activities included: (i) knowledge aggregation, demarcation of the domain of interest, and specification of requirements; (ii) functional and architectural conceptualization of the active recommendation framework; (iii) computational implementation and operationalization of the demonstrative modules; and (iv) validation of the usefulness of the recommendations generated by the implemented demonstrative modules of the ARF.

Research cycle 1

The objectives of the first research cycle were: (i) to get a deeper insight into the studied research phenomenon, (ii) to get an overview of state of the art based on the related scientific literature and professional web repositories,and (iii) to synthesize a starting 'home base' for the investigations and a knowledge platform that can be used follow-up developments. Knowledge aggregation included the study of the system engineering frameworks (SEFs) that were implemented and used to develop systemlevel reasoning in

the context of S-CPSs (i.e., system knowledge, situation awareness, context-sensitive reasoning, decision-making, and system adaptation). A comprehensive literature study was done by applying both quantitative and qualitative methods. The quantitative study aimed at exploring the landscape of publications related to the overall research phenomenon and the closely related specific phenomena. A bibliometric map was constructed based on a wide range of key terms.

According to the reasoning model, the qualitative analysis was narrowed down to three domains: (i) the domains that provided the context information for the research, namely: cyber-physical systems and system smartness, (ii) the domain of discourse of the research including the methodological details of framework development from multi-perspectives, and (iii) the domains that provided content information for studying frameworks. The requirements for the ARF were derived by considering the implications of the findings. The requirements were formulated in regular textual form and their relationships were explored and represented as semantic maps.

Research cycle 2

In this research cycle, we worked on a novel concept of the ARF for the development of ASRMs. At the first step, the operation of the ARF was deepened by setting up a comprehensive scenario for the design of ASRMs. Based on this, the service packages to be provided by the ARF were defined. The two essential mechanisms needed for the implementations of the functions of process monitoring and decision-support were devised. Concerning the process monitoring functionality, the conceptualization of the ARF was done according to the case of type B observation of an NUE. Towards systematic methodological approach, the ARF development process was modeled by the four-layer structure that included (i) specification of functionality, (ii) allocation of the functionality into system architecture, (iii) specification of computational algorithms and data structure, and (iv) organization of the operation workflow, including communication with the designer. The computational mechanisms of the ARF have been decomposed into six main functions. Assuming a one-to-one relationship, each function was mapped onto one architectural module: (i) facial expression-based non-usual event recognition (NUE-D); (ii) dialogue-based obstacle identification (DOI); (iii) construction of the reference process protocol (CRP); (iv) reference process protocol-based procedural obstacle identification (ROI); (v) advisory content generation (ACG); and (vi) designer's decision evaluation (DDE).

Research cycle 3

The goal of the third research cycle was to realize the two selected computational mechanisms of the ARF. The efforts were invested in (i) the implementation of the demonstrative modules of the ARF, and (ii) the application testing of the system-level functionality of the ARF in the context of an automated parking assist system (APAS). From a computational point of view, the implementation of the whole ARF had an innate complexity and it could

not be implemented in full scale. Therefore, with a view to the time and capacity available, only a demonstrative part was specified. Nevertheless, this demonstrative part was able to display the novel functional abilities of the fully-fledged implementation. The divide-and-conquer strategy was applied to avoid an uncontrollable complexity of the implementation. It involved using (i) a multi-layer structure, (ii) modular design technique, and (iii) object-oriented programming. This reduced the implementation efforts to four modules closely associated with recommendation generation. The implemented modules included: (i) the DOI module; (ii) the CRP module; (iii) the ROI module; and (iv) the ACG module. The functional validation of the demonstrative part was completed by focusing only on the design process elements in the working principle exploration (WPE) session of the APAS. Concretely, this aimed at the development of search algorithms for selecting the proper motion path for the actual parking problem. The testing of the functionality was done based on a scenario that included the design actions relevant for the WPE session.

Research cycle 4

The fourth research cycle focused on the quality of the recommendations provided by the ARF. Usefulness was chosen as the measure of the quality, and it was captured by indicators. The objective of the fourth research cycle was to validate the usefulness of the recommendation provided by the ARF. We aimed at examining how useful the provided recommendations were for the designer to overcome possible procedural obstacles in the design process. The concept of a synthetic validation agent (SVA) was introduced as the surrogate of designers. This proved to be an appropriate means to handle the situation. An SVA was conceptualized to simulate the decision-making behavior of (human) designers as a new validation means and approach. In our context, the SVA mimicked the flow of procedural decisions of the human designer as they were made after obtaining the recommendations. The expected outputof the behavioral simulation made by using the agent was a data set that included the patterns of the decisional behavior of the designer. We aimed at using this synthesized dataset to validate the usefulness of the individual recommendations.

Main findings and conclusions

Our main findings can be summarized as follows:

- From the viewpoint of a computational system, the ARF was proposed as a design action driven context-sensitive recommender system. The ARF is capable of (i) monitoring what is happening in the design process, (ii) identifying where a procedural obstacle is, and (iii) offering personalized recommendations to the designer to help proceed in the design process. This assumes not only monitoring of the process but also dealing with the information contents of the design activities.
- The main contribution of the demonstrative implementation was the context-sensitive recommendation generation by relying on the RPP. This process representation lent

itself to a systematic exploration of the potential design activity flows as well as to the investigation of the design process and its action elements. The hybrid inference was proposed as a novel functionality of the ARF system. It determined which design entities and methods had to be involved in the process analysis-based recommendation generation.

- The proposed ARF has been equipped with the abilities to support the design of ASRMs in the target application context of APAS. A ML-type algorithm has been developed based on typifying the design activities and representing the design activity flow in the RPP. The functionality test confirmed that the adapted ML-type algorithm was able to select the proper parking cases.
- In the evaluation of the usefulness of recommendations, the decisional modes had direct relations with the acceptance probability of the recommendation. A higher probability of acceptance offered a higher possibility of having a useful recommendation. A key issue was how to determine the optimal proportion of the common knowledge that was shared by the SVA-mimicked useful recommendation. A key issue was how to determine the optimal proportion of the common knowledge that was shared by the SVA-mimicked designer and the RPP. We argued that this information could be used to enhance the usefulness of recommendations.

Initially proposed by researchers of the hosting Section of Cyber-Physical Systems Design, the concept of an active recommendation framework as significant novelty and supposed to play an influential role in the future. The term *"framework"* was used to refer to a purposeful enabler that arranges and rationalizes design activities, information processing, and designer-system interaction. The term *"recommender"* expresses that, as a complex system, the ARF derives context-dependent advice for the designer based on a comprehensive system model of the concerned (specific) design process. The term *"active"* refers to the fact that the ARF continuously monitors the design process and spontaneously interacts with the designer wherever it is needed in the design process. Our conclusions have been that the ARF goes well beyond the concepts of traditional SEFs and static recommender systems.

Samenvatting

Achtergrond van het onderzoek

Moderne geconstrueer de systemen krijgen een steeds intellectueler karakter. Deze trend in de huidige ontwikkeling van cyberfysieke systemen (CPS'en) vroeg om een nieuwe classificatie en identificatie van de diverse CPS-generaties. Dit promotieonderzoek richtte zich op intelligente cyberfysieke systemen, ook wel afgekort tot 2G-CPS of S-CPS (smart cyber-physical systems). De ingebouwde computationele intelligentie stelt deze systemen in staat bewustzijn te ontwikkelen, doelstellingen en de toestand van bewerkingen te beredeneren, plannen aan te passen en diensten aan te bieden, zelfs binnen een dynamisch veranderende context. Gebruikelijk is dat S-CPS'en één specifieke beredeneringsstrategie en -mechanisme toepassen op eenvoudige problemen, en een combinatie van beredeneringsstrategieën op samengestelde problemen. Het ontwerpen van complexe beredeneringsmechanismen (reasoning mechanisms, RM's) is ingewikkeld en vereist een hoge mate van abstrahering en een logisch model dat voldoende omvattend is. De razendsnelle groei van S-CPS'en brengt een toenemende behoefte aan verschillende taakgeoriënteerde en applicatiespecifieke beredeneringsmechanismen (application-specific reasoning mechanisms, ASRM's) met zich mee. Deze dienen in synergische verbinding met elkaar te staan, uitgaande van de logica van de kennis die wordt verwerkt en beredeneerd. Het ontwerpen van ASRM's voor CPS'en met intelligente functionaliteit schept een nieuw vraagstuk op het gebied van zowel systeemonderzoek als systeemontwikkeling.

De voortschrijdende ontwikkeling van technologieën en de toenemende behoefte aan applicaties bieden steeds meer kansen voor het ontwerpen van intelligente systemen. Elke snelle technologische verandering brengt echter problemen voor systeemontwerpers met zich mee. Indien de meest recente technologische en methodologische kennis hierin niet is meegenomen, blijft hun innovatief en competitief potentieel beperkt. De behoefte aan ondersteuning bij het ontwerpen van beredeneringsmechanismen voor S-CPS'en door CAD-systemen (computer aided design) kan vanuit twee gezichtspunten worden benaderd: (i) het technologische gezichtspunt en (ii) het menselijke gezichtspunt. Vanuit technologisch gezichtspunt moeten S-CPS'en worden gebaseerd op integratie van meerdere nieuwe technologieën. Ze moeten een oplossing bieden voor het problematische, uitdagende karakter van technologische combinaties. Vanuit menselijk gezichtspunt dienen ontwerpers (i) te worden beschermd tegen veroudering en tekortkomingen van kennis, (ii) te worden beschermd tegen onbekende technologieën en onbeheersbare complexiteiten, en (iii) te worden ondersteund bij het op efficiënte en betrouwbare wijze oplossen van hun

ontwerpproblemen.

Onderzoeksproblematiek

Ons onderzoek richtte zich op een onlangs gesignaleerd probleem met cyberfysieke systemen (cyber-physical systems, S-CPS'en). De kern van het probleem is dat S-CPS'en gebaseerd zijn op applicatiespecifieke beredeneringsmechanismen (application-specific reasoning mechanisms, ASRM's) waarmee ze contextafhankelijke oplossingen voor diverse applicatieproblemen kunnen genereren. Uitgangspunt voor de conceptualisatie en uitvoering van het promotieonderzoek was de onderzoekshypothese dat een CAD-ontwerphulpmiddel kan worden opgevat als een 'actief aanbevelingskader' (active recommender framework, ARF) voor een compositioneel ontwerp van ASRM's voor S-CPS'en. Het ARF wordt gekenmerkt door meervoudige functionaliteit op systeemniveau, op basis waarvan de functionaliteit voor intergerelateerde procesbewaking en besluitvormingsondersteuning is meegenomen in het promotieonderzoek. De onderzoeksproblemen werden op drie niveaus benaderd: (i) het fenomeen van ontwerpondersteuning van ASRM's door een ARF in zijn totaliteit; (ii) de diensten van een actief aanbevelingskader, in het bijzonder in de context van S-CPS'en en de methodiek voor ondersteuning van de ontwikkeling van een ARF, en (iii) het toepassingsgebied waarbinnen het ARF aanbevelingsdiensten dient te leveren ter ondersteuning van het oplossen van procedurele en kennisgerelateerde problemen, in de vorm van een geïntellectualiseerd assistentiesysteem.

Onderzoeksmethoden en -activiteiten

Het onderzoeksproject als geheel werd methodisch ingekaderd binnen een logische stroom van vier onderzoekscycli. Iedere onderzoekscyclus richtte zich op specifieke aspecten van de ARF-ontwikkeling. Het geselecteerde toepassingsgebied vormde een specifiek onderdeel van het ontwerpproces van een assistentiesysteem voor geautomatiseerd parkeren als doel-ASRM. De onderzoeksactiviteiten omvatten: (i) kennisaggregatie, afbakening van het interessegebied en specificatie van eisen; (ii) functionele architecturale conceptualisatie van het actieve aanbevelingskader; (iii) computationele implementatie en operationalisatie van de demonstratieve modules; en (iv) validatie van de bruikbaarheid van de aanbevelingen die werden gegenereerd door de geïmplementeerde demonstratieve modules van het ARF.

Onderzoekscyclus 1

De doelstellingen van de eerste onderzoekscyclus waren: (i) een dieper inzicht verwerven in het bestudeerde onderzoeksfenomeen, (ii) een overzicht krijgen van de 'state of the art' op basis van de relevante wetenschappelijke literatuur en professionele webrepository's, en (iii) het synthetiseren van een 'uitvalsbasis' voor het onderzoekswerk en een kennisplatform dat kan worden gebruikt om ontwikkelingen te blijven volgen. De kennisaggregatie omvatte onderzoek naar de systeemontwerpkaders (system engineering frameworks, SEF's) die werden geïmplementeerd en gebruikt voor het ontwikkelen

van redeneringen op systeemniveau binnen de context van S-CPS'en (systeemkennis, situationeel bewustzijn, contextgevoelig redeneren, besluitvorming en systeemadaptatie). Er is uitgebreid literatuuronderzoek verricht met toepassing van zowel kwantitatieve als kwalitatieve methoden. Het kwantitatieve onderzoek richtte zich op de verkenning van het landschap aan publicaties met betrekking tot het onderzoeksfenomeen in algemene zin, alsmede nauw verwante specifieke fenomenen. Er werd een bibliometrische kaart geconstrueerd op basis van een breed spectrum aan kernbegrippen. Uitgaande van het beredeneringsmodel werd de kwalitatieve analyse toegespitst op drie gebieden: (i) de gebieden die de contextinformatie voor het onderzoek leverden, namelijk: cyberfysieke systemen en systeemintelligentie, (ii) het gebied van het discours over het onderzoek, met inbegrip van de methodologische details van kaderontwikkeling vanuit meerdere gezichtspunten, en (iii) de gebieden die contentinformatie leverden voor het bestuderen van kaders. De eisen voor het ARF werden afgeleid door te kijken naar de implicaties van de bevindingen. Deze eisen werden in standaard tekstvorm geformuleerd en de onderlinge relaties werden verkend en weergegeven als semantische kaarten.

Onderzoekscyclus 2

In deze onderzoekscyclus werkten we aan een nieuw ARF-concept voor de ontwikkeling van ASRM's. Als eerste stap werd de werking van het ARF verfijnd door het opzetten van een uitgebreid ontwerpscenario voor ASRM's. Op basis hiervan werden de door het ARF te leveren dienstenpakketten gedefinieerd. Voorts vond ontwikkeling plaats van de twee essentiële mechanismen die nodig waren voor de implementatie van de functies van procesbewaking en besluitvormingsondersteuning. Met betrekking tot de functionaliteit voor procesbewaking werd de conceptualisatie van het ARF uitgevoerd in overeenstemming met de casus van type B-observatie van een NUE. Om tot een stelselmatige methodologische aanpak te komen werd het ARF-ontwikkelingsproces gemodelleerd volgens een vierlaagse structuur die het volgende omvatte: (i) specificatie van functionaliteit, (ii) toewijzing van de functionaliteit aan systeemarchitectuur, (iii) specificatie van computationele algoritmen en gegevensstructuur, en (iv) organisatie van de operationele workflow, inclusief communicatie met de ontwerper. De computationele mechanismen van het ARF werden opgesplitst in zes hoofdfuncties. Uitgaande van een één-op-éénrelatie werd iedere functie toegewezen aan één architectuurmodule: (i) gezichtsexpressie-gebaseerde NUE-D (non-usual event recognition, herkenning van ongebruikelijke gebeurtenissen); (ii) DOI (dialooggebaseerde obstakelidentificatie); (iii) CRP (constructie van het referentieprocesprotocol); (iv) ROI (op het referentieprocesprotocol gebaseerde procedurele obstakelidentificatie); (v) ACG (adviescontentgeneratie); en (vi) DDE (designer's decision evaluation, evaluatie van de beslissing van de ontwerper).

Onderzoekscyclus 3

Doel van de derde onderzoekscyclus was het verwezenlijken van de twee geselecteerde computationele mechanismen van het ARF. Er werd gekeken naar (i) de implementatie

van de demonstratieve modules van het ARF en (ii) applicatietests van de functionaliteit op systeemniveau van het ARF binnen de context van een geautomatiseerd parkeerassistentiesysteem (automated parking assist system, APAS). Vanuit computationeel oogpunt bezat de implementatie van het ARF als geheel een inherente complexiteit die implementatie op volledige schaal onmogelijk maakte. Met het oog op de beschikbare tijd en capaciteit werd dan ook alleen een demonstratief deel gespecificeerd. Niettemin bleek dit demonstratieve deel in staat de nieuwe functionele mogelijkheden van een volwaardige implementatie zichtbaar te maken. De verdeel-en-heersstrategie werd toegepast teneinde onbeheersbare complexiteit van de implementatie te vermijden. Dit omvatte het gebruik van (i) een meerlaagse structuur, (ii) een modulaire ontwerptechniek, en (iii) objectgeoriënteerde programmering. Hierdoor bleef de implementatie beperkt tot vier modules die nauw verband hielden met het genereren van aanbevelingen. De geïmplementeerde modules omvatten: (i) de DOI-module; (ii) de CRP-module; (iii) de ROI-module; en (iv) de ACG-module. De functionele validatie van het demonstratieve deel werd afgerond door ons uitsluitend te concentreren op de ontwerpproceselementen binnen de WPE-sessie (working principle exploration) van het APAS. Concreet richtte deze zich op de ontwikkeling van zoekalgoritmen voor het selecteren van het juiste bewegingstraject voor het daadwerkelijke parkeerprobleem. Testen van de functionaliteit vond plaats op basis van een scenario waarin de relevante ontwerpacties voor de WPE-sessie waren opgenomen.

Onderzoekscyclus 4

De vierde onderzoekscyclus richtte zich op de kwaliteit van de aanbevelingen die door het ARF waren gedaan. Bruikbaarheid werd gekozen als maat voor de kwaliteit en werd vastgelegd door middel van indicatoren. Het doel van de vierde onderzoekscyclus was het valideren van de bruikbaarheid van de door het ARF verstrekte aanbeveling. We wilden onderzoeken hoe bruikbaar de geleverde aanbevelingen voor de ontwerper waren voor het wegnemen van mogelijke procedurele obstakels in het ontwerpproces. Het concept van een SVA (synthetic validation agent) werd geïntroduceerd als vervanging van ontwerpers. Dit bleek een geschikte methode te zijn om de situatie aan te pakken. Er werd een SVA geconceptualiseerd teneinde het besluitvormende gedrag van (menselijke) ontwerpers als nieuwe methode en benadering van validatie te simuleren. Binnen onze context simuleerde de SVA de stroom van procedurele beslissingen van de menselijke ontwerper na het verkrijgen van de aanbevelingen. De verwachte output van de gedragsmatige simulatie met gebruikmaking van de agent was een dataset waarin de patronen van het besluitvormingsgerichte gedrag van de ontwerper waren geïntegreerd. We wilden deze gesynthetiseerde dataset gebruiken om de bruikbaarheid van de afzonderlijke aanbevelingen te valideren.

Belangrijkste bevindingen en conclusies

De belangrijkste bevindingen kunnen als volgt worden samengevat:

- Vanuit het gezichtspunt van een computationeel systeem werd het ARF voorgesteld

als ontwerpactiegestuurd, contextgevoelig aanbevelingssysteem. Het ARF is in staat tot (i) het bewaken van wat er in het ontwerpproces gebeurt, (ii) het identificeren van een procedureel obstakel, en (iii) het bieden van gepersonaliseerde aanbevelingen aan de ontwerper ter ondersteuning van de voortgang van het ontwerpproces. Hierbij wordt niet alleen verondersteld dat er procesbewaking plaatsvindt, maar ook dat de inhoudelijke informatie met betrekking tot de ontwerpactiviteiten wordt verwerkt.

- De belangrijkste bijdrage van de demonstratieve implementatie was het genereren van contextgevoelige aanbevelingen op basis van het RPP (referentieprocesprotocol). Deze procesrepresentatie bleek geschikt voor stelselmatige verkenning van potentiële ontwerpactiviteitsstromen, maar ook voor het analyseren van het ontwerpproces en de bijbehorende actie-elementen. De hybride gevolgtrekking werd voorgesteld als nieuwe functionaliteit van het ARF-systeem. Deze bepaalde welke ontwerpentiteiten en -methoden in het procesanalyse-gebaseerd genereren van aanbevelingen dienden te worden meegenomen.
- Het voorgestelde ARF is uitgerust met functionaliteit ter ondersteuning van het ontwerp van ASRM's binnen de doelapplicatiecontext van APAS. Op basis van de typering van de ontwerpactiviteiten en het representeren van de ontwerpactiviteitsstroom in het RPP werd een algoritme van het ML-type ontwikkeld. De functionaliteitstest bevestigde dat het aangepaste algoritme van het ML-type in staat was de juiste parkeersituaties te selecteren.
- Bij de evaluatie van de bruikbaarheid van de aanbevelingen waren de besluitvormingsmodi rechtstreeks gerelateerd aan de waarschijnlijkheid van acceptatie van de aanbeveling. Een hogere mate van waarschijnlijkheid van acceptatie vergroot de kans op een bruikbare aanbeveling. Een belangrijk probleem was hoe te bepalen wat het optimale aandeel moest worden van de algemene kennis die door de SVA-gesimuleerde ontwerper en het RPP werd gedeeld. Wij stelden dat deze informatie kon worden gebruikt om de bruikbaarheid van aanbevelingen te vergroten.

Het concept van een actief aanbevelingskader, dat in eerste instantie werd voorgesteld door de als gastheer fungerende afdeling Cyber-Physical Systems Design, is een baanbrekende nieuwe ontwikkeling die naar verwachting een belangrijke rol zal gaan spelen in de toekomst. De term *"kader"* verwijst hier naar een doelgerichte enabler die ontwerpactiviteiten, informatieverwerking en de interactie tussen ontwerper en systeem rangschikt en rationaliseert. De term *"aanbeveling"* duidt erop dat het complexe ARF-systeem contextafhankelijk advies voor de ontwerper afleidt op basis van een uitgebreid systeemmodel van het betreffende (specifieke) ontwerpproces. De term *"actief"* verwijst naar het feit dat het ARF het ontwerpproces continu bewaakt en telkens spontaan interactie met de ontwerper aangaat wanneer dit noodzakelijk is binnen het ontwerpproces. Wij kwamen tot de conclusie dat het ARF de concepten van traditionele SEF's en statische aanbevelingssystemen in ruime mate overtreft.

Table of contents

CHAPTER 2: AGGREGATION OF KNOWLEDGE AND EXPLORATION OF REQUIREMENTS

Chapter 1

Introduction

1.1. Background of the research

1.1.1. Manifestation and evolution of cyber-physical systems

Modern engineered systems are becoming smart. Cyber-physical systems (CPSs) have the affordances to behave as general smart systems. The notion of '*cyber-physical systems*' was introduced in 2006 by researchers previously working in related fields such as embedded systems, advanced robotics, real-time systems, hybrid systems, and control systems [1]. Although numerous papers have been published on functional and architectural definitions of CPSs, achieving a shared understanding has been difficult due to the different backgrounds and viewpoints of researchers [2]. There is currently no agreement on the exact definition of CPSs, but they are essentially understood to be systems that closely integrate constituents from the cyber and physical domains [3]. The cyber components are discrete, logical, and connected, and responsible for computation, communication, and control through a network of sensors and actuators. The physical components operate in continuous time and are responsible for changing material and energy flows, system states, and stakeholder and environment interactions. CPSs can be applied to various domains such as manufacturing, transportation, infrastructure, healthcare, and defense.

The current state of advancement of information technology and communication systems enables system nodes to connect and communicate with the other systems and their environments. CPSs may be implemented on various scales, ranging from the nano-world to large-scale systems of systems [4]. This feature enables an ensemble of CPSs to manifest as a system of systems (SoSs), which are supposed to dynamically adapt to the changes in the environment and manage the resources that are needed to achieve the shared

operational objectives and performance [5]. On the other hand, there is an increasing need for a higher-level intelligence for CPSs in order to deal with increasing uncertainty and unpredictable situations. This is also needed because of unforeseen dynamic and emerging behaviors that can occur during runtime operation [6].

The characteristics of recent CPSs are gradually evolving beyond what were identified by the early definitions of CPSs. They are becoming non-composite, open, smart, autonomous, agent-like, resilient, adaptive, evolutional, and replicative [7]. Engell et al.indicated a shift in the upcoming generation of CPSs towards more sophisticated operations, which poses research challenges such as [8]: (i) situation awareness in large distributed systems with decentralized management and control; (ii) handling large amounts of data in real life to monitor the system performance and to detect faults and degradation; (iii) learning good patterns from past examples, autoreconfiguration, and adaptation; and (iv) monitoring user behaviors, analysis of needs, and detecting anomalies. Due to the increasing need to clarify the theoretical, methodological, and computational issues of system smartness, the above topics have been identified as objectives in various branches of CPSs research.

In our view, CPSs must be seen as networked, knowledge-intensive, and multi-actor systems. On the other hand, the literature offers only very few progressive classifications of CPSs. The trends and traits of the current development of CPSs call for new classification and identification of the upcoming generations of CPSs. The changing functional and control paradigms indicate that they can be assumed to evolve through generations. The generations can be classified based on two aspects: (i) the level of intelligence, and (ii) the level of organization [9]. The identified generations of CPSs are shown in Figure 1.1. This reflects the reasoning model of our research team, which has been dealing with cognitive engineering of cyber-physical systems.

In the light of this reasoning model presented in [9], conventional (plant-type) CPSs are deemed the first-generation CPSs (1G-CPSs). They are self-regulatory and self-tuning systems. Thus, an embedded system is seen as a representative of the zeroth generation CPSs (0G-CPSs). It applies to look-alikes, embedded systems, and partial implementation of CPSs, which are regulated by feedback-based control sub-systems. Self-awareness and self-adaptation are the distinctive features of second-generation CPSs (2G-CPSs), which are often referred to as smart-CPSs. Smartness is regarded as a system-level characteristic of these systems. The third-generation CPSs (3G-CPSs) are self-cognizance and self-evolution systems. The fourth-generation CPSs (4GCPSs) predicted to behave as self-conscious and self-reproducing systems.

1.1.2. Paradigmatic features of smart CPSs

A paradigmatic feature is a system-level feature that refers either to a logically-based or a physically-based abstraction of a system as a whole. The set of paradigmatic features differentiate one generation or manifestation of CPSs from other comparable systems [10]. As mentioned above, self-regulation and self-tuning are the distinguishing paradigmatic features of the first generation of CPSs. Basically; CPSs are regulated by a feedback control

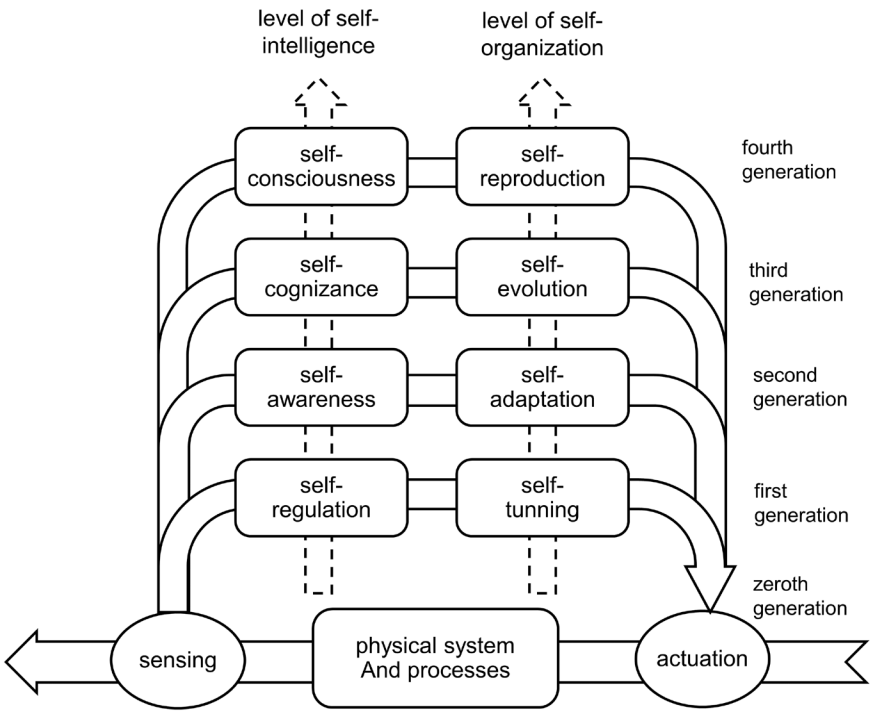


Figure 1.1: Generations of CPSs (taken over from [9])

loop that means they are the self-regulated systems by nature. Self-tuning is the capability of managing performance and resource allocation in order to satisfy the requirements of different users [11]. These systems include algorithms and software components to enable the feedback control loop. The second-generation CPSs have been distinguished by self-awareness and self-adaptation as paradigmatic features. Self-awareness is a characteristic of those CPSs that are able to understand a changing situation based on the extent of information available. When a system can realize what the most possible situation is, it can respondto the situation accordingly. In fact, as situations are changing over time, second-generation CPSs should be able to adapt themselves to deal with the dynamic situations while maintaining a level of performance, or even improving it when the systems are confronted repeatedly with similar situations.

According to [9], the third-generation CPSs are going to be characterized by selfcognizance and self-evolution. This makes them self-supervised. Self-cognizance is a new term in the field of CPSs. It is a cognitively higher-level ability than selfawareness. It can be interpreted in the following way: While system awareness can build a world model in a given operational situation, system cognizance is supposed to be able to differentiate the different situations and develop multiple models from various perspectives. Thus, self-cognizance is the capability of recognizing and building awareness of the operational situations in a given local world and proposing and adopting a finite number of situated

operation models. The term ‘evolving systems’ refers to systems that can adjust themselves according to dynamic or even evolving environments based on life-long learning and adaptation. Practical examples of 3G-CPSs are expected from the development of AI-enabled complex problem solving systems.

The distinguishing paradigmatic features of 4G-CPSs are self-consciousness and selfreplication. Self-consciousness is a manifestation of human-like consciousness. It is deemed to be a capability that allows the system to view and model itself based on objectives, operations, experiences, and social relationships. Although there have been attempted to create machine consciousness, what we have today is far from humanconsciousness. For instance, in papers [12]-[15] a replica of human biological consciousness is sought after in the form of machine consciousness. Many researchers have considered the human brain as an analogue computing device. The most fundamental mechanism of consciousness has not been explained yet. Pragmatic theories view it as a result of the operation of a series of integrated bio-physiological, psychological, and cognitive activities

Machine consciousness, as well as robot consciousness, is reduced to the process of informing, reasoning, and computation with digital data. However, from the perspective of the 4G-CPSs, the notion of consciousness is not elaborated sufficiently either from an ontological point of view, or from a methodological point of view. Just as with humans, consciousness of the 4G-CPSs should operate as an all underpinning phenomenon and work simultaneously with an infinite number of world models. Selfreplication is another paradigmatic feature of 4G-CPSs. Self-replication is an essential feature in the context of living things [16]. However, no system can self-replicate itself without being in the necessary intellectual conditions and having the necessary resources [17]. Self-replication should be seen as an emergent (non-preprogrammed, but conditioned) property and an essential characteristic of 4G-CPSs.

As far as the necessary intellectual conditions and resources are concerned, knowledge acquisition and generation by reasoning and learning are of paramount importance. Sophisticated mechanisms of these are not yet clarified theoretically or computationally. It is unclear if they can be derived based on an extrapolation from the current mechanisms. For instance, networked CPSs interact as distributed systems and share knowledge with each other. The individual bodies of knowledge shared by the component systems can be further synthesized and used to create new systems, which are cognitively enhanced versions of the predecessors. Together with selfconsciousness and self-replication, this form of knowledge synthesis also contributes to the progression of the paradigms of CPSs towards 4G-CPSs. Only the future will tell how artificial general intelligence will influence their self-consciousness and selfreplication potentials.

The classification of CPS generations gave opportunity for the research team to envisage a research and development roadmap for the next generation CPSs. In view of this, we focus on 2G-CPSs referred to as ‘Smart Cyber-physical systems’ in this research. These systems are equipped with such level of computational intelligence that makes them

capable of building awareness, reasoning about the fulfilment of the objectives, the results of completing operations, the state of the system, and the necessity and possibility of self-adaption.

1.1.3. Complexification of the functionality of S-CPSs

CPSs embed computational devices for, among other things, (i) physical or visual sensing, (ii) processing and storing data, (iii) energy harvesting, and (iv) wired/wireless communication [18]. The generalized operation process of firstgeneration CPSs typically starts with detecting, monitoring, and streaming sensor data. The computing parts process the input data, and the results are channeled to the effector parts, which actuate the intended state of the physical components and processes. These generic tasks are included in the Sensing-Processing-Actuation loop [19]. This closed-loop operation is a simplification that cannot describe the operation process of open networked CPS configurations, as additional functional nodes may join or may leave the ensemble. Wireless sensor networks can provide both local and remote control over the networked devices [20]. Data can be accessed at the device and network levels. These technological opportunities enhance both the flexibility and the expandability of CPSs [21], while they also allow for timing variability andstochastic behavior [22]. Harmonized operation of all constituents requires a network managing function.

Using Internet of Things resources, CPSs can operate as networked information systems [23]. The quality of the data collected from different sources is usually location and time dependent. Extension of CPSs increases the amount of the to-beprocessed dataexponentially [24]. The acquired data may be stored both as structured and as unstructured data. Extracting information and deriving patterns from massive unstructured data requires dedicated data analytics and data mining [25], [26]. In addition, a wide variety of simulation models – ranging from low-level physical signals through high-level data constructs to abstract events – are also needed.

Processing data or pieces of information on statistical or syntactic levels might not be sufficient to support cognitive capabilities of S-CPSs. For example, more is needed for situation-awareness, decision-making, and problem-solving. The data should be transformed into semantically rich constructs, context information, and applications pecific knowledge. Situated reasoning and semantic reasoning are the two basic functions to ensure that data arriving from several sources are processed with the same meaning and communicated consistently over all processes [27], [28]. In addition, many other reasoning, learning, and integration processes are needed to make CPSs truly smart.

Considering the abovementioned operations, a generic function diagram of S-CPSs is shown in Figure 1.2. All functions are co-dependent; changing one of them in design will directly impact the other functions of the concerned system [29]. This is becoming a serious issue as S-CPSs is becoming functionally and architecturally more complex. It means that designers should take care of a high number of interactions among the components inside and outside the S-CPSs and should pay attention to the increase of

complexity of the embedding environment. These characteristics render many traditional design methodologies inadequate or irrelevant, especially if they focus on a separation of concerns during the system design and implementation processes. It is a new challenge for designers to think about heterogeneous and complex systems (including hardware, software, and cyberware constituents) in a holistic manner, and to take all functions and functional interaction into account. The fact of the matter is that, in addition to functions, the computationally implemented operations and the behavior expected under regular and irregular circumstances should also be given attention. The result is an extra mental and professional load on the designers.

1.1.4. The need to support designing reasoning mechanisms for S-CPSs

The advancement of the technologies and the growing demand for application offer more and more opportunities for designing smart systems. However, any rapid change in the technologies creates difficulty for a system designer..If the designers are not equipped with the latest technological and methodological knowledge, their innovation potential and competitiveness are reduced. Continuous learning helps, but it also takes a lot of time from the creative work and cooperation. These are all reasons why both researchers and managers have recognized that designers require effective system support in their processes. This also concerns designing application specific reasoning mechanisms for smart CPSs, which are also getting more complex and sophisticated. The need for the support of designing reasoning mechanisms for S-CPSs by computer aided design systems can be considered from two aspects, (i) technology aspect, and (ii) human aspect. From the aspect of technology, as mentioned above, S-CPSs should be based on multiple novel technologies and the challenging nature of technology combination should be resolved. From a human perspective, designers should be (i) protected against knowledge obsolesces and deficits, (ii) defended against unknown technologies and unmanageable complexities, and (iii) supported in solving their design tasks in an efficient and reliable manner.

The above argumentation underpins the emerging need for non-conventional design support tools. The non-conventionality means that they may utilize novel design principles and operationalize novel approaches to providing support. The support systems should provide tailored and convenient support for the designer in an effective (visible) manner without diverting the attention of the designer from the work. In the history of office automation systems, there are many examples of useful support solutions. For instance, implementation of a spell checker function in a text/document editing programs (such as a word processor, chatting apps, or search engine) is a relevant example. A spell-checker automatically checks and corrects for misspellings or even grammatical errors and style issues in a text. Technically, there are two functions included in this service, namely (i) informing function, and (ii) corrective function. The former function interprets the meaning of an unusual word and offers synonyms and application examples. The latter function automatically corrects observed typing errors without any human interaction and rephrases a complete sentence if a more obvious or appealing formulation is possible. The active recommendation framework tried to apply this analogy in a more complex context, i.e., in

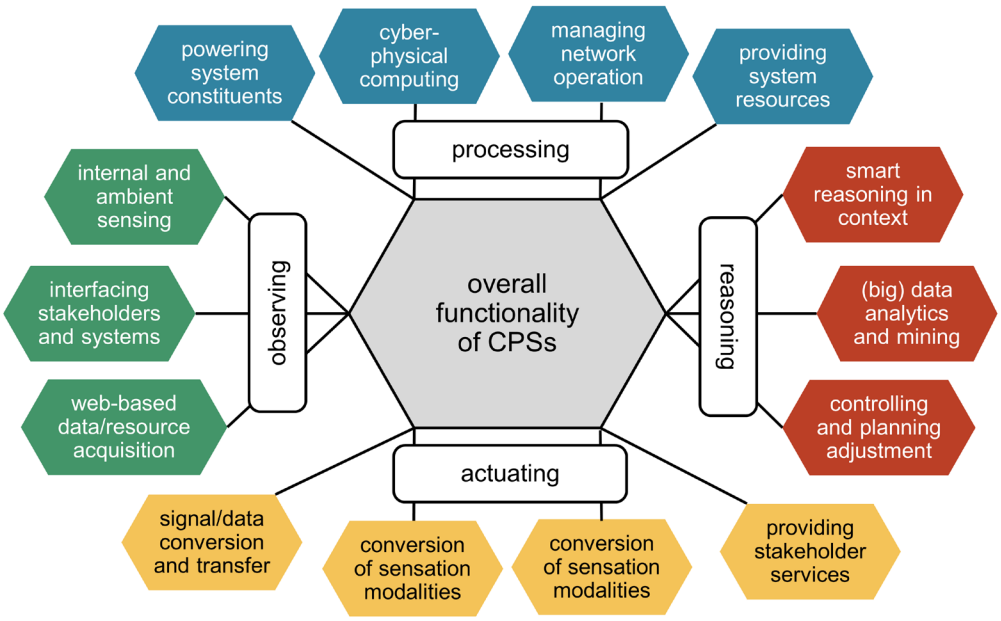


Figure 1.2: Generic functionality of S-CPSs (based on [29])

the support of designing application specific reasoning mechanism for S-CPSs – a problem which soon moves out from academic research laboratories and into industrial design offices.

The computational support of designers has been based on the following consideration. In the context of designing, an informing function is needed when the designer needs information about an existing system component, a technological process, a design action, or a corresponding part of the design problem. The corrective function is activated when the designer generates content and commits content and procedural errors in the design process. Our active recommender framework concept rests on these conceptual ideas. A novel ARF functionality is proposed in terms of integration of the dedicated informing functions and corrective functions. The active recommender framework was intended to offer two types of the recommendation services. Providing content related recommendations represents the informing part of support. Providing procedure related recommendations in various situations represents the corrective part of support. There are many challenges to take into account at applying the concept of ARF in the development of a design support tool for the above-described purpose. The most significant ones are (i) the novelty of the software technologies, (ii) inherent functional complexity, (iii) conceptual interrelationships and dependences, (iv) technological complicatedness and technology sensitivity, (v) the design orientation and the non-standard design approach, and (vi) the integration of all demands in a holistic approach.

1.2. Description of the addressed research challenges

1.2.1. The essence of designing reasoning mechanisms for S-CPSs

Reasoning is the logical process of drawing a specific conclusion by utilizing human problems-solving strategies [30]. Typical instances of logic-based reasoning are deductive reasoning, inductive reasoning, case-based reasoning, probabilistic reasoning, fuzzy reasoning, intuitivist reasoning, and analogical reasoning. In addition to these computational (evaluational) approaches of reasoning, the literature also shows interpretative (explanatory) approaches, and the juxtaposition of these two approaches. The human mind can handle both of them very well, but interpretative reasoning is extremely difficult to implement on computers that do not have an intrinsic sense of semantics and meaning in context. There were sophisticated reasoning strategies proposed, which assume data exchange with physical processes or within a network of connected devices, and application specific information. One example is procedural abduction [31]. The abovementioned three categories, as well as, their purpose-driven combinations, are regarded as potential reasoning strategies for SCPSs. These S-CPSs are supposed to be able to extract, collect, and combine data and information even from a dynamically changing, noisy, and uncertain physical environments, and to convert them into useful system cyberware (useable knowledge) in real time.

Conceptually, S-CPSs are supposed to apply one specific reasoning strategy for simple problems or a combination of reasoning strategies for compound problems. Designing complex reasoning mechanisms is a complicated task that needs a high-level abstraction and a sufficiently comprehensive logical model. It should be guaranteed that the knowledge provided for problem solving by the reasoning mechanisms is 'fitfor-purpose', that is, it (i) has the power to address the task, (ii) is coherent and consistent with regards to its elements, and (iii) can be used for the computational scheduling and interlinking of the algorithms included in the mechanisms. As mentioned above, a representative example of this is procedural abductive reasoning that involves multiple procedural elements, such as (i) ambient sensing, (ii) event recognition, (iii) building awareness, (iv) dynamic contexts evaluation, (v) situated reasoning, (vi) operation strategy planning, (vii) selective decision-making, (viii) functional and/or structural adaptation, and (ix) actuating effectors [31]. A metaframework of this complex operation and control is shown in Figure 1.3.

Operationally, the manifestation of procedural abductive reasoning (PAR) mechanisms in S-CPSs starts with the acquisition of raw data with sensor devices from several sources, and continues with a combination of syntactic and semantic data processing, recognizing events based on context information, inferring and identification of a situation, and learning from dynamic shifts of situations. The transformations of the knowledge throughout the stages of reasoning require appropriate computational methods and algorithms. These need to be individually selected, adapted, or developed. For the compositional operation of the PAR, a two-level control is required. The operation controller (which activates the necessary reasoning algorithms) provides the lower-level control, whilst the strategic

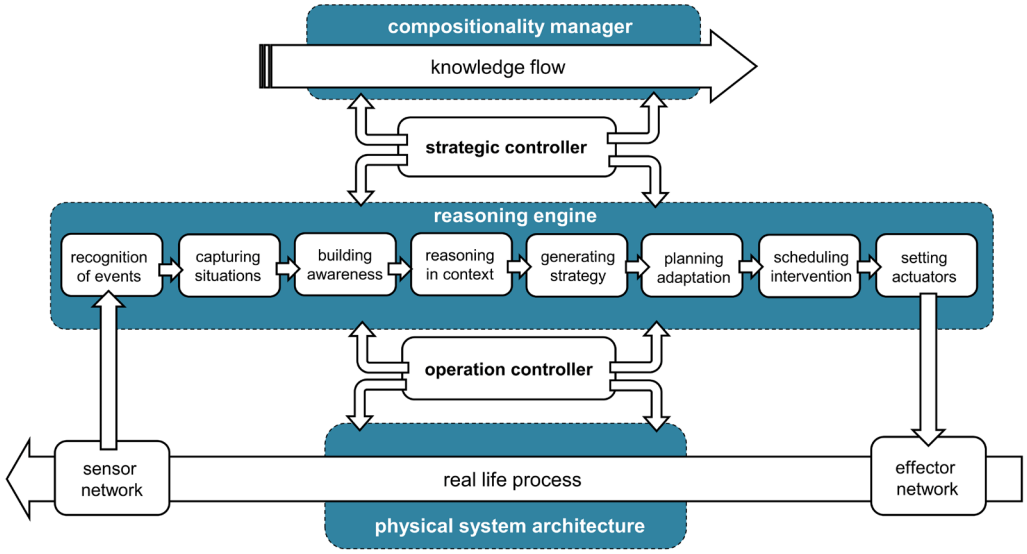


Figure 1.3: A meta-framework for procedural abduction as a reasoning mechanism for S-CPSs (modified from [31])

controller (which arranges the actual contents of the reasoning engine) provides the higher-level control. The cognitive capability of the reasoning mechanism may eventually lead (or is likely to lead) to novel useful information and/or knowledge concerning the task at hand and the solution process, in addition to those available at the start of the reasoning process.

As a first design activity in the process of reasoning mechanism development (RMD), the smart reasoning components are conceptualized on the system level and these specifications are used as contracts for the implementation of the components. To support this procedure, various architectural and behavioral models are created and used concurrently. As a result, S-CPSs will implement a form of computational creativity, at least in terms of a combined use of reasoning, decision-making, and adaptiveness in specific contexts. It is practically impossible to implement these reasoning enablers by one single computational means – a fact that calls for multiple interoperating reasoning mechanisms.

1.2.2. Application-independent versus application-specific reasoning mechanisms for S-CPSs

The reasoning mechanisms of S-CPSs, which are used in real-life operations, can be sorted into two types. The first is application-independent reasoning mechanisms (AIRM) – which are content-independent software systems doing generic inference and problem-solving without referring to any application contexts. Typical examples are, for instance, rule-based reasoning, case-based reasoning, and probabilistic reasoning. The second is application specific reasoning mechanisms (ASRM) – which are content-dependent software systems designated for managing a particular application. This section discusses the differences

between these two types of reasoning mechanisms in detail, as well as the challenges of designing ASRMs. The comparison of characteristics of AIRM and ASRM is shown in Table 1.1.

1.2.2.1. Application-independent reasoning mechanism

Often replaced by the acronym AIRMs, the term application-independent reasoning mechanisms is known from the research done in the field of artificial narrow intelligence. These reasoning mechanisms are often also referred to as general problem solvers [32]. AIRMs are problem-driven (rather than application-driven) configurations of directly coupled computational algorithms such as a production rules-based reasoning engine [33], or a procedural abduction mechanism [31]. In addition to the proprietary algorithms, they also include standard computational algorithms. An AIRM executes a logically complete reasoning process that is needed to solve a given inference, reasoning, or decision-making problem. AIRMs can be implemented in various forms, for example, as deterministic mechanisms (such as rule-based or analogy-based reasoning), or as probabilistic mechanisms (such as neural networks, Bayesian classifiers, and hidden Markov models). Procedural reasoning systems implement multiple sense-plan-act loops. Knowledge-intensiveness of AIRMS is increased either by task-specific knowledge repositories, or by computational ontologies. Ontology-based reasoning uses various language-specific representations of specifications of conceptualizations of entities and their relationships [34].

The AIRMs are utilized by putting them in the application context. Figure 1.4, for example, shows the conventional rule-based and case-based reasoning applied in the reasoning mechanism of the computer aided detection and diagnosis (CAD&D) system [33]. The rule

Table 1.1: Comparison of application independent reasoning mechanisms and application specific reasoning mechanisms

aspects	AIRM	ASRM
theoretical basis	formal logic and inferring	complex semantic computational procedures
purpose	general purpose inference without context	domain-specific inference with consideration of context
reasoning process	relying on a single or a hybrid reasoning process	require multiple forms of reasoning processes
Computational implementation	either composable or compositional configuration	compositional configuration
problem solving method	task and goal-driven solution	purpose and context-driven solution
knowledge handling	uniform representation of knowledge	several forms of knowledge representation are combined
design methodology	able to comply the conventional design methodology	require an adaptive design methodology

base stores the experience of a doctor's diagnosis, and the case base stores some typical cases in the diagnosis task. Using a Bayesiannetwork, the system recommends the most likely tasks for the doctor according to the diagnosis results, and simultaneously providing a reference for medical decision-making. The generality of AIRMs is an advantage in development, but it results in a disadvantage from the perspective of applicability. This has become obvious with the appearance of multiapplication oriented smart cyber-physical systems and/or systems of systems S-CPSs and/or S-CPSoSs

1.2.2.2. Application-specific reasoning mechanisms

We introduced the term 'application-specific reasoning mechanism' to refer to complexapplication dependent computational mechanisms. As the term implies, ASRMs are application-driven (or application context-driven) configurations of directly coupled computational algorithms that are (i) tailored to specific application objectives and tasks, (ii) not exclusively reliant on one of the conventional forms of reasoning (e.g., logical, semantic, procedural, qualitative, probabilistic, analogical, etc. reasoning), and (iii) benefiting from the background information, underpinning knowledge, and context information of the problem. As a combined software and cyberware, an ARSM enables S-CPSs to solve concrete real-life problems based on data elicited from the target application. Computationally, the reasoning process of ASRMs is a continuous runtime-activated sequence of the operations of software components. Typical examples are (i) automated parking assist systems, (ii) autonomous mobile robots, (iii) and manufacturing execution system (MES) which are based on context-dependent logical, cyber-physical, spatial, temporal, etc., reasoning.

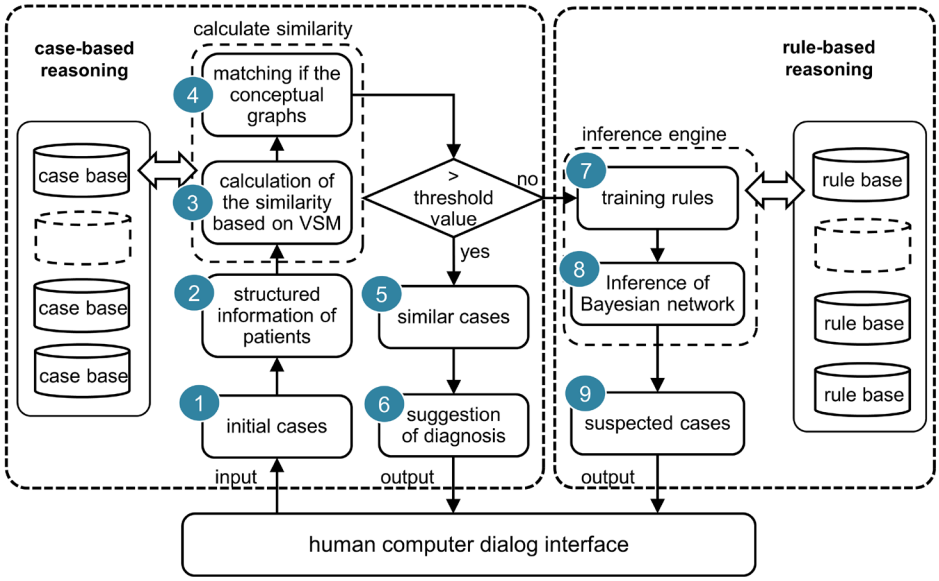


Figure 1.4: Workflow of the reasoning mechanism of the CAD system supported by rule-based and case-based reasoning (Modified from [33])

One example of an ARSM is shown in Figure 1.5. The proposed reasoning mechanisms were architecturally constructed by three interconnected modules (e.g., semantic modelling module, semantic information processing module, and semantic autonomous navigation) to enable the robot to perform cognitive tasks. The procedural reasoning process converts the sensors' data with the sematic information processing to endow the robot with cognitive vision capability. The visual object recognition encompasses metric information, geometric features, and image information that can be measured using a set of sensors. Then, the inference is performed to localize the place using the objects present in the surrounding environment. It leverages the recognized objects with the semantic information stored in its database. In the next stage, the mission task planning is performed through a sequence of necessary actions (e.g., behavior planning, motion planning, and task planning). These tasks were performed by using different types of algorithms. The interoperation of thesoftware components should be guaranteed at runtime in the dynamic situations. These components are also needed to be modified or require an interface to couple them seamlessly and the processes are influenced by both structural adaptation and implementation constraints. This characteristic of ASRMs is known as compositionality. It is one of the challenges at designing an ASRM which may not be taken into account when designing an AIRM.

1.2.3. The issue of compositionality in an application- specific reasoning mechanism

Designing ASRMs differs conceptually from designing AIRMs. It goes beyond a conventional composability orientated approach that systems can be composed in a bottom-

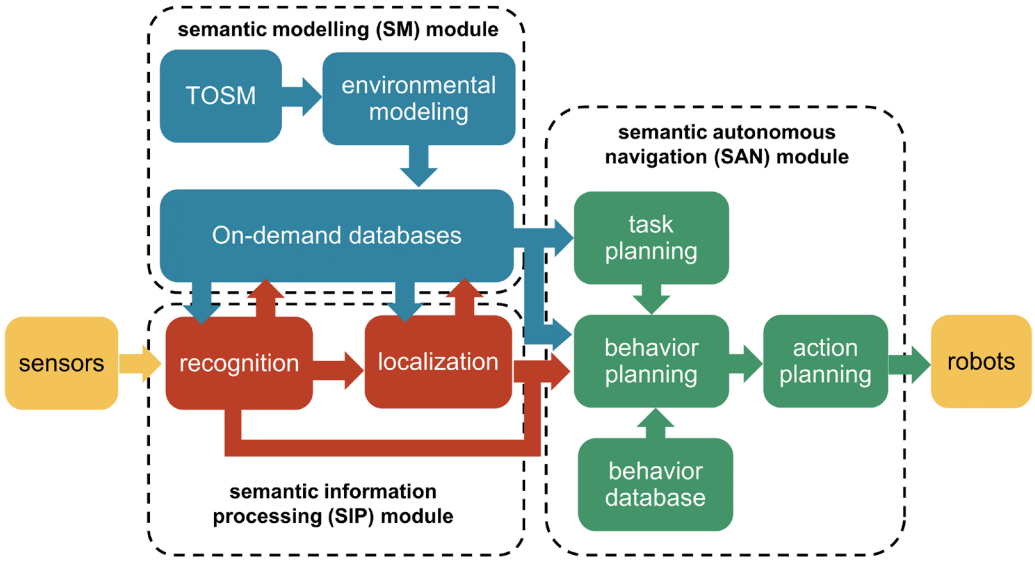


Figure 1.5: Interrelationships of the architectural components of an ARSM for an intelligent robot (based on [35]).

up manner by interfacing non-adaptable components [36]. Compositionality was introduced as a fundamental system manifestation principle for the development of reasoning mechanisms that intend to create a synergy among the functional elements of the systems in order to realize system-level smartness [37]. The hypothesis that compositionality is necessary in the case of smartly behaving S-CPSs originates in the fundamental assumption of classicalcognitive science, that complex mental representations are compositional.

ASRMs are multi-functional and knowledge-integrated systems. They are supposed to be a compositional arrangement of multiple computational algorithms where the wholeness (e.g., completeness and orientatedness) of problem solving creates not-composable interrelationships among the algorithms. The compositionality regarding ASRMs is assumed to be manifested in two levels, (i) concerning operation level, which should ensure that the interoperation of software components will be synergistic at runtime, and (ii) concerning system-level, which should confirm that the transformation of required knowledge and information has been done throughout the entire computational process for multi-task problem solving. Hence, the requirements for ASRMs will be defined based on the overall objectives and role of the planned S-CPSs at design time. The computational components (executables) are in compositional relationship since they should connect and work in a synergic way to meet the objectives of operation. This also applies to the chunks of the process control and problem solving knowledge required by the ASRMs since they are semantically interrelated with each other and should be composed into a proper body of knowledge at runtime [38].

1.2.4. Challenges of designing application-specific reasoning mechanisms

The functional specificities and contextualized nature of ASRMs make their development more sophisticated and challenging than what is typical for AIRMs. Numerous conventional methodological approaches are used in system design engineering, for instance, feature-driven development, V-model, waterfall model, agile software development, and spiral model. However, there is no standard approach for the development of ASRMs for S-CPSs. Evidently, the concomitant challenges vary according to the issues related to the target applications, but there are challenges that are to a large degree independent from concrete applications. These are: (i) the difficulty of foreseeing a real-life situation during design-time, (ii) the need for an adaptive learning mechanism to reason with imperfect information, (iii) the need for multiple algorithms that work in synergy in the implemented ASRMs, (iv) the increased computational complexity as the number and interrelationships of components grow, and (v) the need for verification of the system-level reasoning at design time to guarantee that the ASRM will be properly composed at runtime.

As an example of ARSMs for intelligent mobile robots, discussed in Section 1.2.2.2, the reasoning mechanisms need multiple algorithms of different types for the realization of their functionality. For the visual object recognition, the above reasoning mechanism requires

deep learning type algorithms (e.g., convolutionneural network). For the localization, the inference engine requires ontologically-based reasoning. For the mission planning task, it requires reinforcement learning-type algorithms. In the real-life operation, the different types of algorithms must interoperate in a compositional manner. When the robot is moving, a different As an example of ARSMs for intelligent mobile robots, discussed in Section 1.2.2.2, the reasoning mechanisms need multiple algorithms of different types for the realization of their functionality. For the visual object recognition, the above reasoning mechanism requires deep learning type algorithms (e.g., convolution neural network). For the localization, the inference engine requires ontologically-based reasoning. For the mission planning task, it requires reinforcement learning-type algorithms. In the real-life operation, the different types of algorithms must interoperate in a compositional manner. When the robot is moving, a different situation happens and some required data might be missing. This may lead to a visual recognition failure. To handle this unfavorable situation, functional adaptation at runtime is required. We believe that through the discussed example we demonstrate that ARSMs are indeed complex by nature and pose several previously unexperienced design challenges.

1.2.5. The chosen research problems and the related challenges

1.2.5.1. Designing the smartness

Designing ASRMs for CPSs with smart capabilities is a new issue both for systems research and for system development. Smartness is an inherent quality of human thinking, feeling, doing, and making. In our view, it is not equivalent with, but a subset of the elements of human intelligence [39]. A distinctive characteristic of smart systemsis that the relationships among the component properties create unique patterns of operation on a system level that can only be assigned to the whole and not to any individual components. In this sense, as performed by S-CPSs, smartness is a holistic system characteristic. Smartness implies the need for compositional system engineering, like safety, dependability, andadaptiveness. To achieve system-level smartness, the functional synergy and purposeful (inter) operation of all hardware, software or cyberware components at the low-level of operationalization of S-CPSs are assumed.

Smartness also raises the need for reasoning mechanisms designed for specific applications of S-CPSs. The practical implementation assumes a procedural synthesis of various computational mechanisms such as context-based reasoning, goal-driven strategy planning, functional adaptation, and behavioral evolution. The interplay of software components should be designed so as to produce and take care of smart behaviors. The traditional design methodologies have not been developed to support the designing of smart features of S-CPSs. Thus, the issue of developing novel dedicated design methodologies is combined with the theoretical underpinning and technological realization of software-induced smartness. With regard to knowledge engineering and reasoning mechanisms development, consideration of compositional manners is inevitable. This was one of the background concerns at conceptualization of the proposed ARF.

1.2.5.2. Specification of the research problems

The essence of the problem was that S-CPSs are based on ASRMs that enable them to generate context-dependent solutions for various application problems. Our research concentrated on a newly emerged problem related to the designing of a reasoning mechanism for S-CPSs. More specifically, the embedding research project looks for a theory, a methodology, and a computational realization to support a designing of the smartness capability in S-CPSs. The research problems were addressed at three levels as follows:

Research problem 1: the entire phenomenon of supporting design by the active framework concept is not sufficiently known yet.

Research problem 2: the concept of an active recommender framework (ARF) is new, in particular in the context of S-CPSs, and thus development methodologies for ARF are not available yet.

Research problem 3: there is very limited practical experience with using ARFs in the development of ASRMs for various application contexts.

1.3. Research methodology

1.3.1. Research vision and assumption

Conventional system engineering frameworks (SEFs) play multiple roles in the design processes such as, explaining a phenomenon, addressing a problem and proposing problem-solving methods, offering means of creating a new concept, providing an architectural structure for developing a system, and evaluating and testing system performances. We argue that this type of framework does not provide sufficient support for designing ARSMs for S-CPS, which feature cognitive capabilities such as reasoning, learning, and adapting themselves. Ideation and design of ASRMs are complicated and require diverse cognitive processes, which need both creative thinking and practical pragmatism from the designers. Based on the strong need to support design activities for the development of ARSMs, we proposed the concept of active framework for this purpose.

The research vision was to have a design enabling tool called ‘an active recommender framework (ARF)’ supporting the development of ASRMs. The integration of process monitoring and decision support functions was proposed as the novel functionality. This makes the ARF capable to identify an obstacle in the design process and to offer a recommendation to a designer in order to remove the obstacleand continue the design process. The computational mechanisms of the ARF were able to perform the (semi-) automated operation with the minimum number of the interactions with the designer to generate the personalized recommendation, including process related and content related recommendations.

Due to the challenges and the complicated nature of the research topic, we have made the following assumptions concerning the methodological aspect in order to conduct the research:

Assumption 1: the project should be properly scoped so as to achieve a rationally manageable complexity and challenge, even if it means ignoring many related issues and simplifying processes;

Assumption 2: the research process as a whole can be decomposed into a minimal number of logically connected research cycles;

Assumption 3: the objectives of the research cycles can be defined so as to bridge knowledge aggregation through concept development and implementation, as well as to testing of the proposed theories and implementation principles;

Assumption 4: the ARF can be elaborated on from theoretical, methodological, interaction, epistemology, and praxiological dimensions up to the level that presents it as a new paradigm of active recommender systems;

Assumption 5: the lively interaction between (the development of) the ARF and real-life application problems can be demonstrated in the promotion research by including a concrete application problem;

Assumption 6: the research activities can show the parallels between two activity flows, namely (i) the framework-orientated knowledge aggregation and development of ARF algorithms, and (ii) the application problem solving-orientated knowledge aggregation and development of ASRM algorithms;

Assumption 7: with a view to the foreseen complexity and capacity investment issues, the promotion research is not supposed to be exhaustive, but to achieve a decent level of theoretical clarification and computational demonstration;

Assumption 8: in addition to the propositions closely related to the conduct and content of the promotion research, attention is given to the broader scientific meaning and implications of the studied phenomena, principles, and inventions.

1.3.2. Research objectives

The overall objective of the research was to develop a theory, a methodology, and a feasible demonstrative implementation for an ARF to actively and insightfully support the development of ARSMs of S-CPSs. The specific objectives of the research were (i) to aggregate knowledge concerning the development of reasoning mechanisms for SCPSs and system engineering frameworks (SEFs); (ii) to specify the requirements and the alternative approaches for conceptualization of ARF for RMD; (iii) to conceptualize an ARF that systematizes the development of application specific reasoning mechanism (ASRM); (iv) to propose the novel functionality of the ARF and identify the required algorithms and data structure; (v) to develop a computational implementation of the demonstrative parts of the ARF for a particular design session of an ASRM in an application context; (vi) to validate the functionality of the implemented modules of the ARF in the considered application context; and (vii) to validate the usefulness of the recommendations generated

by the demonstrative modules of the ARF.

1.3.3. Research questions and hypotheses

To achieve the main objective, the following guiding research question was formulated:

> *In what way can an active recommender framework (ARF) support a designer in the development of an application specific reasoning mechanism for a particular family of S-CPSs?*

With regard to the specific objectives, a number of working research questions was synthesized. The questions were classified into four groups. The first group focused on the epistemological aspects of ARF development, concerning knowledge aggregation, specification of requirements, and building a knowledge platform for the follow-up research activities.

RQ1: What framework concepts are currently used to support the implementation of system smartness with regard to S-CPS?

RQ2: What knowledge is needed for a designer concerning the conceptualization and implementation of ARSMs for S-CPS?

RQ3: What requirements have to be fulfilled by an ARF in order to efficiently support a designer in RMD processes?

The second group of research questions dealt with the ideation, conceptualization, and specification of constituents of an ARF:

RQ4: What is the underpinning principle of the conceptualization of an ARF and what structured methodological approach can be used?

RQ5: What functionalities are to be considered at the conceptualization of an ARF?

RQ6: What algorithms are needed for a computational implementation and functional validation of the ARF in the application context of ASRM?

The third group of research questions concerns the implementation phase of the ARF, specifically focusing on the detailing, computational implementation, and testing of the implemented modules:

RQ7: What modules of the ARF should be implemented to arrive at a demonstrative implementation?

RQ8: What critical algorithms and data constructs are needed for the realization of the functionality of an evidencing demonstrative part?

RQ9: What way can the implemented modules be functionality tested in the application context?

Finally, two research questions were posed concerning the validation of the implemented modules and to confirm the quality of recommendations:

RQ10: What way can the performance of the implemented modules be validated

concerning the usefulness of the recommendations?

RQ11: What is an indicator and measure of the usefulness of the recommendations?

Based on these research questions and the forerunning knowledge aggregation, the following hypothesis was formulated:

Research hypothesis: A computer-aided design support tool with new features, including a purposeful coupling of the process monitoring and the decision support functionalities towards a compositional design of application specific reasoning mechnism for S-CPSs, is the needed new paradigm for realization of an active recommender framework.

1.3.4. Overall methodological framing of the research work

At the early stage of the PhD. research project, the Topic and Work Specification (TWS) was documented. This section was revised based on the contents of the TWS. For the methodological framing of the promotional research, a multi-methodological framework was introduced. This includes a combination of research in design context (RiDC), the design inclusive research (DIR), and practice based research (PBR) approaches [40]. Basically, RiDC and PBR involve two phases: (i) explorative research actions, (ii) confirmative research actions. Besides these, the DIR involves a set of constructive design/prototyping actions as a 'transitive relation' between the explorative phase of the research cycle. Thus, this link formed by creative/design activities extends the kernel cycle procedurally with a third stage.

Focusing on data generation and theory development, the explorative phase was divided into three stages: (i) knowledge aggregation, (ii) stating research assumptions, and (iii) theory development. Focusing on theory justification and validation, the confirmative phase consisted of three stages: (i) logical justification, (ii) validation of the conduct and the findings, and (iii) consolidation of the new knowledge. The goal of this systematic approach was to properly explore, describe, understand, and explain the studied phenomenon and its implications.

Design inclusive research uses evolving design as a research means in order to access information and knowledge that cannot be accessed and studied otherwise. As briefly mentioned above, the DIR cycle is extended with a constructive phase. The constructive phase is actually an embedded design and/or construction process, including three stages: (i) conceptualization, (ii) detailing, and (iii) prototyping. The explorative and confirmative research actions are conducted as with the RiDC approach. The research means can be an artifact, a process, a visual/virtual entity, a chunk of knowledge, and a synthetic phenomenon. Some external factors can be considered in the design phase such as technological opportunities, implementation requirements, and usable resources. The goals of the design phase can be achieving a better understanding, inventing new concepts, constructing physical and/or digital models, devising methodologies, and providing a better solution through creating pilot versions of an artefact.

Concerning the specific objectives of the whole research project, the research design was elaborated with a view towards the research constructs, processes, and means/instruments. The overall research process was then divided into four research cycles which included a logical flow of activities. The methodological framing applied to the research cycles resulted in the overall research approach shown in Figure 1.6.

Research cycle 1 focused on the description of the phenomenon regarding the need for a new framework paradigm to support the development of reasoning mechanisms for S-CPS. The first research cycle was framed according to RiDC methodology. The objectives of this cycle were (i) to describe and explain the above generic phenomenon, (ii) to gain sufficient insights into and to explore fundamental principles related to the studied phenomenon, and (iii) to provide a comprehensive knowledge and a rigorous description of it. The completed

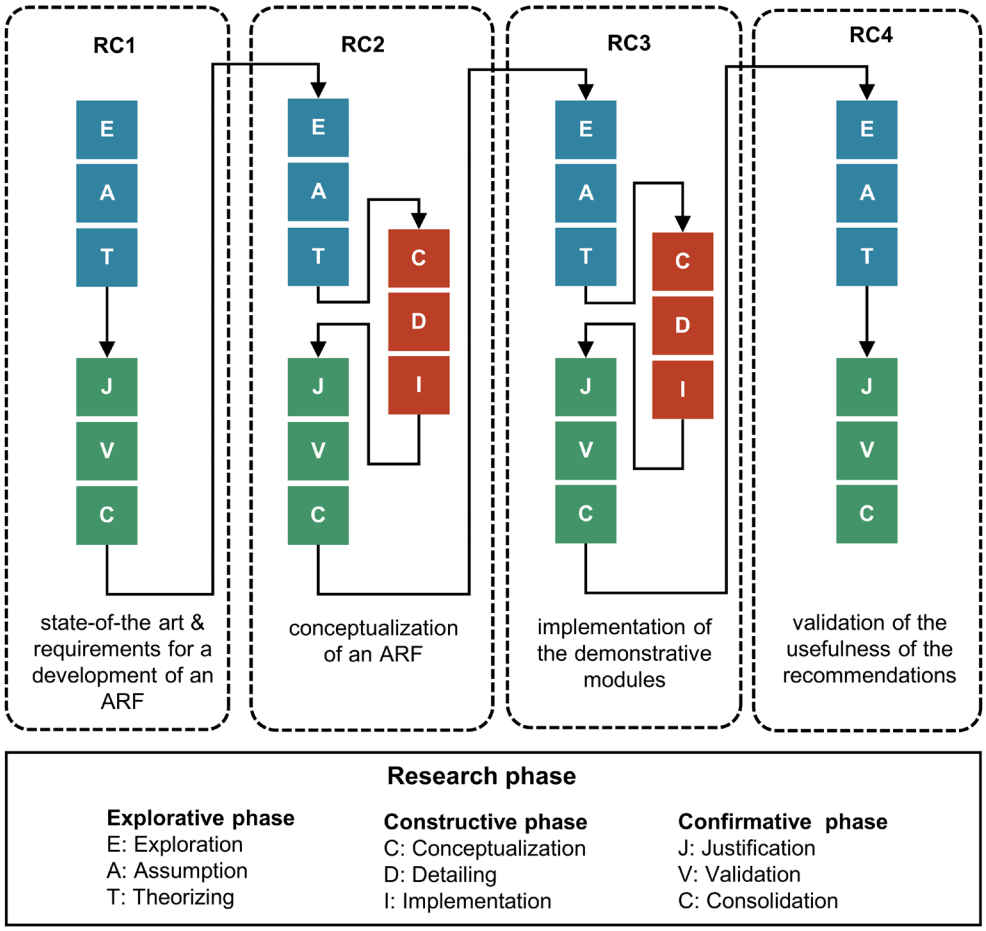


Figure 1.6: The methodological framing of the promotional research

literature study focused on (i) the proliferation and features of S-CPSs, (ii) the essence of system engineering frameworks (SEFs) development, (iii) the enablers of system-level reasoning and services. Both quantitative and qualitative methods were used in a mixed way. A reasoning model was developed for the qualitative analysis of the contents. An objective critique was applied with regard to the SEFs currently used in the development of reasoning mechanisms. Furthermore, the knowledge required for the development of an ARF was circumscribed. Subsequently, a gap between the currently available knowledge and the specific knowledge needed for conceptualization and implementation of an ARF was identified. The implications of the findings as well as their interplay were critically analyzed with the goal of deriving the requirements for the ARF development.

Research cycle 2 focused on conceptualization of the active recommender framework for supporting ASRM development. The proposed concept was validated and tested in the application context of APAS. In this research cycle the design inclusive research (DIR) approach was applied. The design activities concerned the functional conceptualization and system level architecting (decomposition and integration) of the constituents of the ARF. The specification of the functions of the ARF started out from the conceived design actions of the APAS development. Critical system thinking combined with actions of design science research was used to predict what the overall architecture and the functional specification of the framework should look like. Three main research activities were carried out. In the explorative phase, a broadly-based knowledge aggregation was conducted to provide input for the ideation about what services an ARF can offer to a designer. Various assumptions for the development of the ARF were formulated. The design scenario for the development of ASRMs was also analyzed and critically discussed. In the constructive phase, the fundamental concepts of the ARF were specified based on the assumptions and the implications of the concrete findings in the pre vious stage. The ARF was conceptualized based on a multi-perspective approach, which progressed through (i) specification of the functionality of the mechanisms, (ii) system-level architecting and specification of the modules, (iii) construction and specification of algorithms and data constructs, and (iv) organization of the computational workflow. In the confirmative phase, the research activities centered on the feasibility testing of the overall concept and on the demonstration of the operations of the conceptualized part of the ARF in the target application context. In addition, the goal of this phase was to explore and formulate requirements for the implementation of the demonstrative part.

Research cycle 3 was methodologically framed as a DIR approach. The main research activities concentrated on the implementation of the demonstrative moules of the ARF. The activities were completed in three procedural phases. The first phase included the explorative research actions. In the second phase the necessary design actions were performed. In the third phase, the results were processed and conclusions were drawn based on the confirmation research actions. In the exploration phase, a technical specification for the implementation of the demonstrative part of the ARF was elaborated. The research activities focused on the specification of the principles of the implementation of the targeted demonstrative modules and the analysis of the resources (programming environment)

available for the implementation. In the constructive phase, the contents of demonstrative modules were specified on the module, component, and algorithms level. The confirmative phase placed the implemented WPE algorithms into the specific context of the APAS. The set of algorithms develop for the APAS represented a demonstrative implementation of a particular ASRM in the target application. This demonstrative part was intended to test the functionality of the proposed ARF could be implemented with success in the above specific application context.

Research cycle 4 dealt with the validation of the demonstrative implementation with a view towards the usefulness of the recommendations. The research cycle was methodologically framed according to the structure of a practice based research (PBR) approach. The validation aimed at generating indicators for the usefulness of the recommendations. A synthetic agent was designed to act (decide) as a human designer does. Based on this, an agent-oriented validation process was developed and completed. It aimed at mimicking the decisional behavior of the designer and generating an input dataset for validation without including human designers in the validation process. The research activitieswere completed in two phases. The explorative phase dealt with the development of the synthetic validation agent (SVA) and the operationalization of the SVA. As mentioned above, the output of the explorative phase was the validation dataset generated by means of the SVA. In the confirmative phase, the statistical analysis of the dataset and the correlation testing of the considered variables were conducted. Various indicators, derived by using prognostic reasoning, were proposed to measure the usefulness of the recommendations. The final stage incorporated the discussion and the interpretation of the findings with the goal of evaluating the methodology and the usefulness of the recommendations.

1.4. Structure of the thesis

The overall methodological framing presented in the previous sub-chapter was also considered at the structural organization of the thesis. The concrete research activities completed in the four research cycles are discussed in the subsequent chapters. Chapter 2 is dedicated to the second research cycle. It starts with a presentation of the design of the literature study, followed by the conducting of the systematic literature review, analysis of the contents, and then discussion of the findings. A set of requirements for the ARF development was derived based on the implications of findings in the knowledge aggregation procedure. In Chapter 3, the process of conceptualization of the ARF is discussed. This chapter first provides a notional clarification, and then defines the needed functionality, and deals with the issues of efficient architecting and system integration. It also deals with various aspects of ASRM development. In addition, the specification of the modules and algorithms and the planning of and the requirements for implementation are considered.

Chapter 4 is dedicated to the implementation of the demonstrative part of the ARF. The fundamental concepts of the implementation are discussed and the details of the implementation of the modules and the computational algorithms are presented. The implemented modules were tested in the application context of the chosen ASRMs. In

Chapter 5, the validation of the usefulness of the recommendation was conducted. This chapter presents the argumentation about the decisional behavior of the designers, describes its simulation with the synthetic validation agent, the analysis of the validation dataset, and the evaluation of the usefulness of recommendations. In Chapter 6, the completed research project is summarized, including self-reflections on the contributions of the research from various perspectives, as well as the conduct and outcomes of the individual research cycles. This chapter also includes the detailed descriptions of the scientific propositions and makes recommendations for future research.

1.5. List of own publications

1. Tepjit, S., Horváth, I., & Rusák, Z. (2019). The state of framework development for implementing reasoning mechanisms in smart cyber-physical systems: A literature review. *Journal of Computational Design and Engineering*, 6(*4*), 527-541.
2. Tepjit, S., Horváth, I., & Rusák, Z. (2018). An analysis of the state of framework development for reasoning in smart cyber-physical systems. In *Proceedings of the 25th ISPE Inc. International Conference on Transdisciplinary Engineering*, (Vol. 7, pp. 82-92). IOS Press.
3. Horváth, I., Tepjit, S., & Rusák, Z. (2018). Compositional engineering frame-works for development of smart cyber-physical systems: A critical survey of the current state of progression. In *Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 51722, pp. 1-14). ASME
4. Tepjit, S., Horváth, I., & Rusák, Z. (2020). Conceptualization of an active recommendation framework system to support application-driven reasoning mechanism development for smart cyber-physical systems. In *Digital Proceedings of TMCE* (pp. 1-16).
5. Tepjit, S., Horváth, I., & Rusák, Z. (2022). Conceptualization of algorithms for an active recommender framework to support exploration of application-specific working principles. (in preparation).

References

[1] Gunes, V., Peter, S., Givargis, T., & Vahid, F. (2014). A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Transactions on Internet and Information Systems (TIIS), 8*(12), 4242-4268.

[2] Horváth, I., and Gerritsen, B. (2012). Cyber-physical systems: Concepts, technologies and implementation principles. In *Proceedings of TMCE 2012* (Vol. 1(2), pp. 7–11).

[3] Wan, J., Yan, H., Suo, H., & Li, F. (2011). Advances in cyber-physical systems research. *KSII Transactions on Internet and Information Systems (TIIS)*, 5(*11*), 1891-1908.

[4] Rajkumar, R., Lee, I., L. Sha, and Stankovic, J. (2010). Cyber-physical systems: the next

computing revolution. In *Proceedings of the Design Automation Conference* (pp. 731–736).

[5] Poovendran, R. (2010). Cyber-physical systems: Close encounters between two parallel worlds. In *Proceedings of the IEEE* (pp.1363–1366).

[6] Dumitrache, I. (2011). Cyber-physical systems-new challenges for science and technology. *Journal of Control Engineering and Applied Informatics, 13*(3), 3-4.

[7] Horváth, I., Rusák, Z., Hou, Y., and Ji, L. (2014). On some theoretical issues of interaction with socialized and personalized cyber-physical systems. In *Lecture Notes in Informatics (LNI), Proceedings-Series of the Gesellschaft fur Informatik (GI)* (pp.1995-2000).

[8] Engell, S., Paulen, R., Reniers, M. A., Sonntag, C., & Thompson, H. (2015). Core research and innovation areas in cyber-physical systems of systems. In *International Workshop on Design, Modeling, and Evaluation of Cyber Physical Systems* (pp. 40-55). Springer, Cham.

[9] Horváth, I., Rusák, Z., and Li, Y. (2017). Order beyond chaos: Introducing the notion of generation to characterize the continuously evolving implementations of cyber-physical systems. In *Proceedings of the ASME Design Engineering Technical Conference* (pp.1–12).

[10] Pourtalebi, S., Horváth, I., and Opiyo, E. Z. (2014). First steps towards a mereo-operandi theory for a system feature-based architecting of cyber-physical systems. In *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)* (pp. 2001–2006).

[11] Salehie, M and Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems, 4*(2), 1-42.

[12] Zhang, X., and Le Zhou, C. (2013). From biological consciousness to machine consciousness: An approach to make smarter machines. *International Journal Automation and Computing, 10*(6), 498–505.

[13] Long, L. N., and Kelley, T. D. (2009). The requirements and possibilities of creating conscious systems. In *AIAA Infotech at Aerospace Conference and Exhibit and AIAA Unmanned...Unlimited Conference* (pp. 1949).

[14] Mainzer, K. (2015). The emergence of self-conscious systems: From symbolic AI to embodied robotics. In *Philosophy, Computing and Information Science* (pp. 57–66). Taylor and Francis.

[15] Kubota, N., Kojima, F., and Fukuda, T. (2001). Self-consciousness and emotion for a pet robot with structured intelligence. In *Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS* (pp.2786–2791).

[16] Chirikjian, G. S., and Suthakorn, J. (2003). Toward self-replicating robots. In *Experimental Robotics VIII* (pp.392–401).

[17] Lee, K., and Chirikjian, G. S. (2007). Robotic self-replication. *IEEE Robotice and Automation Magazine, 14*(4), 34–43.

[18] Crepaldi, M., Grosso, M., Sassone, A., Gallinaro, S., Rinaudo, S., Poncino, M.,... & Demarchi, D. (2014). A top-down constraint-driven methodology for smart system design. *IEEE Circuits and Systems Magazine, 14*(1), 37-57.

[19] Nawaz, K., Petrov, I., and Buchmann, A. P. (2014). Configurable, energy efficient, application- and channel-aware middleware approaches for cyber-physical systems. In *Studies in Computational Intelligence* (pp. 3–65).

[20] Ali, S., Bin Qaisar, S., Saeed, H., Khan, M. F., Naeem, M., and Anpalagan, A. (2015). Network challenges for cyber physical systems with tiny wireless devices: A case study on reliable pipeline condition monitoring. *Sensors (Switzerland), 15*(4), 7172–7205.

[21] Xia, F., Kong, X., and Xu, Z. (2011). Cyber-physical control over wireless sensor and actuator networks with packet loss. In *Wireless Networking Based Control* (pp. 85–102).

[22] Lee, E. (2008). Cyber physical systems: Design challenges. In *Proceedings of the 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing* (pp. 363–369).

[23] Dillon, T. S., Zhuge, H., Wu, C., Singh, J., and Chang, E. (2011). Web-of-things framework for cyber-physical systems. *Concurrency and Computation: Practice and Experience*, *23*(9), 905–923.

[24] Wang, J., Zhao, Q., and Zhao, Y. (2013). An effective framework to simulate the cyber-physical systems with application to the building and energy saving. In *Chinese Control Conference (CCC)* (pp. 8673–8641).

[25] Niggemann, O., Biswas, G., Kinnebrew, J.S., Khorasgani, H., Volgmann, S.,and Bunte, A. (2015). Data-driven monitoring of cyber-physical systems leveraging on big data and the internet-of-things for diagnosis and contro. In *CEUR Workshop Proceedings* (pp. 185–192).

[26] Sharma, A. B., Ivančić, F., NiculescuMizil, A., Chen, H., and Jiang, G. (2014). "Modeling and analytics for cyber-Physical systems in the age of big data. *Performance Evaluation Review*, *41*(4), 74–77.

[27] Dasgupta, R and Dey, S. (2013) A comprehensive sensor taxonomy and semantic knowledge representation: Energy meter use case. In *Proceedings of the International Conference on Sensing Technology* (pp.791–799).

[28] Smirnov, A., Levashova,T., Shilov,N., and Sandkuhl, K. (2014). Ontology for cyber-physical-social systems self-organisation In *Conference of Open Innovation Association (FRUCT)* (pp.101–107).

[29] Seshia, S. A., Hu, S. Y., Li, W.C., and Zhu, Q. (2017). Design automation of cyber-physical systems: Challenges, advances, and opportunities. *IEEE Transactions on Computer aided Design of Integrated Circuits Systems*, *36*(9), 1421–1434.

[30] Håkansson, A., Hartung, R., and Moradian, E. (2015). Reasoning strategies in smart cyber-physical systems. *Procedia Computing Science*, *60*, 1575–1584.

[31] Horváth, I. (2017). Procedural abduction as enabler of smart operation of cyber-physical systems: Theoretical foundations. In 2017 *International Conference on Engineering, Technology and Innovation (ICE/ITMC)* (pp. 124-132). IEEE.

[32] Hao, J. G., Bouzouane, A., Bouchard, B., and Gaboury, S. (2018). Activity inference engine for real-time cognitive assistance in smart environments.*Journal of Ambient Intelligence and. Humanized Computing*, 9(*3*), 679–698.

[33] Chen, R., Hua, Q., Ji, X., Liu, Y., Wang, H., Li, J., ... & Feng, J. (2017). An interactive

task analysis framework and interactive system research for computer aided diagnosis. *IEEE Access*, 5, 23413–23424.

[34] Mart, E. (2015). Adaptive sensor fusion architecture through ontology modeling and automatic reasoning. In *2015 18th International Conference on Information Fusion (Fusion)* (pp. 1144-1151). IEEE

[35] Joo, S. H., Manzoor, S., Rocha, Y. G., Bae, S. H., Lee, K. H., Kuc, T. Y., & Kim, M. (2020). Autonomous navigation framework for intelligent robots based on a semantic environment modeling. *Applied Sciences*, *10*(9), 3219.

[36] Tanik, U. J., & Begley, A. (2014). An adaptive cyber-physical system framework for cyber-physical systems design automation. In *Applied cyber-physical systems* (pp. 125-140). Springer, New York.

[37] Horváth, I., & Gerritsen, B. H. M. (2013). Outlining nine major design challenges of open, decentralized, adaptive cyber-physical systems. In *Proceeding of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (pp. 1-12). ASME.

[38] H. Muccini, M. Sharaf, and Weyns, D. (2016). Self-adaptation for cyber-physical systems: A systematic literature review. In *Proceeding of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 75-81).

[39] Horváth, I., Zeng, Y., Liu, Y., & Summers, J. (2021). Smart designing of smart systems. *AI EDAM*, *35*(2), 129-131.

[40] Horváth, I. (2008). Differences between 'research in design context' and 'design inclusive research' in the domain of industrial design engineering. *Journal of Design Research*, *7*(1), 61-83.

Chapter 2

Research cycle 1:
Aggregation of knowledge and exploration of requirements

2.1 Objectives and methodological framing of the first research cycle

2.1.1 Objectives

As mentioned in the first chapter, the need for generating and processing coherent and consistent problem solving and process steering system knowledge flow makes SCPSs compositional in nature. It means that a compositional S-CPS manifests as a purpose-driven arrangement of computational reasoning mechanisms, which in turn are purpose-driven arrangements of computational reasoning algorithms. A system that can automatically find a parking lot and park a car is a typical example for such computational mechanisms and algorithms. Such applications need situation dependentand/or context-driven reasoning in runtime operation, since it is not feasible to pre-program all possible parking solutions. Eventually, computational reasoning mechanisms are data-driven. Due to their complexity and compositionality, application specific, runtime active reasoning mechanisms pose many novel design challenges, particularly in terms of creating system-level smartness and constructing applicationtailored knowledge flows. Traditionally, system-engineering frameworks (SEFs) have been used to support the activities of designers.

The objectives of the first research cycle were (i) to get a deeper insight into the studied research phenomenon, (ii) to get an overview of the state of the art based on the related scientific literature and professional web repositories, and (iii) to synthesize a starting ‘home base’ for the investigations and a knowledge platform that can be used follow-up developments. Knowledge aggregation included the study of the SEFs proposed for supporting

the development of a reasoning mechanism for S-CPSs. The content development was based on the assumption that the overall objective of the promotion research was the development of a new generation of active framework to support (at least a part of) the design process of application-specific runtime reasoning mechanisms.Specific attention was given to SEFs that are used for the development of system-level design reasoning in the context of S-CPSs.

The objectives of the completed survey were:

(i) to study the SEFs available for the development of reasoning mechanisms from the perspective of S-CPSs (more specifically, the frameworks that are proposed for handling the fuzzy front end of the reasoning mechanism's development process);

(ii) to investigate the enablers of system-level reasoning as necessary ingredients of implementing smartness in S-CPSs;

(iii) to identify the requirements for the development of an active recommender framework supporting the design process of ASRMs.

In addition, the knowledge aggregation activities included the identification of the existing knowledge gaps and collection of requirements as part of the knowledge platform for an active recommender framework (ARF) development. The guiding research question was how an ARF had to facilitate the development of ASRMs for SCPSs. Towards an underpinning theory, answering the following three questions was the focus of the knowledge aggregation activities:

(i) What was the state of the art of SEFs in the field of system design and engineering?

(ii) What knowledge was needed by a designer for the implementation of a reasoning mechanism for S-CPSs?

(iii) What requirements had to be fulfilled for an ARF to support a designer in the development of ASRMs?

2.1.2 Methodological framing of the first research cycle

In line with the above-described objectives (i.e., knowledge aggregation and requirements exploration), the research cycle was methodologically framed as research in design context (RiDC). The research context was the targeted support of designers in RMD processes by a dedicated ARF. The research cycle was executed in twoconsecutive phases as shown in Figure 2.1. In the explorative phase, after identification of relevant sources of publications, a systematic literature study was conducted at the earlier stage of the research project. The obtained information was used to map the questions-relevant literature publications to a landscape of concerns related to SEFs. In combination with critical system thinking, both quantitative and qualitative methods were applied. As far as the quantitative method was concerned, a bibliometric map of the publicationswas constructed with the goal to visualize the relationships of the specified key terms included in the corpus (the collection of publications). The reasoning model was derived based on

the findings of the quantitative analysis. It would be used for the qualitative analysis.

Since the time of this first conducted literature study, many related research efforts and results have been published in the literature. Therefore, in the process of preparing the dissertation, the original bibliometric map has been updated in line with a new additional collection of publications. Besides updating the first literature study, the second analysis was also intended to confirm that a proper choice was made concerning the phenomenon and methodological orientation of the promotion research.The robustness of the choice was indicated by the facts that (i) other researchers also initiated research on the same or related topics, and (ii) the topic of the literature study was far from exhausted. Nevertheless, though the original reasoning model could be reused, the need for an extension arose (see Figure 2.6).

The content analysis was used as the qualitative method. It was narrowed down to three domains according to the reasoning model: (i) the domains that provided the context information for the research, namely: cyber-physical systems and system smartness, (ii) the domain of discourse of the research including the details of framework development from multi-perspectives, and (iii) the domains that provided content information for studying frameworks. It complicated our study that there were many epistemological and methodological relationships among the domains and their elements.

The confirmative phase was orientated to the logical consolidation of the findings. First, the findings concerning the research and development opportunities of the SEFs were synthesized to support the design process of ASRMs. Currently, representative examples of SEFs are used in various domains of interest. The most important characteristics of an ARF have been identified as

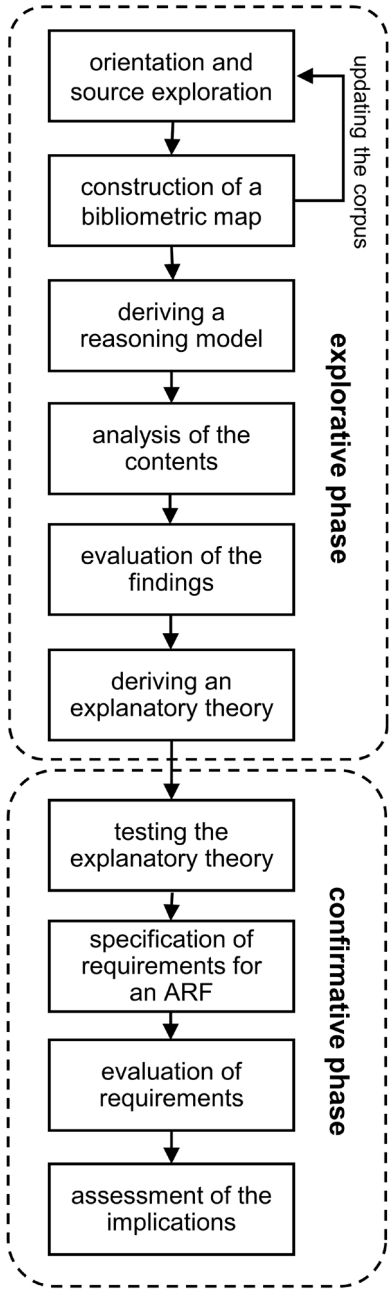


Figure 2.1: The procedural approach of RC1

the provided functionality, the architectural components, the implementation principles, and appropriateness of recommendations. To benefit from these, the ideation process of the ARF was completed with a view to the specific operational characteristics including, for example, the possible services, the primary support functionality, the software architecture, and computational mechanisms.

Having the potential characteristics of an ARF identified, various sets of requirements were identified based on the findings of the literature study, employing brainstorming technique, and critical systems thinking. The fact of the matter is that most of the requirements were drawn by analyzing the implications of the findings of the literature study. They were collected from multiple sources and analyzed to arrive at a technically meaningful and logically consistent list of requirements for an ARF. The requirements were clustered into groups (e.g., functional requirements, structural requirements, computational requirements, and application-oriented potential requirements) for the purpose of further analysis. The requirements were justified by critical system thinking and their consistency within and across groups were checked. In addition, with the aim of validation of their feasibility, the whole set of requirements was evaluated for achievability and practicability.

2.2 Design of the literature study

2.2.1 Procedural phases of the literature study

Including the follow-up complementing study, the literature studies were completed in three subsequent phases, namely: (i) orientation and source exploration, (ii) content analysis, and (iii) synthesis and consolidation of the findings. In the first and the third phases, critical system thinking was the main method of analysis. In the second phase, both quantitative analysis and qualitative interpretation were used. In the web-hosted study, (i) various search engines, (ii) a reference management tool, and (iii) a (bibliometric map) visualization tool was used as a research instrument. The primary criteria for picking up a document for further examination were (i) relevance of content, (ii) time of publication, and (iii) the quality (recognition and reliability) of the source. We intended to achieve both comprehensiveness and coherence throughout all phases of the study. We used the reference management software not only to sort the publications into clusters, but also to support the storage of the corpus.

As the primary source of data, the core collection of the Web of Science was used. To extend the base of the literature study, other sources, for example, databases in specific CPSs-related disciplines and web repositories were also used. The term framework was considered as a main keyword. The exploration of frameworks for reasoning was to be done in the context of S-CPSs, which was considered as a family of CPSs. Other relevant keywords were defined including self-awareness, self-adaptation, smartness, smart cyber-physical systems, system knowledge, context and situation awareness, reasoning mechanisms, and system adaptation. The preliminary inquiry hypothesis addressed the relationships of the keywords to each other. These relationships were investigated in the

completed content analysis.

According to the original literature study published in 2018, the publications included in the study were covered over the past ten years (2008-2017) at the time of compilation of the papers. There were 807 publications that fulfilled the criteria of the conducted search and served as the knowledge base for the literature study. While preparing the dissertation, we developed the complementing study by updating the publications from 2018 to mid-2021. The objectives of the study update were: (i) what new results were published in the domains identified by the original reasoning model, (ii) what important novelties emerged that were not included in the original reasoning model, but are important, (iii) what similar works have been initiated since 2018 in the same field of interest, and in different fields of interest (e.g., law, commerce, and education), to create active recommender frameworks (i.e., for supporting the design of application-specific reasoning mechanisms), Since the time of performing the original literature research, the sample has grown with a number of 1,249 publications. Thus, the extended corpus of the complete literature study was a total of 2,096 publications. The statistical trend of yearly publications is shown in Figure 2.2. The increased number of publications over the period of the completed study gave us the impression that the topic chosen for the promotion research was interesting for the concerned research communities (e.g., in the field ofCPS development, cognitive system engineering, and multi-disciplinary information engineering).

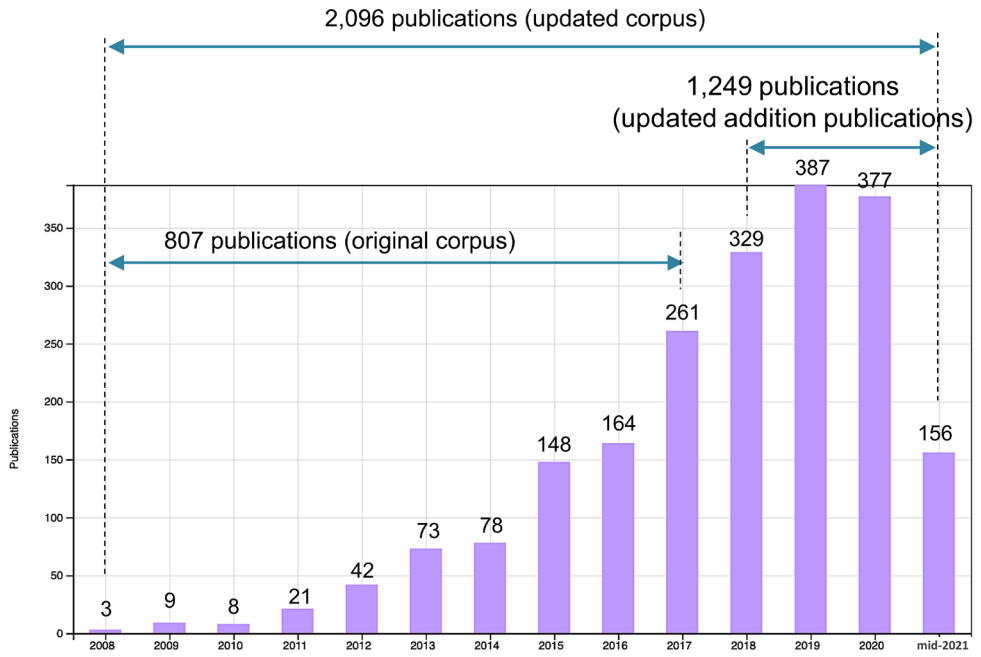


Figure 2.2: Number of publications included in the literature study over the period from 2008 until mid-2021.

In the third phase, a bibliometric map was constructed. This map relied both on the initial search words/phrases and on the found key terms (Figures 2.3 and 2.4). The objective of constructing the map (notional landscape) was to capture and visualize the network formed by the key terms extracted from the corpus. The VOSviewer software was used for this bibliometric mapping. The exploration using the VOSviewer was conducted in three steps: (i) extraction of key terms from the collected publications by using text-mining technique, (ii) manual filtering (removing or merging) key terms that resembled each other, and (iii) creating the map as a network of co-occurring key terms (with nodes and edges). The software assigned the nodes of the map to a cluster for which it was assumed to be relevant. Technically, the clustering used by VOSviewer was based on optimization using the smart local moving algorithm [1].

In the original literature study, the bibliometric map contained 67 nodes and over 2,000 links. These were sorted into three main clusters as shown in Figure 2.3. These clusters were regarded as knowledge domains and (intuitively and manually) named as: (i) systems engineering, (ii) knowledge processing, and (iii) system behaviors. It was interesting to observe that an intersection of the above three clusters was formed and populated with key terms such a framework, models, structure, and application. The key term '*framework*' appeared as an intermediary link across different terms in the different domains.

The original bibliometric map was modified according to the additional new publications. As a result, the modified map contained a total of 26 nodes and over 2,500 links (see Figure 2.4). The key terms were also sorted in the tree clusters, which could be identified with the same domains of knowledge. The term *'framework'* had been the intermediary link across the domains. Some new terms appeared, for instance, related to (i) cognition in system behaviors, (ii) artificial intelligence (e.g., machine learning, deep learning), (iii) recommendation in knowledge processing, and (iv) application contexts in the field of system engineering of CPSs (e.g., industry 4.0, smart city). In spite of the emergence of some new terms, the main key terms and their relationships remained relevant in the map. This confirmed the correctness of the starting hypothesis, which expressed the need for proper SEFs in the development of S-CPSs.

2.2.2 Devising the reasoning model for the original literature study

In the quantitative analysis, the number of occurrences of the key terms 'singular' occurrence and 'coupled' occurrence were taken into account. In the case of a coupled occurrence, two or more key terms simultaneously appeared in a particular publication. The key terms were visualized as node of the map, and the links among them as edges. This *'landscape map'* provided information about the distribution as well about the interrelatedness of the key terms. Based on the number of the incoming and outgoing edges, the centrality of the individual key terms could be seen. The ones with large number of interconnections formed a kind of hub. Thus, our quantitative analysis could use both frequency analysis and topical distance evaluation.

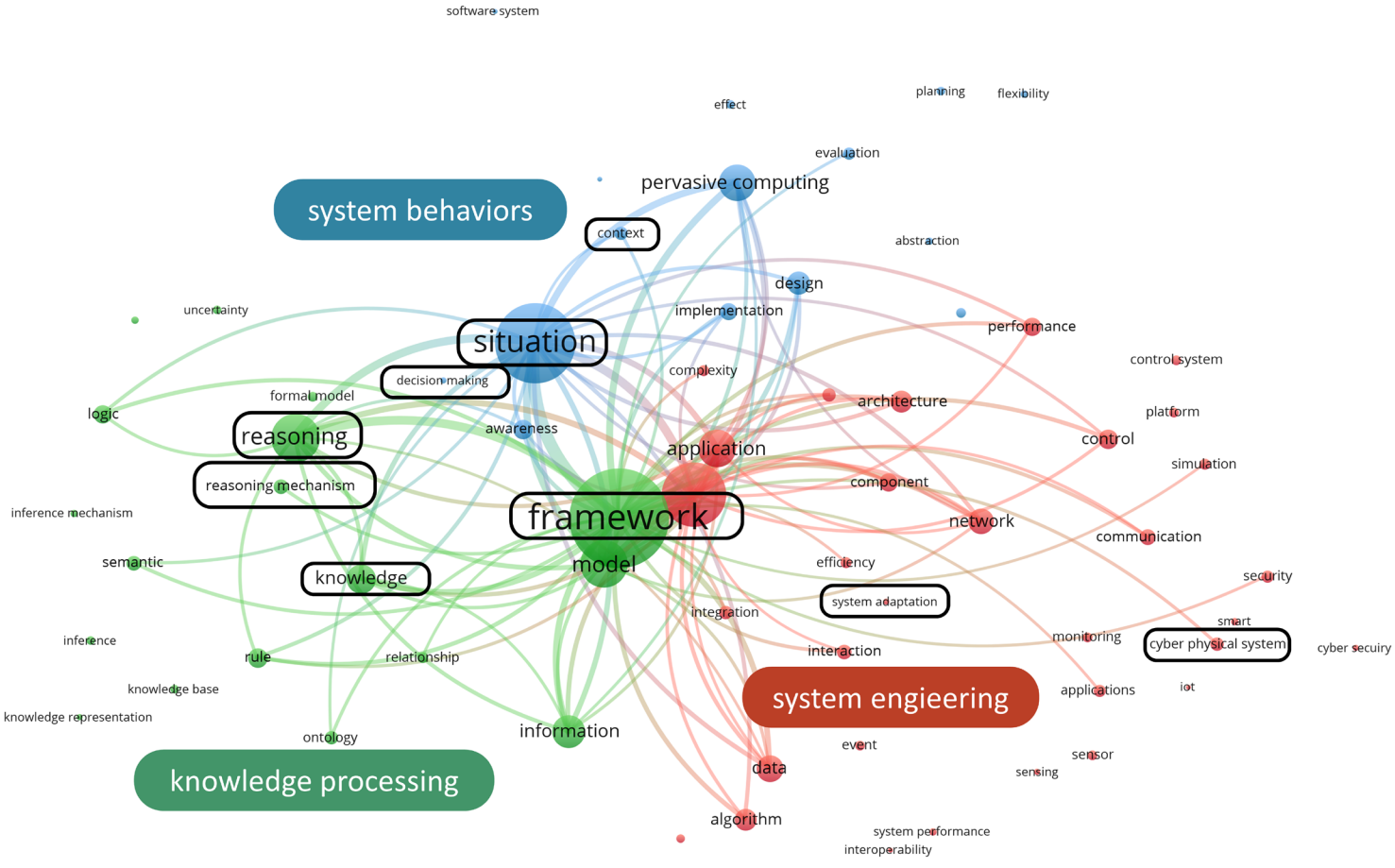


Figure 2.3: Original bibliometric map of the specified search phrases and the found key terms (Based on the own publicatio [no.3])

The topical distance of the key terms was enumerated based on the total number of intermediate nodes on the shortest path (sequence of edges) between considered two key terms. The number of edges between the key terms (i.e., the number of mutual occurrences) was used as a strength indicator of their interrelationships. Concerning the complete set of the search words/phrases we used, it was observed that some key terms were only weakly related to others. Having omitted these weaker relationships, the map size was reduced and was named *power map*. The outcome of this quantitative characterization was used in the qualitative interpretation of the bibliometric map.

The road to the reasoning model led through the following preparatory steps: (i) defining the concept structure, (ii) formulation of a tree of search words/phrases, (iii) generation of the bibliometric map, and (iv) contents analysis of the network formed by the key terms.

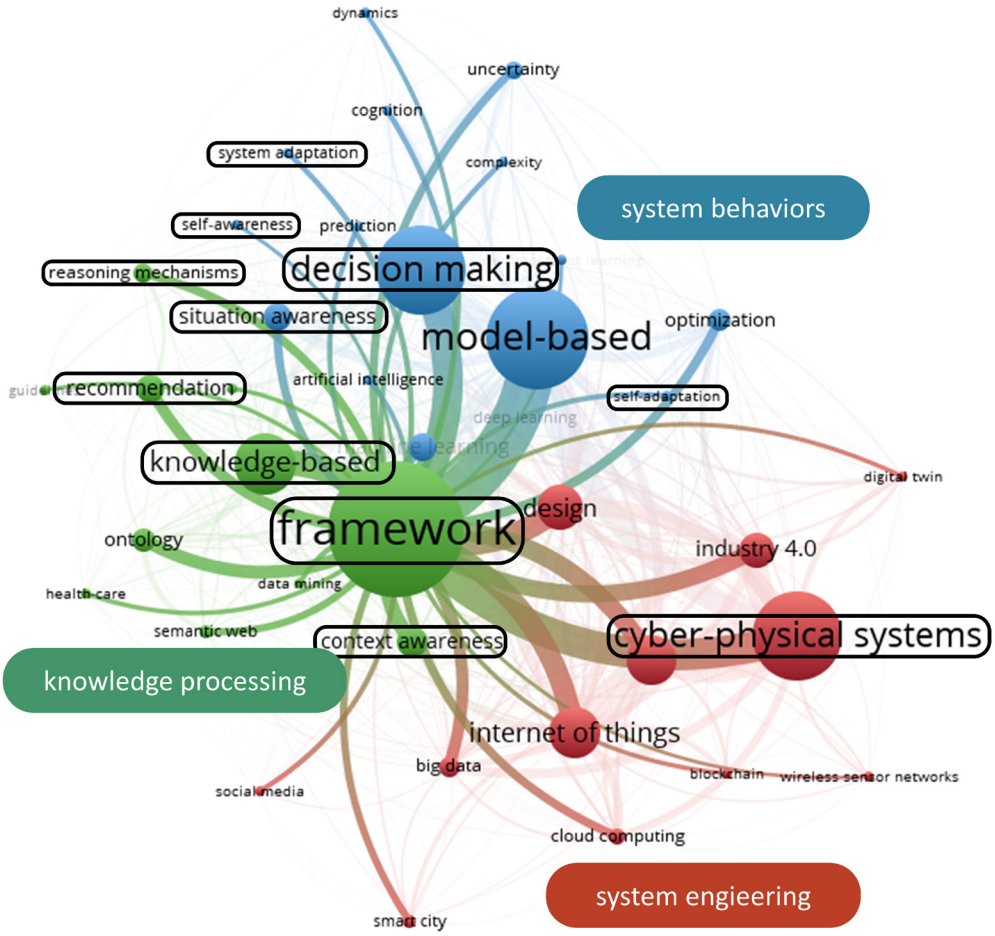


Figure 2.4: The updated bibliometric map (Modified from the own publication [no.3])

Contents analysis is a widely used qualitative research method for literature studies. It is a systematic review that draws conclusions based on a single, or multiple bodies of text to explore trends, themes, and other common (or even distinct) characteristics of data. Usually, contents analysis applies specific coding and assigns a structure of meta-data to the textual data.

We also completed these activities of contents analysis in our study. The meta-data were such as: (i) descriptors of the clusters, (ii) chronological tendency of citation, and (iii) the type of contents (as knowledge). Based on the latter, the relationships found between the key terms were characterized qualitatively. As shown in Figure 2.5, the key term *'framework'* has strong couplings with other key terms, such as decision-making, knowledge-based, and cyber-physical systems. This indicated the roles of frameworks in these areas (e.g., decision-making framework, knowledge-based frameworks).

Notwithstanding, continuous paths could be observed among many weakly coupled key terms (e.g., self-awareness, system adaptation). Based on the power map, the original reasoning model was derived to guide the literature study. Our considerations and decisions concerning the interest domains included in a possible reasoning model were as follows: '*Cyber-physical systems*' was taken into consideration as a broad domain of context

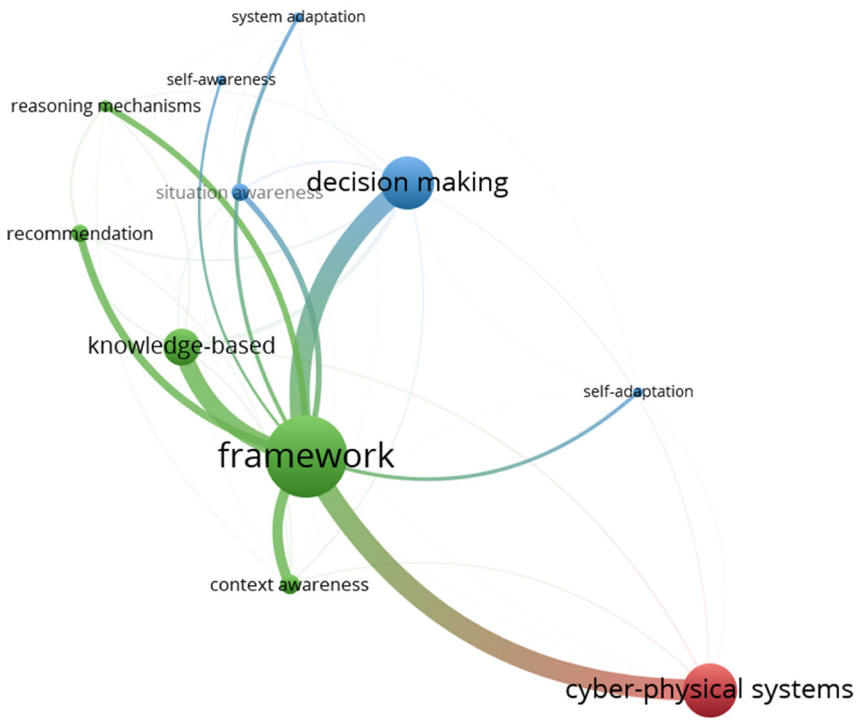


Figure 2.5: Power map of the network of keywords embedded in the updated bibliometric map (Modified from the own publication [no.3])

and '*system smartness*' as a narrower domain of context for our investigation. '*System engineering frameworks*' were taken into consideration as the primary domain of discourse.

As pertinent subject matters (domains of contents) for the investigation, (i) system-level reasoning, (ii) synthetic knowledge, (iii) system reasoning, (iv) system awareness, (v) decision making, and (vi) system adaptation were considered. According to the modified bibliometric map, the term 'recommendation generation' was introduced to the domain of contents. This is shown in Figure 2.6. The relationships of the subject matters were made explicit based on this formal reasoning model, and were used in the qualitative interpretation of all collected relevant publications. The next sub-chapters and sections were arranged according to the brushed up reasoning model shown in Figure 2.6. It must be also mentioned that some sections included below have been published in our publications no. 1 and no. 3. The revised versions of these contents were included in Sections 2.3.1-2, Sections 2.4.1-3, and Sections 2.5.1-5. The textual modifications were included with the intent of improvement.2.3 Overview of the state of the art in the context domains

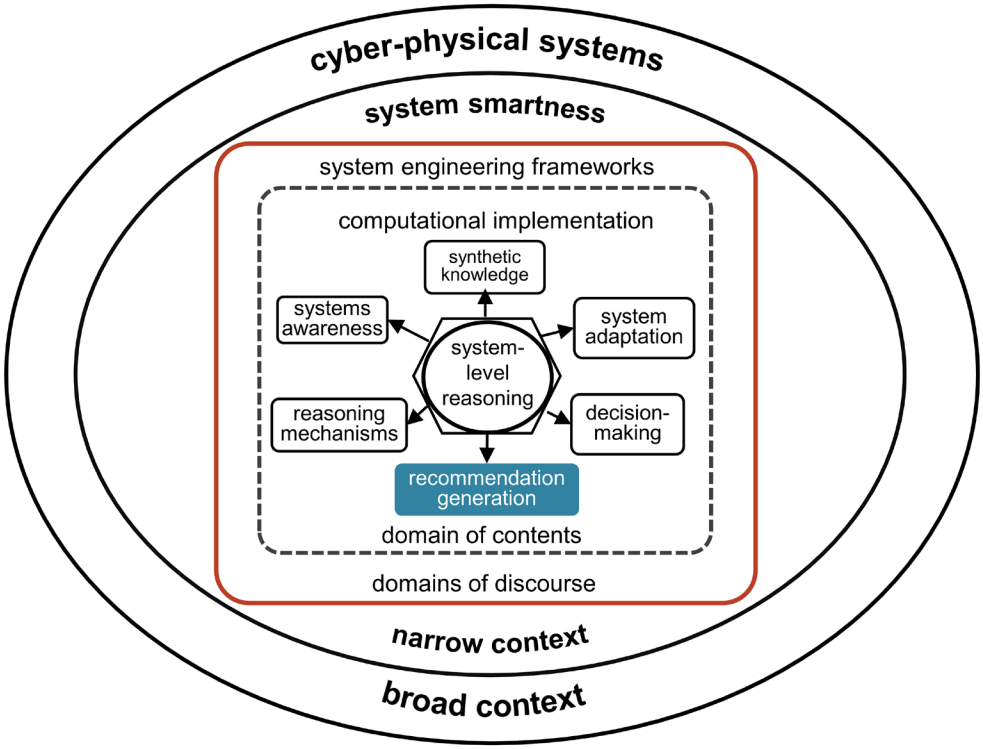


Figure 2.6: Derived reasoning model for the literature study (taken over and modified from the own publication [no.1])

2.3.1 Progress in the development of cyber-physical systems

Cyber-Physical Systems was coined around 2006 by the group of researchers previously working in related domains of interest such as embedded systems, advanced robotics, real-time systems, hybrid systems, and control systems [2]. In 2007, there were twelve papers found when 'Cyber-Physical Systems' was searched on Web of Science. Recently (2021), it has increased exponentially to over 12,800 papers in total. This indicates that CPSs are the informed engineered systems, which have been developed intensively since the last decade. On the scientific domain, CPSs seem to be a kind of model for next generation engineered systems. They have emerged from the integration of two dominant areas that are (i) system with embedded software, and (ii) global data networks [3]. However, there are various perspectives on CPSs that could not be defined by a unified term. This is one reason why their ontological, epistemological, and methodological foundations are still missing.

Another one is the observable rapid shift in the paradigm. While 'a tight integration of computing and physical parts' was an agreement on the essence of CPSs [4], 'deep penetration into physical, biological, social, human, cognitive, etc. processes in a self-organized manner' is now getting accepted by the majority of experts [5]. This makes it possible to introduce CPSs in various human, social, etc. task domains, and not only in purely industrial application fields. Many of these applications could be not addressed successfully by other paradigms of engineered systems. Most of the CPSs are highly heterogeneous conglomerates of interconnected analogue and digital hardware, control software platforms and application tools, and structured knowledge repositories and data streams as cyber-ware [6].

CPSs are capable to provide a very wide range of functionalities, since they usually incorporate a large number of networked actor nodes that can perform smart anticipating behavior. From an architectural perspective, these networked actor nodes form either a closed or an open system, which in turn can be a constituent of a more complex system of systems [7]. The IoT-based interconnection of distributed devices via the Internet offers quasi-real time connections to both users and the physical world. The proliferating cloud platforms offer a wide variety of capabilities and provide virtually unlimited on-demand resources for current implementations. Modern CPSs are designed to be functionally dependable, secure and safe, adaptive and selfsupporting [8]. Self-adaptive software architecture plays a paramount role in the implementation of adaptive CPSs [9].

Many researchers have put the issues of evolutional and replicative CPSs in their research portfolios as well [10]. The typical methodological strategy of designing and managing 1G-CPSs is using a model-based approach. In the overwhelming majority of cases, model-based design is complemented with component-based realization. Both of these have reached the status of a de facto standard. System models are multifold and varied in terms of their contents, representation, and interoperation. However, they are based on knowledge structures, and workflows implied by the traditional systemengineering frameworks (TSEFs). They usually capture descriptive and prescriptive data/information,

which may be extended with predictive information in the form of situational and inferential rules, but do not support fully data-driven development and/or operation [11].

An intense diversification can be observed not only in terms of the application fields, but also in the system functionalities [12]. There are systems whose output is dominantly information service, while other systems provide transformative services for real-life processes and stakeholders. Due to the gradual intellectualization of operations of CPSs, the concept of cognitively enabled CPSs has emerged. This gives a ground to new research and design challenges such as:

(i) attaining situation awareness in the case of large distributed systems with decentralized management and control [13];

(ii) handling large amounts of data in real time with the objectives of (i) providing adapted services, (ii) monitoring system performance and environmental dynamics, and (iii) detecting faults and degradation [14];

(iii) learning useful patterns of auto-reconfiguration and self-adaptation from past examples [15]; and

(iv) analysis and smart reasoning concerning user behavior, exploration of non-predefined needs, and detection of intents and activities [16].

As examples of cognitively enabled CPSs, 2G-CPSs: (i) deeply penetrate into physical, computational, social, cognitive, emotional, etc. real-life processes, (ii) collect data and derive information runtime, (iii) generate alternative operation strategies based on the acquired data, and (iv) operationalize the best matching strategy through functional and architectural adaptation. Normally, they manifest as system of systems [17]. The capability of building awareness and making adaptations is crucial for 2G-CPSs [18]. This enables them to work properly in emergent situations and/or in dynamically changing environments. However, the former is still limited by the premature computational reproduction of self-consciousness, the latter by the physical constraints of resource provisioning [19]. Preprogrammed system/control models either pose significantrestrictions on runtime alteration and adaptive behavior or simply do not support them [20].

System level reasoning and synthetic knowledge are two major enablers of this kind of operation [21]. None of these can be associated with one single component of a system. In the light of the above facts, our major methodological findings are:

(i) the methods and tools available for (system-level) synthesis and modeling of CPSs rest on the principle of reductionism and show an incompatible diversity,

(ii) though compositionality is recognized as a paradigmatic feature of smart systems, its manifestation in artefactual and servicing contexts is not sufficiently understood [22], and

(iii) no specific system engineering framework has been proposed for compositional realization of smart CPSs.

2.3.2 Achievement in the implementation of system smartness

The term '*smartness*' is ambiguously defined and interpreted in the current literature, and is used in diverse perspectives [23]. In certain publications 'smart' is used as a synonym of 'modern', 'sophisticated' and 'up-to-date', whereas in other publications it is equivalent of 'intelligent', 'adaptable', and 'cognitive'[24]. For example, it is stated that SCPSs exhibit a high level of 'intelligence' in terms of opportunistic cooperation, dynamic self-organization, self-healing, and self-adaptation as a characteristic feature[25]. Other authors claimed that S-CPSs could adaptively collaborate with other systems at runtime [26]. Consequently, there are still ambiguity and uncertainty related to '*smart cyber-physical systems*', although the concept has appeared in scientific publications since 2014 (See for example [21], [27]). System theory interprets smartnessas a holistic behavioral characteristic [28]. This holistic view is rooted in the following:

(i) the overall behavior of the whole system cannot be explained by decomposing it into isolated parts, and

(ii) the relationships among the components of a system may give rise to distinctive behavioral patterns that may largely differ from the intrinsic properties of the individual parts.

Smartness of systems may come from interpreting and reasoning with incoming sensor data and combining the outcome with the knowledge owned by the system concerning the respective processes and with the knowledge of how the system works internally [29]. This leads to situated reasoning and decision making in contexts.

S-CPSs have been identified as the next rational step in realization of industrial systems [30]. In order to show 'system intellect' in their operation, their control regime must be more sophisticated and capable to implement many self-*characteristics. In the process of transitioning from 1G-CPSs to 2G-CPSs, the system capabilities of self-regulation and self-tuning are supposed to be replaced by self-awareness and self-adaptation. These go beyond (i) self-adjustment, (ii) self-healing, (iii) self-optimization, and (iv) self-protecting capabilities, and the other forms of selfmanagement and self-organization normally expected from 1G-CPSs [31].

Smartness enables systems to build their own model concerning what the most probable operational situation is, and how to respond to this situation with a preferable objective and in a favorable manner. Should the situation (task, state, environment, etc.) dynamically change over time, a smart system must adapt itself to deal with the dynamics while maintaining the requested level of performance or even improving it when repeatedly confronted with the similar situations. Wang (2009) presented a cognitive reference model of architectures and behaviors of cognitive robots, which reveals the architectural differences and behavioral characteristics of cognitive robots beyond conventional imperative robots [32].

Higher-level system intelligence and operational autonomy need even more advanced system understanding, knowledge intensiveness, and complex anticipation [33]. Evidently,

these expectations go together with many new challenges for conceptualization and design of next-generation CPSs. One of the design challenges is 'partial design' This means that systems are not (cannot be) defined exhaustively in the design stage, because they adapt and/or develop themselves during runtime according to the internal and external operational conditions, the predefined and the possible objectives, and the available or acquirable resources. This ultimately means that a part of the design tasks is delegated to CPSs that learn, reason, and evolve (under a 'remote' strategic supervision of humans). Future system intelligence is conceived as a multi-functional abstract intelligence (αI) that blends the core of neural, cognitive, functional, social, and logical inferences into a common and unified framework [34].

Our conclusion has been that system smartness, as with other holistic system level features, assumes a compositional design that in turn needs to be based on a compositionality-enabling framework [35]. In compositional systems, the function and the architecture of a non-primitive component depend on the wholeness that is formed by the total of the components [36]. The necessary function and architecture of each component can be determined by means of applying some sort of recursive behavioral composition rules that work in parallel with physical interoperation composition rules. This is actually the so-called '*principle of compositionality of system behavior*'.The realization of this typically raises the need for adaptation of the components by the designers in the design stage, or for self-adaptation of the system in the operation stage [37].

2.4 Investigation of system-engineering frameworks for S-CPSs

2.4.1 Progress in the area of system-engineering frameworks

System-engineering frameworks (SEFs) are starting points at the development of systems. They propose a structure of thinking that establishes a body of concepts and provides representations for conceptualizations. Frameworks are also regarded as blueprints that can be converted into various artefact and process models instrumental for building concrete solutions [38]. Typically, frameworks are discussed from ontological, epistemological, and methodological viewpoints. In line with the objective explained in Section 2.1, we contemplated frameworks only from an ontological viewpoint (what exists in the form of frameworks) and from an epistemological viewpoint (what bodies of knowledge a framework captures). However, we intentionally ignore discussing them from a methodological viewpoint (e.g., how a framework can be operationalized in practice).

Based on the survey, the notion of framework is defined as an essential supporting structure underlying a system, a concept, or a text. But the survey also emphasized that there was no consensus either on the definition of the term, on the epistemological consistency, or on any rules in the usage of it [39]. The fundamentals of SEFs found in the literature could be classified into four concepts, namely:

(i) A framework as a structure of logical thinking [39]

This is a general concept of a traditional framework. It is a structure of something serving a particular purpose. For example, in [40], the framework shows the logic flow of the recommendation process.

(ii) A framework as a meta-model of a system

A framework represents an abstract level conceptualization of a system that can be developed into multiple models. Its aims at providing a user with modelling elements, a concept structure, and a set of constraints for creating a system model [41]. A concept structure is used as a frame describing how modelling elements are connected in a particular application model, as seen in a model-based framework of design and verification [42], and four-layer decision cycle framework [43].

(iii) A framework as a model integration

In contrast with a meta-model, a framework allows multiple models integrated at a higher level of abstraction [45]. In the OpenMETA integration framework [44], multiple models are converted into a meta-level model. Based on this, a range of concepts, models, techniques, and methodologies can be clarified and/or integrated as shown in Figure 2.7.

(iv) A framework as a concept creation tool

A framework provides a set of concepts and possible connections. The relationship of concepts either within or across domains of concepts is able to create a new concept for system development [46]. Within the dual cognition design framework [47], it exemplifies the ideation on product development that is created by the interrelationships of domains

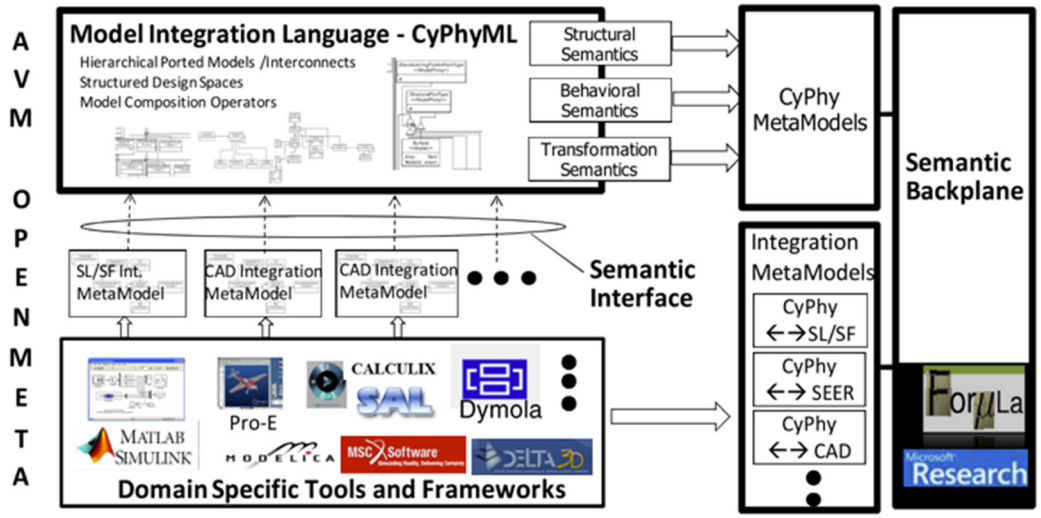


Figure 2.7: OpenMETA – model integration framework (courtesy of [44])

of design processes (included inspiration, decomposition, and integration) and domain of design spaces (included problem space, idea space, and concept space).

2.4.2 Ontological dimension of system-engineering frameworks

In the most general meaning of the word, framework is an arrangement of interrelated things with a particular objective based on a set of non-conflicting assumptions. Frameworks are created by human intention and thinking. In this sense, they are transcriptions of human mental models into some kind of formal representation. Cognitive frameworks are generated intuitively based on conscious thoughts and memories. They are used to guide daily activities and longer-range decision making without any formal representations [48]. Obviously, the content captured by a framework strongly depends both on the objective of creation and on the context of application [49].

For example, a theoretical framework is defined as a composition of (elements of) theories and a set of assumptions about their proper relationships. This type of framework can be used, for instance, to describe, explain, and predict a phenomenon in a particular discipline or across disciplines. In the field of software engineering, the notion of "framework" is interpreted as a set of cooperating classes that forms the basis of a reusable design for a specific class of software and that provides architectural guidance by partitioning thedesign into abstract classes and defining their responsibilities and collaborations.

Based on our study, it revealed that in the domain of system design, the most frequently discussed frameworks are:

General frameworks – (i) arrange entities and represent their relationships by flowcharts and causal diagrams, (ii) outline structure, modules, and entities, (iii) identify main topics related to the content, and (iv) list the requirements [50]. The overall and formal system frameworks can be sorted into this category [51], [52].

Conceptual frameworks – arrange a set of notions, conceptual definitions, buildingblocks and relevant variables, or concept variations in contexts [53]. Usually, these are presented as narratives, but can also be graphics, causal diagrams, procedures, and algorithms [54].

Logical frameworks – are typically used to define and consistently/coherently arrange abstract or concrete entities. They can capture logical or procedural connectivity of artefactual and/or process entities for specific purposes [55]. Primary representations are logical languages, logical expressions, logical variables, functions, predicates, and inference rules, but formal models, class diagrams, constraints networks, requirements trees, and computational procedures are pertinent too [56].

Architectural frameworks – specify the blueprints of system structures on various levels of abstraction or concreteness in both design and re-design processes [57]. They may represent the developed system in (i) static and dynamic dimension (entities, arrangement,

interactions, behavior), (ii) design domain dimension (contextual, functional/logical, and physical viewpoints) [58]. and (iii) abstract dimension (objectives, interdependences, instantiations) [59].

Component-based frameworks – focus on physical manifestation and relationships of designed or existing system components that are usually reusable, replaceable, and extensible modules and elements [60]. These frameworks provide guidelines for a component developer on how to produce custom components[61]. Typical representations are UML, class diagrams, use-case diagrams, schematic representations, and computational algorithms.

Model-based frameworks – reflect abstractions and simplifications of modeled systems [62]. A model-based framework, such as MoZaRT, is the basis and a tool for constructing of a set of models. Ontologies play the role of models [63]. The objective is to help designers to cope with scheduling analysis and to be more autonomous during the analysis stage [64].

Contextual frameworks – make conceptual distinctions by referring and organizing things and ideas that are supplementary, but influential in a situation [65]. They are also an analytical means to handle possible variations, models, or conflicts of contexts [66]. Frameworks for context data modeling and analytics form a special group of these [67].

Temporal frameworks – are means for arrangement of events and relations in chronological order [68]. They also provide underpinning mechanisms for articulating temporality from: (i) a diachronic/synchrony aspect, and (ii) an instant/duration aspect. Some frameworks capture spatial or procedural characteristics and uncertainties [69].

Composite frameworks include sub-sets or any arbitrary combinations of the previously mentioned types of frameworks in purpose-driven manner [70]. For instance, Feng, S. et al. (2016) proposed a framework for cyber-physical systems with a human in the loop [71].

2.4.3 Epistemological dimension of system engineering frameworks

Concerning the knowledge captured, framework specifications can be decomposed into four major elements: (i) the purpose and contexts of creating the framework, (ii) the set/kinds of entities included in the framework, (iii) the explicit/implicit relationships of the entities, and (iv) the explanation provided by logical interpretation of the framework. As it comes from the ontology of frameworks, the possible sets/kinds of entities can be theories, concepts, functions, definitions, components, variables, notations, and methods. They can be uttered verbally and textually, but are typically visualized in various graphical forms. The (possible) relationships of the entities depend on both their semantics and manifestations. Usual representations of relationships are causal relationships, hierarchical diagrams, logical expressions, topology graphs, connectivity diagram, flowchart, and mathematical models.

The specification of a framework (intent-driven, semantically proper interconnecting of

entities) is supposed to provide sufficient information about the consequences (implications) of applying the framework. It may provide the information in several forms, i.e., as prescriptive guidance, explanatory accounts, generative constructs, logical mechanisms, or predictive models. The main criteria for justification of a framework are consistency and coherence. A properly constructed framework captures a pattern that enables prediction at above the chance level. Aspects of validation are such as completeness, parsimony, and feasibility. The prediction provided by a framework can be in the logical, virtual, spatiotemporal, or material domain. Recognition of and elaboration on the logical/semantic pattern offered by a framework help recognize, for example, cause and effect relationships and prediction of other dependences.

Computationally, a framework is represented as a purposeful arrangement of data structures derived based on a network of concepts. This is, however, nothing other than a reductionist attempt to create computational models based on mental structures. In the interpretation of Bridgens and Lilley (2017), a framework is intended to lend itself as a tool, which can be used to combine information from multiple sources [72]. They presented a framework, which shows nteraction of material type, intrinsic and extrinsic properties, stimuli, physical material changes, and experiential responses to changes. A compositional example is a framework for error recoverable software, which provides a set of reusable abstractions for [73]: (i) defining recoverable units; (ii) detecting and diagnosing errors; (iii) providing coordination and control protocols necessary for recovery; and (iv) providing communication protocols between recoverable units. Handling compositional abstractions seems to be an important issue [74].

Our study had to reveal that no formal approach or standard model exists for framework specification in engineering, whilst the established SEFs are supposed to guarantee not only the synergy of the modeling knowledge, but also the consistent operation and event orders of the developed system. The frameworks currently used for the development of 1G-CPSs usually benefit from the principle and opportunity of composability. The existing frameworks that were found in the literature were usually developed by one or more approaches in five categories:

(i) Ad-hoc development

Ad-hoc development constructs a new framework from scratch. There is no standard procedure in a development of framework. It is a kind of bespoke systems dependingon a developer's expertise and viewpoints without any formal guidelines [75]. However, without a rigorous approach, it is not possible to achieve correct, efficient, reliable, and robust designs.

(ii) Holistic approach

The Holistic approach abstracts or adapts an existing meta-model to develop a framework. The iterative-incremental development at the core of the design process generates the methodology in a top-down fashion [76]. It initiates from the general lifecycle to the details of activities by using the requirements and methodology descriptions as a basis. A

development framework is usually constructed in the multiple-layer structure that provides the relationships of components in multiple viewpoints [43], [77]. The hierarchical layers also range from the high level of abstraction to the concreate structure of the system [78].

(iii) Model-based approach

The Model-based approach creates simplified representations of a system that helps a designer understand its characteristics at a certain level of abstraction. A model can be used as the first artefact driving the framework development process. In Anwar et al. (2019), they used the model-driven framework represents structural, behavioral and verification requirements at a higher abstraction level. The development framework is supposed to be a model integration which provides modelling formalism tointegrate multiple models [79]. In the different classes of models, semantics are needed for the integration process [44].

(iv) Architecture-based approach

An architecture-based approach conceptualizes a (logical) structure to support the development of the concrete architecture and specific functionality of a system. The architectural structure represents the first design choices in creating conducive and effective architecture descriptions, as discussed in [80]. Typically, the architectureimplied system engineering approaches capture different concerns (aspects) of system development. These can be sorted into three classes: (i) service-oriented architecture (SOA)-based frameworks [81], (ii) multi-agent system (MAS)-based frameworks [82], and (iii) other aspect-oriented frameworks [83]. Compared to SOA-based frameworks, MAS-based frameworks are more lightweight and more scalable in practice. In addition, multi-aspect formal frameworks have also been proposed to improve the functional performance of CPS.

(v) Component-based approach

A component-based approach operationalizes a framework through identifying and interrelating components retrieved from a repository. Components are constituents of systems characterized by their interfaces (e.g., an abstraction that is adequate for composition and re-use). Component-based approach manifests in a bottom-up way of working. In software engineering, this approach is in relation to object-oriented software design, and implementation [84]. The process commences with the evaluation of the known functional requirements [75], and identifies a set of components that satisfies the requirements. In the next step of the process, a large variety of components, each having different characteristics, are dealt with. An example is provided in [85]. The develoment framework itself helps identify those key components which play an important role in early modelling of a system. In some cases, a component-based framework allows the designers to consider components that are unavailable at the construction time and to integrate them into the application later, when the deployment is going on [86]. The major challenge of this approach is a composition of such components to ensure that they interoperate correctly. It needs semantic interpretation encompassing heterogeneous composition [87].

2.4.4 Analysis of the system-level functionalities of active frameworks

The functionality of a system is the total set of all its functions. It was noted in the previous section that the traditional frameworks play multiple roles in the design processes. These roles implied by their functionalities included:(i) supporting obser-vation and understanding of a phenomenon, (ii) addressing problems and proposingproblem-solving methods; (iii) offering means to combine cross-domain knowledge to create new concepts, (iv) providing a logical structure to verify conceptual ideas, and (v) providing multi-level architectural structure that can be seen as a blueprint for designing a system. The utilization of these frameworks was done in a passive manner. They cannot capture the behavior change in the procedural process of working environments.

We argue that a framework will be actively performed if it is able to recognize the changes in states of the observed system, to reason what situation is happening, and to take an action to response to the situation. The action can be considered in several levels of automation, namely: (i) warning a user when an anomaly is detected, (ii) proposing a decision-support recommendation to a user to solve a problem, or (iii) automatically solving the problem by the framework itself. This section aims at exploring and analyzing the system-level functionalities of SEFs by focusing on: (i) process monitoring; (ii) context modelling; (iii) situation reasoning; (iv) decision support; (v) problem-solving, as shown in Table 2.1.

As proposed in [88], the Trace and Trigger framework is an agent-based adaptive framework that helps agents detect adaptation requirements dynamically at runtime. It consists of two main mechanisms – a dynamic monitoring mechanism and an adaptation assistant mechanism. Functionally, it performs the event-based monitoring which observes traces in the event log file. By using event tracing, agents can publish, request, and cancel subscriptions dynamically in order to send and retrieve only the information that is interesting at each moment. Interestingly, two resembling frameworks as proposed in [89] and [90]. They have the same set of system-level functionalities, i.e., process monitoring, context modelling, situation reasoning, and decision support. Both of them collect information about the user's daily activities real-time. The former offers suggestions to an individual user, who responds dynamically to the situations at runtime. The latter provides a personalized routine plan to the user about physical activities. As indicated, only the former one performs in an active manner. According to the findings, we conclude that the frameworks having process monitoring functionality were able to perform in an active manner by collecting data in real-time, processing the data, and providing the services, which responds to the runtime operations [91]–[94].

2.4.5 Exposition of the findings and first propositions

The primary objective of the knowledge aggregation was to explore the need and the progress concerning SEFs for a compositional synthesis, modeling, analysis, simulation, verification, and validation of S-CPSs. The major findings and first propositions can be summarized as follows:

Table 2.1: Analysis of system-level functionalities of frameworks

frameworks	functions					avtive manner	application context
	process monitoring	context modelling	situation reasoning	decision-support	problem-solving		
Reasoning-based FW [92]	•		•	•		Y	driving safety warning system
OpenMETA [44]				•	•	N	designing CPSs
ACPSF [51]				•	•	N	design adaptive CPSs
Trace &Trigger FW [88]	•			•		Y	business process management
Context-aware adaptive FW [95]		•	•	•		N	e-health monitoring system
Situation reasoning FW [96]		•	•	•		N	situation inferring using IOT sensor data
Computational intelligence FW [97]		•			•	N	design support
Fog computing-based FW [98]	•				•	N	predictive maintenance in cyber-manufacturing
KN-based reasoning & recommendation FW [90]	•	•	•	•		N	personal activity for wellness recommender system
Context-aware reasoning FW [89]	•	•	•	•		Y	smart home recommender systems
Predictive process monitoring [93]	•			•		Y	business process management
Risk-based decision FW [91]	•			•	•	N	structural health monitoring
POLAR++ [99]				•	•	N	recommender systems
MAS-based self-healing FW [94]	•		•		•	Y	fault tolerance & automatic restoration in distribution networks

A SEF may play several roles in the development of these systems [100]. For instance, it may: (i) specify the range and boundaries of the target system, (ii) serve as a starting arrangement of concepts from various fields in an integral system concept (ISC), (iii) provide a rational basis for the functionality and feasibility of an ISC, and (iv) provide multi-aspect representations as blueprint for the implementation of an ISC. A SEF should safeguard the proper outcome of synthesis and modeling of S-CPSs, as well as of the other downstream activities of system development. The task is to identify or specify synergistically interoperating constituents, rather than only interfaceconnected, self-

contained components. A SEF should explain how to bring them into operational and architectural relationshipsconsidering varying contexts.

Considering the recent trends, we believe that the time has come for a rigorous investigation of this topic, starting with idea generation. It is with high probability that a single monolithic framework will not be sufficient for these purposes. Instead, some form of composite frameworks is foreseen that is able to capture multiple aspects of system manifestation in one construct. However, the majority of existing academic publications focused on traditional frameworks, which are, in the overwhelming majority of cases, static knowledge (concept, function, etc.) structures. They cannot capture procedural or behavioral changes, which are typical in constructive processes as well as in recommendation services. Changing the architecture and updating the knowledge of a static framework typically needs experts [101], and tailoring and using their contents in dynamic applications go together with heavy limitations. They are not able to learn and adapt to situational dynamics. Based on the analysis of system-level functionalities of SEFs, two essentialfunctionalities should be considered to equip a framework with an active manner: (i) real-time process monitoring; and (ii) runtime recommendation provision.

2.5 Investigation of the enablers of system-level reasoning

2.5.1 Phenomenon of system-level reasoning

Informally, system-level reasoning (SLR) is a capability that makes cognitive inference by intellectualized systems in an orchestrated manner. SLR is performed as recurrent cycles of sensing, inferencing, learning, and adaptation activities. In principle, it may involve all constituents of a system, no matter if they are hardware, software, or cyberware components [102]. The concept of SLR has arisen in the context of traditional (mission-critical) systems. Implementation of SLR needs more than only specific artificial intelligence algorithms, though each of the above mentioned four activities can be supported by or can be based on AI algorithms [103].They are assumed to be functionally,architecturally, and computationally coupled and complemented. System-level reasoning can be implemented on various behavioral modes using analytic and synthetic approaches.

A practical example of an analytic computational approach is the vehicle-level reasoning system presented in [104], which deduces information about the overall operational health of an aircraft. The reasoning is based on a combination of hardware devices and an artificial intelligence-based software application, whose computationalfunction is to generate conclusions from available knowledge using logical techniques of deduction, diagnosing and prediction or other forms of reasoning. SLR fuses information from the several sub-systems. Analytic SLR is also crucial to achieve smartness and other holistic qualities in large software systems. In addition, means of analytic reasoning about system-level properties have been pioneered in the areas of security and authentication[105]. In the case of CPSs, SLR is still a maturing research phenomenon. If the system does not

have self- managing intelligence, then usually a wide range of traditional abstraction-based modeling and functional simulations are considered in the design phase. A major challenge is that sub-system level data and results are not available on the system level.

A synthetic computational approach of SLR is based either on a single or on a composite logical theory and/or computational approach such as induction, deduction, abduction, retrospection, probability, analogy, learning, or production. The term‘synthetic’ means that the reasoning architecture, process, and results are produced by computational synthesis and that synthetic computational reasoning tries to achieve a relatively high fidelity in comparison with human reasoning. Artificial intelligence and machine learning research extensively studied these fundamental logical mechanisms, as well as their specific combinations [106]. Based on the analogy of human reasoning, common, modular, distributed, and collaborative reasoning approaches have been identified as distinct forms. The fact is that SLR intends to mimic human reasoning during information processing and knowledge development in complex situations, whereas classical artificial intelligence systems use dedicated methods to solve specific problems in a way, which can be measured and assessed concerning humanistic results.

Crowder et al. (2014) proposed a collaborative cognitrons framework called Synthetic, Evolving, Life Form – Dialectic Argument Search (SELF DAS). Its constituents work towards distinct and separate learning objectives [107]. The architecture included unsupervised and semi-supervised cognitrons. An active resonance theory-based, fuzzy, unsupervised neural network (FuNN) structure was used for the implementation of SELF DAS, which included shared fuzzy antecedents and conclusion classifiers, and processing nodes for fuzzy inference rules. It must be recognized that the number of possible links among the entities may grow fast if systems become bigger. It may lead to a combinatorial explosion, which means that no computer or living cognitive system could ever compute it [108]. Lieto et al. (2002) presented a rationale for knowledge-level integration in a cognitive system, DUAL-PECCS, which supported conceptual representation and categorization with two different cognitive architectures[108].

2.5.2 Knowledge as enabler of system-level reasoning

Knowledge is awareness and familiarity of the semantic meaning of information, and the cognitive potential of solving explicit problems in a given context. Recently, structured and coded knowledge has become an essential enabler of smart CPSs, but also other genres of engineered systems, such as AI applications [30] [109]. Coded human knowledge and self-acquired synthetic knowledge allows engineered systems to perform cognitive processes such as sensing, event detection, situation recognition, reasoning, action planning, and actuating through a feedback-controlled loop [20]. However, a purposeful integration of various kinds of knowledge is needed to perform specific task. For instance, (i) common sense knowledge is needed to reason about the utility of common (everyday) things, (ii) spatial-temporal knowledge is needed to describe system states at different points of time, and (iii) encyclopedic knowledge is needed to define actions and objects, as discussed in [110].

System knowledge is the symbolization process of knowledge that is deeply linked to learning and reasoning processes [111]. It can be obtained from different sources and captured by knowledge representation. The construction of new knowledge also demands the use of previous knowledge and different cognitive processes. It can be obtained from different sources and represented in several forms, including distributed, symbolic, non-symbolic, declarative, probabilistic, and rule-based [112]. The knowledge has been modelled that ranged from very informal as Object-AttributeValue scheme to strictly formal as OWL DL. Almeida and Lopez-de-Ipina (2012) claimed that ontology is regarded as one of the best approaches to transform context information into knowledge [113].

KnowRob is an example of knowledge processing system which uses ontology representing domain-specific knowledge of daily-tasks, household objects, and events and temporal things needed by service robots [110]. As shown in Figure 2.8, the knowledge contents and functionality of the KnowRob can be extended with the difference modules i.e., for reasoning about knowledge and for grounding the knowledge in the robot's perception and action system. However, in reality, it is difficult to create an engineering ontology manually that could cover all permutations of the enormous number of entities, properties, and attributes. Technically, as the number of triples in the ontology increases, the inference time for environment actions becomes unsustainable [113]. This is actually a well-known drawback of the knowledge engineering-based approach to knowledge modeling.

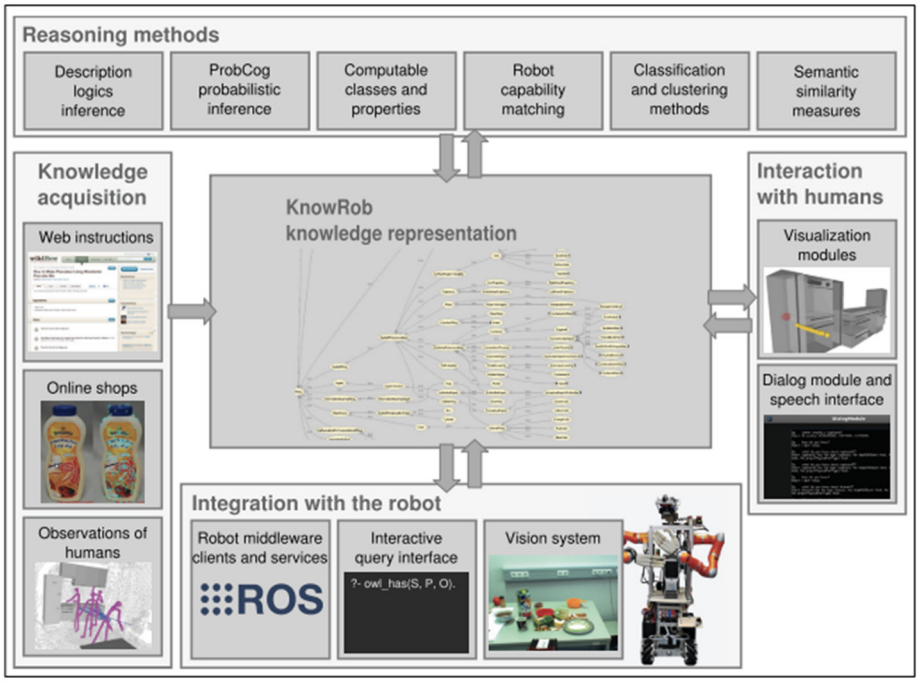


Figure 2.8: Architecture of KnowRob – a knowledge processing infrastructure for cognition-enabled robots (courtesy of [110])

2.5.3 Awareness as enabler of system-level reasoning

Awareness is a product of knowledge processing, and monitoring [114]. It encompasses context, situation, and self-awareness. Systems operating in a dynamically changing environment should be able to build up awareness about (i) their context of operation (e.g., need for dynamic adaptation of tasks and objectives as response to external factors), (ii) the situation they are operating in (e.g., understanding of the impact of the environment on the operation), and (iii) self-awareness of themselves (e.g., understanding of the system's abilities and the availability of its resources for performing operations).

Context can be considered as a kind of knowledge [116]. It refers to any information that used to characterize a situation of an observed entity. A system probably does not recognize a situation from an isolated entity. Context awareness needs multiple entities i.e. person, place, physical or virtual object that combined to model the semantic context to understand the environments and make an adaptation [117]. It also implies an effective exploitation of contexts.

To provide different context treatments, it can comply with the general lifecycle of context awareness included four primary phases as shown in Figure 2.9 [115], namely: (i) context acquisition – to obtain necessary context data; (ii) context modelling – to represent contexts in a machine-readable and process-able form; (ii) context reasoning – to derive high-level contexts from available contexts; and (iv) context dissemination – to distribute useful contexts. It is usually assumed that context modelling using knowledge engineering techniques will create complete accurate models. Different approaches have been used for reasoning considering certain context information. The most widely documented in the literature are, for instance, (i) fuzzy logic, (ii) probabilistic logic, (iii) ontology-based reasoning, (iv) Bayesian networks, (v) hidden Markov models, and (vi) the Dempster-Shafter theory of evidence [118]. Each of these approaches has its own advantages and disadvantages, as it is evidenced by the review presented in [119]. In order to support knowledge-intensive context reasoning, ontology-based models proved to be the most promising technique to yield meaningful context information [115].

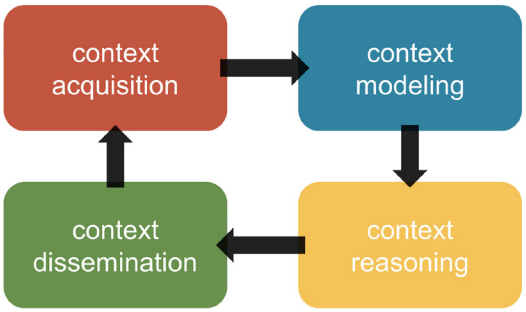


Figure 2.9: Life cycle of context awareness [115]

Situation awareness is a computing paradigm, which usually involves the use of the concept of the situation in real life. If a situation is specified as a set of relations with other objects, then both the objects and their relationships may change with in terms of time and location. In the framework for cognitive situation modelling [120],

situation awareness is a part of situation management, which is based upon the steps of sensing and perception, and is aimed at building an understanding of a current operational situation. Situation modelling and inferring can range from using simple conditional rules to application of more complex techniques. They are classified into specification-based techniques (e.g. formal logic, spatiotemporal logic, and evidence theory), and learning-based techniques (e.g. Bayesian deviations, Artificial Neural Network, and web mining) regarding their correlation to increasing complexity of problem descriptions [121].

Self-awareness can be seen as a higher level of situation awareness [122] forinstance, a system is continuously aware of its operational and servicing states and behaviors. In other words, self-awareness refers to the capability of a system to gather and process information from its environment and to autonomously understand the situation of those external and internal entities that can affect the system in the accomplishment of its operational goal [123]. This capability is based on self-monitoring that is typically implemented by a network of hardware and software sensors. From the engineering perspective, the self-awareness can be considered as an emergent property of the collective systems [124]. As a paradigmatic feature of S-CPSs, self-awareness plays a crucial role in realizing dependable operation under changing circumstances during runtime.

2.5.4 Reasoning mechanisms as enabler of system-level reasoning

Reasoning is the ability to manipulate previously acquired knowledge to draw novel inferences or answer new questions [125]. Consisting of a composition of computational algorithms, a reasoning mechanism is a means to operationalize smart systems. Various reasoning methods were applied in the context of smart systems, intelligent systems, and autonomous systems. Rule-based reasoning offers a natural way of handling and inferring knowledge. A rule-based knowledge system that features modular structure can easily be extended with additional rules, and provides a uniform representation of knowledge [126]. However, it provides limited expressiveness to describe certain complex features and therefore cannot fully exploit the potential offered by events. Case-based reasoning is frequently used in the decision- making process [127]. It generates a conclusion or new knowledge based on the available knowledge or information at hand [128]. Ontology based reasoning is used for conceptualizing the relationships between entities to create knowledge. It is typically combined with other reasoning methods such as rule-based reasoning in order to infer a situation from context information [129], or case-based reasoning in order to automate the decision-making process.

Probabilistic reasoning, such as Bayesian Networks (BNs), and Hidden Markov Models (HMM), is appropriate for reasoning with uncertainty [69] .BNs are used for the analysis of data and expert knowledge, especially in the context of uncertainty. They can easily process probabilistic knowledge from different sources in a mathematically coherent manner [130]. HMM’s have more flexibility to capture unobserved variables and thereby provide

a basis for reasoning about emergent behavior of the system. Fuzzy logic is a well-known approach to deal with uncertainty, imprecision, and other non-deterministic problems [131]. Hybrid reasoning approaches have been proposed in the more recent relevant publications. Combining fuzzy logic with ontologies, and probabilistic modelling, it can cope with qualitative interpretation of probability, treat probability with natural language expressions, and human- like decision making [132]. The degree of integration can be performed in several models [133] i.e. sequential processing, embedding processing, and co-processing.

In complex reasoning mechanisms for such smart CPSs, however, they require multimodal processing with more specific temporal, non-monotonic reasoning, and learning from data. For example to be able to realize a situation, to be aware of the changes in the situation, and to make decisions based on a dynamic situation. Many factors should be taken into consideration to integrate multiple reasoning methods in the procedural reasoning mechanism i.e. domains of applications, an objective of the developing systems, nature of obtained data, and required system performances [134]. In addition, for dynamic processes, reasoning mechanisms should be composed during runtime with high level of interoperability. Although, some methods are able to work together in several degrees of integration, many of these methods are not yet interoperable. Their computational components need to be modified or require an interface to couple them seamlessly. This implies the need for different conceptual framing of reasoning mechanisms and different design principles, since they need holistic compositional approach in terms of the implemented reasoning process and synergy in terms of the generated knowledge.

Several frameworks for reasoning have been proposed in the recent literature. For example, system-level reasoning in AI is usually summarized through the expression '*Sense-Think-Act*' [23] that mimics human thinking by using deductive reasoning [135]. Belief-Desire-Intention (BDI) paradigm is one of the operational architectures commonly suitable used for building complex agent-systems. A classical framework embedded this architecture is the Procedural Reasoning System (PRS) [136]. It includes three main processes: perception, interpretation, and execution. Another example is the FUSION framework, which implements a *Detect*, *Plan*, and *Effect* procedure. The computational implementation of this procedure can be used for designing and implementing the underlying adaptation logic of adaptive software systems. For instance, it supports rule development for adaptation, such as *if the system works (e.g. satisfies the user, obtains the goal), do not change it; when it breaks, find the best fix for only the broken part* [137]. The *Sense-Plan-Act* loop is also used as a reasoning concept for self-adaptive systems [102]. These cycles are basically executed by using rule-based reasoning which is implemented based on the principle of deduction [138].

2.5.5 Decision making as enabler of system-level reasoning

A decision is interpreted as a cognitive process of choosing (expectedly, the best) alternative from the various possible actions for attaining a given goal or multiple goals. Decision-making often involves the integration of data from multiple sources, and harnesses knowledge from multiple domains [139]. The goal of a decision-making process is to

choose the best alternative from a set of possible alternatives that satisfies an objective, or multiple objectives. An optimization is a common problem solving method in decision making [140]. In real-world problems, multiple objectives are always taken into account. They may possibly be in conflict with each other. As the number of *m*-objectives increased, the number of solutions increase exponentially [141]. The challenge is how to handle the computational explosion issues. That is why an optimization model with multiple objectives is not suitable in practice.

The process needs to evolve and adapt in a dynamic situation when a decision-making process is confronted with new situations, goals, and kinds of data. This requires reasoning methods, which is often based on more than logical conclusions [143]. In a human decision-making process, system-level reasoning can be made as a closed loop, for example *Observe-Orient-Decide-Act* (OODA) loop [142]. A decision maker performs the OODA loop repeatedly as shown in Figure 2.10. As discussed in the above publication, the systematic procedure of the OODA loop is as follows:

Step I: *observe* the facts by capturing, fusing, and filtering data about the entities and environment,

Step II: *condense* the information from the facts to Orient with the revealing situation by applying prior knowledge,

Step III: *formulate* hypotheses to explain the observations and Decide based on the best scenario, and

Step IV: *act* following the internal guidance from the orient process and test the hypotheses.

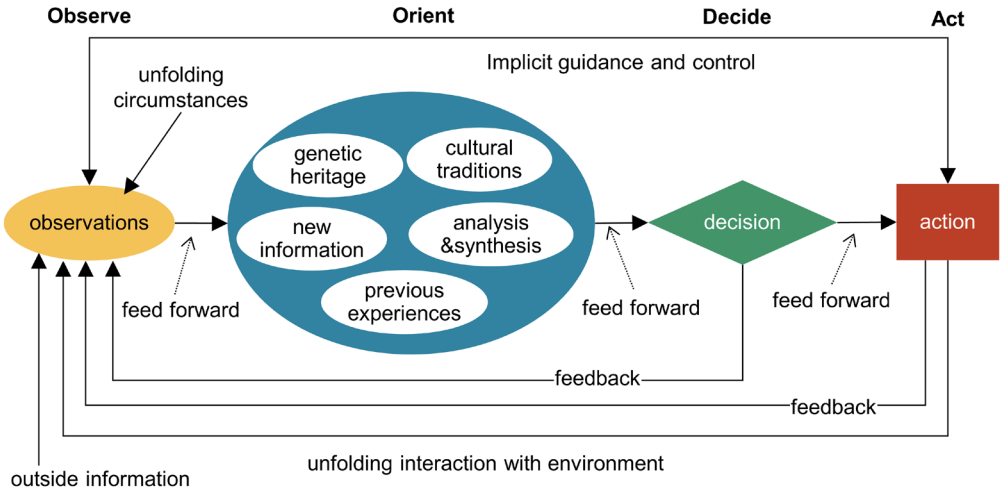


Figure 2.10: Feedback structure of the Observe-Orient-Decide-Act loop (modified based on [142])

Corresponding to the OODA loop, Knowledge Intensive Data System (KIDS) framework is an example of self-adaptive decision making [144]. The framework proposed a flexible data structure based on ontology. Four main reasoning functions are represented by Classify-Asses-Resolve-Enact (CARE) loop. Through the reasoning processes, it transforms input data into facts, perception, hypotheses, and directives, respectively. Each of these is the input of one reasoning function and the output of another. The manifestation of CARE loop can be accomplished in two ways: (i) to customize the reasoning function by adjusting its parameters; (ii) to select different reasoning methods when the current used function is no longer adequate. For instance,reducing data into facts at the classification task would generally use statistical reasoning, but in some cases, logical and probability reasoning was also preferable. This can be explained by various reasons such as (i) change in the environment, (ii) change in the goal of operation, and (iii) a new kind of incoming data construct. The functional and architectural adaptation is needed to response the changes in the decision-making process.

2.5.6 Adaptation as enabler of system-level reasoning

System adaptation is the planning of adaptation based on the outcome of previous processes. In the context of engineered systems, implementation of system adaptation is inspired by biological and natural systems, which have the ability to modify themselves according to a new condition when its environment or purpose changes [145]. The modification is done by adjusting the parameters of the system in response to change in the system itself or in their environments. It also adapts to similar settings without explicitly being ported to them and adapts to solve a new problem [146]. However, no absolute optimization exists in complex systems [147]. The operation of these systems changes to a desired stable state over time. Frequently, there are multiple point attractors [148]. Although the system can modify the parameters and somewhat reach the desired state, it might be shifted to another point as a consequence of the actions. Therefore, the self-adaptive capability should incorporate reasoning about the objective of the system operation, investigating possible strategies for performing adaptation, and planning and executing adaptation plans based on available cyber and hardware resources [9].

In the self-adaptive software research community, self-*properties are organized into levels where self-adaptiveness is at the top, while self-awareness is a primary level like context awareness [149]. A self-adaptive system is typically implemented by control loop mechanisms [145]. Self-adaptive control mechanisms typically include sequential iterative processes of: (i) sensing the context and reasoning, (ii) deciding what kind of adaptation is required, and (iii) implementing the adaptation by reconfiguration [150]. An *Event-Condition-Action* (ECA) rule is usually implemented in the self-adaptation of service based processes [26], [151]. It is also used to describe different responses to various runtime events. The semantics of the rule is as follows: '*when the event has been detected, evaluate the condition, and if the condition is satisfied, then execute the action'*. The general syntax is 'on event-if *conditions*-do *actions*' [152]. In a software adaptive system, *Monitoring-Analyzing-Planning- Executing with knowledge* (MAPE-K) loop is one of the most well-known adaptation mechanisms [18].

Although the terms and notions used for describing the above self-adaptive methods are different, the general process of self-adaptation is implemented based on a rather common concept. This concept includes the following: (i) perceive the current state from input data, (ii) monitor and analyze changes, and (iii) plan and adapt the process/system to the optimal state. The concept of closed loop mechanism limits the possibilities of adaptation when open-loop interaction with the external environment is becoming a fundamental aspect of the system [144]. Zhou et al. (2017) extended the self-adaptation process of CPSs which included the interaction of cyber world and physical world [35] as shown in Figure 2.11. The extended self-adaptation process includes the interaction of long-term and short-term loops. The long-term loop provides decision-support based on the causal reasoning of the MAPE-K loop. The short-term loop is the conventional feedback control loop for dependable decision process that influences the operationalization of the physical world. However, approaches to true self-adaptive behavior are still in their infancy.

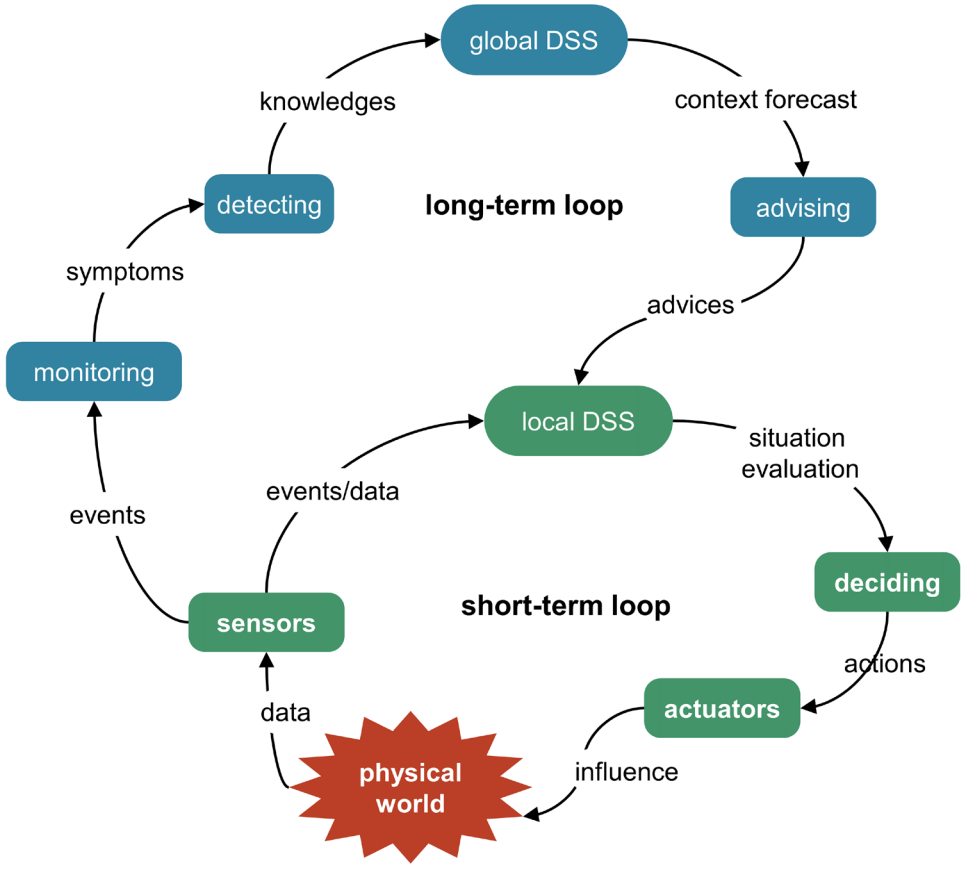


Figure 2.11: The environment-in-the-loop selft adaptation process of CPSs (modified based on [35])

2.5.7 Recommendation generation as form of system-level services

2.5.7.1 Types of recommendation systems

Recommendation systems (RSs) have been gaining increasing popularity in various areas of application. Typical examples are e-commerce systems (e.g., Amazon, e-Bay), social networks (e.g., Facebook, Twitter), personalized recommendations (e.g., Spotify, Netflix), and knowledge-sharing platforms (e.g., ResearchGate, Mendeley). The systems help users to find preferred items in the collection and facilitate the task of deciding on appropriate items. Recently, the concept of recommendation systems has also attracted considerable attention from the domain of multi-disciplinary decision-support systems, such as medical diagnosis systems [153], traffic management systems [154], and design support systems [155]. Often, these systems are also referred to as recommender systems. They manifest in software tools and use algorithmic techniques to provide advisory recommendations concerning the most appropriate content to a user. With a view to their real-life applications, recommendation systems are classified into three main classes: (i) commercial recommendation systems, (ii) social recommendation systems, and (iii)engineering recommender systems.

Commercial recommendation systems are sub-classes of information filtering systems that deal with the problem of information overload [156]. Driven by business intentions, the goal of commercial RS is to support sales and to increase purchase intent [157]. They are typically implemented as websites that collect customer data and automatically analyze them to generate customized recommendations for the customer. Widely-used techniques to match the user's preference and the purchase items are (i) memory-based advising, (ii) collaborative filtering, (iii) content-based filtering, and (iv) demographically-based recommendation techniques [40]. However, there is a need for providing better customer experience than that just offering the best item by means of pure data-driven techniques [156]. Towards this end, developers integrate merchandising rules into the recommendation generation process. These rules include for instance, rules to prevent cold start problems, or rules to prevent out-of stock goods.

Social recommendation systems use information about the user's preferences, environmental influences, and social relations to other users to predict the most useful recommendation [159]. Under this assumption, social recommendation leverages user correlations implied by social relations to push information to target groups both timely and accurately [160]. Various collaborative filtering techniques are adopted in combination with social correlation theories. For example, trust ensemble, trust propagation, and social regularization are applied at building social recommendation systems [158]. Originally published by the abovementioned authors, the concept of a generic user trust network is shown in Figure 2.12, where: u_i is users, v_j is recommendation items, the blue edges are relations of users, and the numbers on the red edges mean rating the items v_j by user u_i.

Engineering recommender systems have been developed alongside the concept and

principles of recommendation systems, but they are applied in specific engineering application domains. In the studied literature, we could not find a generally accepted definition for these types of recommendation systems, and there seemed to be no agreement with regard to their functionalities and implementations. In the rest of thisdissertation, we use the term *'recommender systems'* to distinguish these systems from the tradition commercial and social recommendation systems. In the promotion research, we focused on theoretical concepts, which underpinned engineering recommender systems. Cena et al. (2020) defined recommender systems as "systems that produce individualized recommendations as output, or drive the user in a personalized way to interesting or useful objects in a space of possible options" [161].

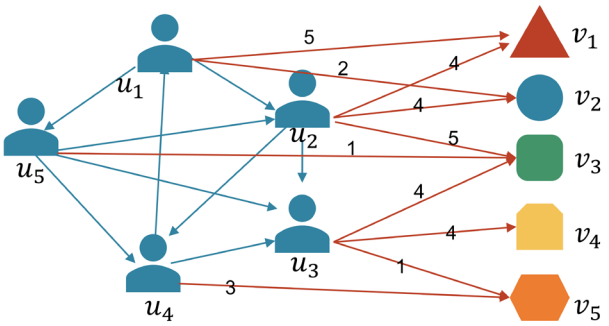


Figure 2.12: User trust network involved in social recommendation generation [158]

From our point of view, recommender systems integrate recommendation services into such smart systems. They can profit from the traditional RSs by allowing a user to augment the recommendation engine with quantitative and qualitative information and users' preferences which cannot be captured in pure data-driven algorithms [162]. For example in the domain of design-support application, Jannach et al. (2016) developed add-on recommendation service to the RapidMiner framework to support the user during the development of ML-based model by recommending additional operations to insert into the currently developed workflow [163]. In the domain of medical treatment services, Bao and Jiang (2016) developed the universal medicine recommender system by applying data mining technologies to the medical diagnostic and recommending the proper medicine for a patient [164].

2.5.7.2 System-level operation of recommender systems

Based on the exploration of the related literature, the following system-level activities will be addressed below: (i) information filtering, (ii) information/knowledge aggregation and structuring, (iii) context management, (iv) problem-driven inferring and reasoning, (v) recommendation generation, and (vi) interaction/communication with stakeholders. The purpose of the analysis is to find reusable knowledge, principles, and solutions about the core functionalities.

(i) Information filtering

Engineering recommender systems should purposefully filter in order to avoid theinformation overload and to make the best suggestion to the user. The simplest solution to information filtering is a popularity-based approach. Basically, it filters the ever-growing

flood of information (which often happens in real-time on social network websites) with the goal to identify the news stories that are the most popular and most current at that moment [165]. However, this filtering technique does not consider users' profiles or other relevant information in the filtering process. The information filtering techniques used to build recommender systems are classified into (i) collaborative filtering methods, (ii) content-based filtering methods, and (iii) hybrid filtering methods.

Collaborative filtering generates recommendations using information about the rating profiles of different users. It can be done because users give explicit preference judgments about items in the form of ratings in the process of data collection [166]. The major limitation of the collaborative filtering method is sparsity of the rating matrix. The recommendation cannot be made unless and until the item is either rated by other users or correlated with other similar items. If users rated only a small subset of the available items then most of the cells in the rating matrix are empty [167]. Content-based filtering recommends items, which are similar to the ones the user preferred in the past. The rating of items is based on the ratings assigned by the user to items that are similar to the new item. The success of applying content-based filtering depends on two major conditions, as it is discussed in [168]: (i) each recommendation item needs to be characterized by well-defined features; and (ii) the users should recognize how these features relate to their requirements. Content-based filtering is not free of limitations. Typical ones are: (i) limited content analysis (e.g., insufficient identified features of items), (ii) indistinguishable features of different item classes, (iii) overspecialization from users' rating, and (iv) less reliable rating due to additional new users. Hybrid filtering methods commonly combine two methods to overcome the limitations of both methods in order to make the system more robust. Seven categories of hybrid recommendation approaches were proposed (i) weighted, (ii) switching, (iii) mixed, (iv) feature combination, (v) feature augmentation, (vi) cascade, and (vii) metalevel [169].

However, all of these methods based on data-driven techniques are known to suffer from a cold-start problem. The reliability of recommendation is still low due to an initial lack of ratings. In the engineering recommender systems, for example in emergency landing planner [162], it assists air-traffic control operators with choosing a diversity airport for distressed aircraft. These data-driven filtering methods are not applicable for these types of recommender systems. The post-evaluation of the recommendations is needed to examine the quality of recommendations based on whether the users agreed upon the solution provided by the systems. The specific set of criteria is defined for the evaluation. These criteria as well as technical constraints will be converted into the decision criteria. A user will be able to provide the information by using weighting techniques or query-based filtering to pre-configure constraints and user's preferences [66]. As result, the possible situations are selected for further processing the recommendation generation.

(ii) Information/knowledge aggregation and structuring

Conventionally, recommendation systems operate on an extensive user-item relationship matrix. As mentioned in the previous sub-section, the rating system underpinned by this

approach is not applicable in specialized recommender systems that isfor limited numbers of users to rate the recommendation items. Recommender systems use formal knowledge representations, which can be processed computationally during recommendation generation. The literature states that knowledge in recommender systems can be stored in various types of structures. Considering the plans for the development of ARF, four types of knowledge representation were investigated: (i) lookup table; (ii) digital corpus; (iii) knowledge graph; and (iv) concept ontology.

A lookup table is one of the simplest ways to create a formal knowledge. It contains three primary elements, decision rules, decision conditions, and actions. A rule set is defined by a Boolean value to map the decision conditions to the corresponding actions. The conditions in the rules and the input data in the schema of the data source must be compliant and mapped to one another for successful integration. The knowledge in the lookup table is static. If the data requirements or mapping for the conditions of the rules are not specified and well defined in advance, recommendation generation will fail [90]. Knowledge is recorded in the digital corpus. It usually used in the recommender systems that already stored a vast collection of records in machine readable format for instance, an Electronic Medical Record (EMR) system [153], and knowledge services for product design developed by [155]. To retrieve relevant documents from the knowledge sources, text similarity techniques are typically used to measure the degree of similarity between the queries which appear in the documents [170].

A knowledge graph (KG) is a heterogeneous graph, where nodes function as entities, and edges represent relations between entities [171]. In tradition recommendation systems, the user-item relationships are converted into a graph model where users and items represent the nodes, and edges express the interactions between user-user or user-item [172]. It generally leverages the semantic representation of user/items in the KG for recommendations. In addition to the user-item graph, Cognitive map (CMs) and Bayesian networks (BNs) have also frequently used to represent a formal knowledge in recommender systems. Cognitive maps (CMs) construct the causal relationships among concepts that presented at nodes and directed links defining the relationships among them. CMs are extended by fuzzy logic as called fuzzy cognitive maps (FCMs) that defines fuzzy set into the casual relations between concepts [173]. They are an effective tool for modelling decision support systems and time series predictions.The value defining between concepts can be changed over time due to the dynamics of the model. Hence, FCMs can be applied for a dynamic recommendation generation. They are also constructed in both machine-readable and human-understandable forms. The drawback of FCMs is that the construction of FCMs is time-consuming with a large data set [174].

Supporting such product design process, and manufacturing process, demands a representation of process flow models. They include multiple decision points throughout the entire process, for example selecting semi-finished parts, selecting production techniques, and sequencing manufacturing process [175]. It requires knowledge at every single point to support the decision-making. The consequence from the preceding decision

also influences the next decision like a dependency graph. Bayesian networks can be developed for modelling the process flow. It is a probabilistic model representing random variables and conditional dependencies in a directed acyclic graph. Through this method, variable connections can be qualitatively defined with a network structure, and correlations between related variables are quantitatively determined using conditional probability distributions [176]. A very useful aspect of BNs is that they can produce good prediction accuracy even with rather small sample sizes. Discovering the patterns in the BNs is also possible. However, BNs support only linear processes, thus it is impossible to represent the iterative processes with feedback loops.

Ontologies are an expert-defined standardized common vocabulary describing the knowledge of a domain [177]. They are expressed in a form of a hierarchy concept tree. The most common role of ontology in recommendation system design consists of providing a taxonomic classification of items [178]. In recommender systems, ontology is an explicit specification of conceptualization consisting of classes, relations, functions, and other objects in the shared domain knowledge. For example, in lighting system design [179], there are various classified concepts and related objects, for instance daylight sensor, lighting controller, power driver, light, communication unit, etc. These concepts should be closed to the objects and relationships in the domain specific knowledge. In general knowledge-based systems, ontologies are increasingly being used because they provide the flexibility, extensibility, and generality to bridge the gap between the requirements of mapping domain knowledge into machine-readable and human-understandable formats [180]. In knowledge-based recommendation systems, ontologies were applied to reduce content heterogeneity and improve content retrieval [181]. The major limitation of using ontologies is the difficulty of transferring specialized knowledge from domain experts to abstract and effective representation.

(iii) Context management

In the context-aware recommendation systems, they promote incorporation of additional contextual information such as time, day, season, user's personality along with users and items related information into recommendation process [182]. Context information found in the recommendation systems can be classified into two types: (i) direct context and (ii) indirect context. The direct context is basically derived from user's profile, user's preferences, features of recommendation items, and rating of items. These are the essential information which directly influence the generation of recommendations. The indirect context refers to additional information for instance, locations, time, and temperature. These contexts occur in the surroundings which do not directly influence the recommendation generation, but they can capture the state of the observed systems in a specific moment. Observing the changes in indirect contexts allows for describing the dynamic situation. Taking the indirect context into account in the recommendation process produces more accurate results [183]. It potentially supports the dynamic recommendation generation.

The context information can be represented in either explicit or implicit ways. The explicit context related to user's profile and recommendation items is typically

managed in a matrix representation. In the user domain, two aspects of implicit context can be considered. One is the level of attention a user has while the systems recommend items to him as well as the degree of interruption a user is willing to accept [184]. These contexts are not shown explicitly in a formal representation. An input from users is needed to expose the context. Discovering patterns in the dataset requires data mining techniques. Another types of context information which mainly include: (i) attributive context refers to the attributive characteristics of an entity (e.g., color, dimension, temperature, quality), (ii) temporal context refers to time (e.g., time-stamp, date, day time, night time, month, year), (iii) spatial context refers to space (e.g., represented in latitudeand longitude data, Cartesian coordination, relative locations and direction), and (iv) spatio-temporal context which considers both data related to space and time. The direct and indirect context information can be constructed in the formal representation for instance in a data table with the grading data format which contains instances about users, recommendation items, event, ranking, and time stamp [185].

(iv) Problem-driven inferring/reasoning

Recommender systems support automated information filtering by using sophisticated algorithms that incorporate preferences rules and heuristics to reduce a potential large number of recommendation items into a smaller cardinality and more manageable subset [186]. The inference mechanisms can be employed to reason about context information to generate a personalized recommendation supporting decision-making in order to solve a problem. In this section, four reasoning approaches are discussed, namely: (i) rule-based reasoning, (ii) knowledge-based reasoning, (iii) model-based reasoning, and (iv) machine learning-based reasoning, which are most frequently used in problem-driven reasoning mechanisms.

Rule-based reasoning is a simple approach to equip a system with intellect and then to manually enter an expert's knowledge or automatically infer probabilistic rules [187]. This method can work without any specific knowledge about the application context. However, it is too expensive in the case of systems with a large amount of data, due to it is manual nature [188]. Knowledge-based reasoning finds solutions that match desires and requirements based on domain specific knowledge about users' profiles, recommendation items, and context information [159]. It is useful in domains where rating-based systems do not work. Case-based reasoning, for instance is applied to solving a problem based on the information/knowledge in the historical cases. It matches the similarity between the desired case with input provided by a user and the cases that are available in the repository [189]. Ontology is informative for knowledge representation which is defined as an explicit specification of conceptualization.Ontology-based reasoning is used in semantic inferring by identifying the similarities among concepts and information provided by a user and select the most similar one into the process of recommendation generation [181]. Using ontology-based reasoning is limited to static recommendation generation due to the domain-specific knowledge.

Model-based reasoning is an analogical inference method which uses deductive

logic to explain the physical world. Several methods have been used in recommender systems. For instance, graph-based reasoning uses a graph representation, where users and items are represented as nodes, and the edges express the interactions between user and user, or user and item. It is capable to model various implicit relations between users and items which reveal the preferences of users on consuming items [171]. Probabilistic inference (using means of Bayesian networks, Dempster-Shafer theory, and fuzzy logic) has been extensively used in designing recommender systems to handle uncertainty, impreciseness, and vagueness in item features and users' behavior [184].

Various forms of machine learning-based reasoning are typically applied to discover the patterns of a vast and high dimensional data set [190]. Recommending classification algorithms based on k-NN method was proposed in [191]. It aimed at assisting a user in selecting algorithms from a large number of candidates for a new classification problem. In [192], the collaboration of clustering and classification algorithms was applied to the medical advice and diagnosis system. The clustering model is used to cluster all patients' medical advice rating into the similar objects. The classification model analyze spatients into distinguished group based on the identified features. These classes are interpreted to a set of medical advices. Another example, decision tree classifier [193] was used in context aware recommender systems. The ID3 algorithm was applied for learning users' contextual preferences in the recommendation process and predicting unknown ratings using collaborative filtering approach.

(v) Recommendation generation

Recommender systems are directly involved in assisting users to make decisions and satisfying their current information need. User preferences are always changing depending upon a range of factors, for instance context, time, location, trust, and new experiences. A static user profile and historical preference cannot judge the actual preference of a user over a period of time. The changing needs of user preferences influence the process of recommendation generations were discussed: (i) static recommendation generation; and (ii) dynamic recommendation generation. Table 2.2 shows examples of application context in different types of recommendation generation and responses to the recommendation.

In static recommendation generation, a recommendation is made at any instant of time depends on the given relations of user preferences and item features. Traditional filtering techniques adopt a static view of the recommendation process and treat it as a prediction problem [194]. The process of recommendation generation works in a sequential manner where at each stage a new list is recalculated based on the users' past feedback [195]. Two common ways to obtain relevance feedback are to use information given explicitly (e.g., rating, text comments, evaluation of recommendations) or to get information implicitly (e.g., purchased history, time spent) from the user's interaction. Although the recommendation generated is inherently dynamic to some degree, the changes of user preferences and item features over time in the current scenario are beyond its coverage [196]. Without the consideration of the dynamic aspect of users' behavior and involved context information, the recommendation generation is still static. It cannot recommend different types of items

that change with time based on their underlying relations to user preferences and working environments as seen in [188] and [197].

Table 2.2: Types of recommendation generation (RG) and the responses to the recommendation

	static RG	**dynamic RG**
reactive responses	web search engine	stock market
	routing navigator	online shopping
		personalized activity recommendation
active responses	manufacturing processing planning	fire evacuation
		traffic management
	predictive maintenance medical diagnostic	fault detection in the distribution network

Dynamic recommendation generation offers the recommendation which is sensitive to the changes of situation at the moment in time [198]. It is able to learn user preferences and context information from a dynamic data stream [199]. Rana and Jain (2015) defined dynamic recommender systems as "the systems which are able to capture the temporal changes occurring in the different domains i.e., user and items related data as well as other environmental changes implicitly or explicitly and accordingly modify their recommendations to the users" [196]. In online recommendation generation for example, the recommendation engine analyzes the current state of user's preferences based on the actual activities done by the user and produces the immediate response for the user [200]. In this sense, dynamic refers to information retrieval patterns over time e.g., the order in which different items are searched by a user [201].

Dynamic recommendation generation is characterized by six aspects [196]: (i) temporal changes – considering changes in user's preferences in time when selecting the next item to recommend and recognize temporal characteristics of recommendation items, (ii) real-time dynamics – considering real-time data related to user behavior and generation of recommendation immediately in response. (iii) context – describing a particular state of the user or the environment at any given period of time, (iv) novelty of recommendation – considering the discovery of new and original items for users, (v) serendipity of recommendation – considering the quality of propensity for making fortunate discoveries while looking for unrelated items; and (vi) diversity of recommendation – which means the variety of choices breaking the barrier of similarity. We focus our attention on the first three aspects for a development of ARF.

(vi) Interaction/communication with users

The section discussed the interaction of stakeholders and the recommender systems in term of communication modality. We assume that the recommender system monitors the behavior of its users over time and then presents a customized set of recommendations in pre-defined navigational situations. Since the recommendation is made, the user takes an action and provides feedback to the systems.

In the conversational recommender systems [202], three types of input and output modalities can be designed as shown in Figure 2.13. In the general recommender system, two main

forms of inputs and outputs, either as the only modality or combined in hybrid approach are: (i) based on form and structure layouts, as in a tradition webbased application, (ii) based on natural language either in written or spoken form. Approaches that are exclusively based on structure layouts include textual-based representation [203],[204] (e.g., Q&A multi-turn dialogue, textual description), and query-based form [66], [205] (e.g., binary input, string-based, selection of choices). Natural language interaction-based approaches were usually found in chatbots. They are also implemented in a smart speaker like Amazon Alexa and Google home. Hybrid approaches that combine natural language with other modalities are not uncommon.

In the application environments, the design choices depend on several factors such as types of application context, the level of user controls, and the support devices. The possible approaches are interactive map as seen in [197], the recommendation shows the best route in the map according to the query of user's preference, representing in 3-D space in interior design application [206], or combined graphical model, text descripting and navigator as seen in the recommender system for emergency landing planner [162]. For users' responses, it is mainly based on structure layouts for rating, text-based description in term of user reviews, comments , shared experiences [207] and post-evaluation of recommendations [162]. The most common approach is the rating system which is represented as a unary value (showing only the relevant items), binary (allowing to distinguish between good and bad items), or as a numerical value on one finite scale [167].

2.5.8 Compositionality in system-level reasoning

The term *'compositionality'* was first introduced in the fields of linguistics, mathematics, and semantics. In linguistics it is defined as the principle to realize the meaning of a complex expression that is determined by the meanings of its constituents [208]. In the context of systems, the set of basic words are the components and modules, and the modifiers

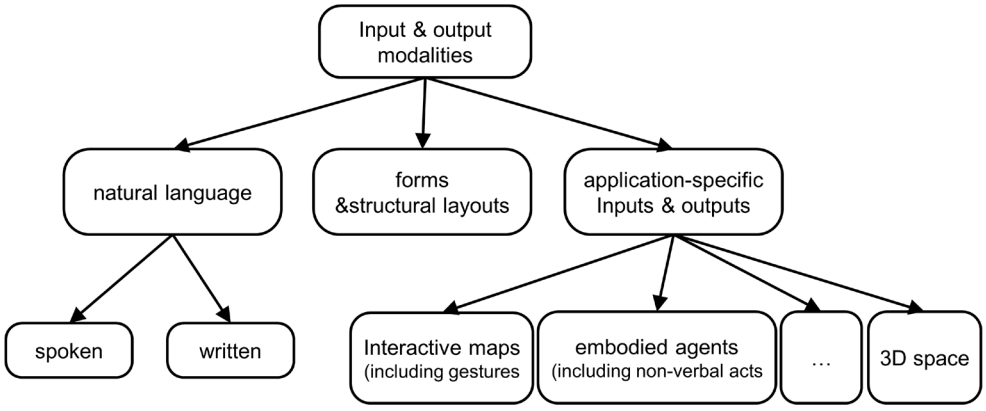


Figure 2.13: Categories of input and output modalities [202]

are the functional, architectural, or morphological adaptations that areneeded to achieve a synergetic behavior of the system as a whole. Natural languages are also good examples of compositional system of systems in the sense that sentences represent primitive systems and the language itself is a complexity (system of systems) formed by (a potentially infinite) set of sentences. System compositionality is interpreted as it is typical in the context of human natural languages where it guarantees that functionally, semantically and grammatically correct sentences are created from a set of basic words (verbs, nouns, adjectives, adverbs, etc.) using modifiers (affixes, suffixes, extensions, tokens, recursive syntactic rules, etc.).

In system science, composability and compositionality have been interpreted as different system development principles (SDPs). At the same time, they were often used interchangeably in the literature [209]. As an SDP and characteristics, composability means that the intended overall behavior of a system can be achieved in an aggregative manner and that the properties of the components do not change by virtue of f interactions with other components. As an alternative of the interpretation of compositionality, Tripakis (2016) argued that "it assumes that the overall behavior of a system is more than the summative operation of its components and that necessary interoperation of components may influence their manifestation" [74]. In computer science, compositionality is the principle of adapting system operation by composing and connecting system components together, and reasoning about the whole system [210].

Driven by the compositional paradigm, the requirements for smart behavior will be defined based on the overall objectives and role of the planned S-CPSs. While smartness cannot be reduced to an aggregation of the operational results of the components of S-CPSs, it needs a particular synthesis of various mechanisms such as context-based reasoning, goal-driven strategy development, functional adaptation and behavioral evolution that interplay in a synergistic manner to produce smartness. In each of the above examples, functional self-tuning and/or architectural self-adaptation are used by the system to achieve a synergistic high-level behavior [211]. The trustworthiness of a semi-intelligent manufacturing system is an example of the need for compositionality through manifestation of abstraction. Yu et al. (2017) interpreted and characterized trustworthiness as a composite paradigmatic feature that can be controlled and measured in terms of three metrics, namely, reliability, availability, and security, but only on system level [212]. Compositionality measures how much trustworthiness as system-level feature can be realized by local properties of the system components. In the field of system design, compositionality frameworks are used for system-level verification [213], system awareness [70], and schedulability [74], but not yet for implementing system-level reasoning as well as reasoning mechanisms for S-CPSs.

2.5.9 Issues of computational implementation of system level reasoning

An implementation of system-level reasoning can be constructed on multiple behavioral levels using analytic and synthetic computational approaches. The former is based on a combination of hardware devices and software application,whose computational function

is to generate conclusions from available knowledge using logical reasoning. The latter is based either on a single logical theory or on a composite logical theory, and/or a computational approach that tries to achieve a relatively high fidelity in comparison with human reasoning. According to the literature, these two approaches are normally used in different levels of abstraction.

An analytical approach offers the computation methods for reasoning about the system-level behaviors. In Dragomir et al. (2016), a compositional semantic and analysis framework is proposed for hierarchical block diagrams of a simulation model [214]. The framework provides a series of predicates and property transformers as semantics of composition in a series, in parallel incorporating the feedback of individual blocks. The approach aims at reducing the complexity of the real system to an abstraction model. For example, a compositional reasoning is proposed in [215] for model-based verification as part of designing embedded systems. incorporating the feedback of individual blocks. The approach aims at reducing the complexity of the real system to an abstraction model. For example, a compositional reasoning is proposed in [215] for model-based verification as part of designing embedded systems. This compositional reasoning applies a formal semantics to capture the features of the system components at a high level of abstraction. On the level of the system model, the reasoning should confirm that the system and its application models have the same behaviors with respect to the considered properties. However, it is a limitation of this approach that it does not include computational models for reasoning about how smart systems are to operate. That is the limitation appears on the control side of the system. It is also important to mention that abstraction is captured in formal or computational models through coding processes in the domains of software engineering, computer science, and AI practices [216], [217].

In a synthetic computation approach, the term synthetic means that the reasoning architecture, processes, and results are produced by computational synthesis. This approach is usually applied in the field of cognitive robots, context-aware systems, and self-adaptive systems by means of AI-based [218], Machine learning [219], and cognitive architecture [220]. For example, Memory-Attention-Composition (MAC) framework [125] is an end-to-end differentiable architecture to perform a multi-step reasoning process. To solve a problem, the model is decomposed into a series of inferred reasoning steps associated with computational units. In [221], the framework is proposed for computational cognitive affordances. The cognitive cycle consists of two parts, namely logical-based representation and a computational architecture that performs a synthetic reasoning, Action-Planning-Reasoning-Sense-Making tasks. The abovementioned approaches do not address the compositionality issue explicitly. An attempt to improve compositionality in CPSs was found in [35]. Several structures of component composition for reliability and durationare illustrated. The composition rules are formulated. These rules confirm compositionality at component level, but an achievement of system-level compositionality cannot be guaranteed. It assumes that if the entire systems are manifested by the composition rules, system-level properties can be achieved.

2.5.10 Overview of the major findings and their implications

The finding confirmed that smartness is not only a collective property of a system, but it is also a holistic and synergistic behavioral characteristic. The orchestration of synergetic interoperation of reasoning methods goes beyond condition-based composition. It should utilize the complementary and strengthening effects of reasoning methods. Designing of a compositional reasoning mechanism requires comprehensive means for supporting the entire design process. This is the expectation that S-CPSs should be able to select and handle knowledge synthesis mechanisms that operate with heterogeneous and/or incomplete knowledge. As far as the applied knowledge representation is concerned, it seems to be necessary that they are equipped with multiple knowledge representation means (in order to be able to cope with the challenges of possible representational variety). Knowledge should be constructed with a wide range of formalisms (i.e., from generic domain knowledge to specific task knowledge).

System awareness is a fundamental ability of S-CPS from the point of view of realization of the overall smart behavior of the system. This ability enables systems to control their performance and operation, and to interact with their embedding environment purposefully. Awareness is built by syntactic and semantic processing of data obtained from a range of hardware and software sensors. Designing for system awareness also requires computational data fusion technologies and models, and various inference mechanisms for transforming data to information and knowledge.

The design process of decision-making mechanisms needs to consider: (i) when a decision can be made by the system based on acquired and inferred knowledge, (ii) what methods of decision making are the most suited for the problem and the knowledge at hand, (iii) how to verify the decisions with regards to the objectives of the system, and (iv) how to evaluate and learn from the consequences of the decisions. Another challenge is designing systems for runtime adaptations. System adaptation goes together with the need to develop strategies for generating alternative operation modes for the system. It requires computational mechanisms (i) to transformthe changing system objectives into feasible action plans, (ii) to decide on the operationalization and timing of the chosen action plan, and (iii) to execute the adaptation in a fully controlled manner.

Designing of reasoning mechanisms covers (i) the selection of the modality of reasoning (i.e., deductive, inductive, abductive) that is the most suited for building awareness, making decisions and adaptation of the system, (ii) composition of reasoning methods, (iii) design of compositional reasoning workflow, (iv) interfacing the elements of the reasoning mechanism for a seamless interoperability, and (v) verification of compositional framework of reasoning. Despite the facts that some of the computational reasoning mechanisms are able to imitate some aspects of human like reasoning, most of them remain data driven and operate according to statistical and/or rule-based methods. While computers are strong in processing of, and making decisions based on, large amount of data and predefined rules, they are currently weak in reasoning with analogies and intuitive inferencing. Efforts, on

the other hand, are already visible in the state-of-the-art literature that aim to mimic human like reasoning and extend the existing approaches with human like capabilities such as intuitive belief network generation.

Towards a development of system-level reasoning, the recommendation generation is considered as the decision support service. The recommender systems recently have been used in the domains of engineering systems (e.g., product design support systems, medical treatment systems, monitoring and warning system, and traffic management systems). The concept of their recommender engines, such as the probabilistic inference, ontology-based reasoning and the ML-based reasoning, are designed to offering personalized recommendations to individual users. According to the various methodological approaches concerning the system-level operation of recommender systems, the traditional filtering techniques are not applicable to context-sensitive process of designing reasoning mechanisms. It needs the interaction with a designer to capture the know-ledge and context information about the design process. The recommendation generation should consider: (i) changes in preferences of users and in the features of recommendation items, (ii) changes in context information concerning the procedural aspects of reasoning mechanism, (iii) proper knowledge representation and reasoning, which potentially handle the changes in the design process, (iv) responsiveness to the changes in runtime, and (v) proper communication modalities which can actively interact with a designer.

2.6 Exploration of requirements for an active recommender framework

2.6.1 The idea of active recommender frameworks

Active recommender framework (ARF) is a conceptual idea that allows 'the framework' facilitating a designer to develop smart systems by monitoring the activity of a user, reasoning and understanding what is being attempted, and then proactively providing smart assistance during execution of the design session for a development of a reasoning mechanism. This concept seems to be new in the contemporary literature, which typically focuses on providing recommendation, task-dependent advising, and support of decision making in non-engineering contexts. They can potentially be applied to offer recommendation services to a designer, but less content- and contextsensitive manner than as can be expected from a specialized ARF system. The specific objective of the ARF is to facilitate the development of a smart reasoning mechanism(integral sets of smart algorithms) by recognizing what is being attempted by the designers and by monitoring their activities, reasoning and performance, and providing context-sensitive recommendation in design sessions proactively. Recommendation includes not only design problem solving information, but also selecting the suitable computational methods and making a proper decision.

To facilitate an achievement of the system-level reasoning in smart RM, the realization of the ARF assumes a comprehensive meta-model (MM), which brings

the foundational concepts into synergy from a computational point of view. This MM can capture and integrate epistemological entities (including chunks of knowledge, modelling constructs, sets of constraints, support services, etc.) as well as the procedural and methodological elements (overall approach, support actions, computational steps, communications, etc.). With regard to its representation, this MM is more of an intuitive cognitive model, rather than a formal information model. This is the reason why it is not specified exactly from an information engineering viewpoint, but remains in the background of the ideation.

2.6.2 Identification of types of requirements

A requirement is defined as a statement that reflects a need, expectation, and/or constraints. The principle of requirement engineering was applied to identify the requirements for the development of ARF. The purpose of requirement engineering is to define the needs for the systems with a set of clear and complete statements and to ensure that the implementable functionalities are correct, reasonable, and effective. Generally, the requirements are classified into two main types: functional requirements and non-functional requirements.

The functional requirements imply the expectation about what the ARF should be manifested in and not about how should it be implemented. The latter is the topic of implementation requirements. Non-functional requirements describe the general properties of the ARF. Several aspects of non-functional requirements can be considered, for instance, (i) performance (e.g., processing time, percentage of accuracy, levels of system stability), (ii) interoperation (standards) requirements (e.g., be compatible with ISO 30401:2018 standard), and (iii) business requirements (e.g., increased revenue, reduced expenses, improved customer satisfaction). With a view to the proposed idea of the ARF, to the implications of the findings of the completed literature study, and to the outcome of the brainstorming within the research group, the non-functional requirements are sorted according to five aspects. These are: (i) structural requirements: (ii) computational requirements; (iii) knowledge management requirements; (iv) interaction requirements; and (v) application-oriented| requirements.

The evidential goal of exploration of requirements is to provide orientation, guidance, and factual expectations for conceptualization of the ARF. This is a complex task, which needs systematization. With this in mind, the exploration of functional requirements was done on four levels, which reproduces the overall decomposition structure of the ARF, namely, (i) framework level (which reflects the characteristics of the whole of the ARF); (ii) mechanism level (which expresses the expectations for the manifestation of the computational mechanisms); (iii) module level (which concerns the lower-level components of the computational mechanisms); and (iv) algorithm level (which deals with the individual algorithms). In this chapter, the exploration of the requirements for the first two levels is discussed. The requirements for the other two levels would be explored in the next chapter, dominantly in the context of implementation of the components and algorithms. It was done by associating the detailed descriptions with the application context. The description of the target application would be elaborated in the sub-section

3.3.5 of the next chapter.

2.6.3 Identification of requirements for system-level framework

Functional requirements

As mentioned in the idea of an ARF, it is supposed to be a context-sensitive recommender system which supports a designer during an execution of the design process of ASRMs. Process monitoring and decision support capabilities are proposed as the fundamental characteristics of the ARF. Hence, the two essentials of functional requirements (FRs) of the whole ARF are:

FR-F01: The ARF should support designers interactively (adapting its operation to the pace of human interaction) in the development process of ASRMs.

FR-F02: The recommendation generated by the ARF should rely on the information actually processed by the designer and the (dynamically changing) contexts associated with the design processes.

Structural requirements

Structural requirements determine the final quality and justify the design decisions that constrain the implementation of functional requirements. The requirements are indicated by quality factors that should be achieved in the implementation of systems,for instance, efficiency, portability, safety, robustness, and maintainability. To deal with the complexity of the ARF, the goal is how to realize the proposed functionality with the lowest possible complexity of architectural structure. Concerning the systemlevel architecture of the ARF, the structural requirements focus on the aspects of implementation simplicity and adaptability.

SR-F01: The system-level architecture of the ARF should be constructed of the lowest possible number of mechanisms.

SR-F02: Within the system-level architecture, the mechanisms should induce the lowest possible number of functional, computational, and control relationships and dependencies with each other.

SR-F03: The structural adaptation of the whole ARF should be done with the lowest possible efforts in terms of the number of adapted constituents

SR-F03: The structural adaptation of the whole ARF should be facilitated by the lowest possible number of modifications of the algorithms.

Computational requirements

Computational requirements are the expected specifications of computational capabilities of an ARF. At the system-level operation, the overall service level should be taken into consideration in order to satisfy the expectation of an end user. From the developer perspective, the availability of resources should be taken into consideration. The

computation mechanism of whole ARF should be testable in a demonstrative case under the resource constraints.

CR-F01: The computational mechanisms of the whole ARF should generate recommendations for the designers in less than 10 seconds in order to keep the designer's attention on the design task.

CR-F02: The (computational) adaptation of the whole of the ARF should be facilitated by the largest possible number of standardized or pre-programmed constituents.

Knowledge management requirements

Knowledge management requirements refer to the needs and/or expectations of the ways to deal with knowledge, for instance capturing, organizing, classifying, reasoning, and storing knowledges. For system-level requirements, we expect the ARF should have a capability to learn. It implies that the system-level knowledge will be growing when the ARF has been utilized by designers. Two possible ways can be considered to extend the knowledge: (i) capturing knowledge from designers, (ii) updating additional knowledge by experts.

KR-F01: The system-level knowledge of the ARF should be extendable internally (by the ARF) or externally (by the knowledge engineers) as needed by the design processes and the required support.

KR-F02: The information/knowledge conveyed by the recommendation should be constructed in both machine-readable and human-understandable forms.

Interaction requirements

We considered the interaction requirements from two aspects: (i) the interactionbetween the ARF and the designer, and (ii) the interaction between the components of the ARF. In the process of recommendation generation, the ARF requires information from the designer. However, too many interactions requested by the ARF might not be supportive to the creative work of the designer and may attract the designer's intention away from the actual task. At the system level, the mechanisms communicate with each other by exchanging situation-dependent data and/or information. In addition, the ARF navigates the designer to the webpage that contains the advisory contents to execute the design action. Hence, the ARF should operate online. Three interaction requirements are addressed:

IR-F01: The communications between the ARF and the designer should be processed with lowest possible number of interactions.

IR-F02: The flow of information should be timely harmonized as requested by the mechanisms in near zero-time operation.

IR-F03: The ARF should connect to the internet during the generation of advisory contents.

Application oriented requirements

To develop ASRMs, the application context should be identified. Basically, it shouldrepresent the characteristics of S-CPSs which can be used to test the proposed concept of ARF. Many

practical cases belonging to the family of S-CPS can be supported by an ARF. We selected an automatic parking assist system (APAS) and used it as a demonstrative case for exploring application-oriented requirements.

Two system-level requirements for the ARF are as follows.

AR-F01: The ARF should provide the specialized recommendation services to support a development of reasoning mechanism for an APAS.

AR-F02: The ARF should offer content-related recommendations of searching a proper parking plan for a street parking problem.

2.6.4 Identification of requirements for mechanism level

Functional requirements

FR-M01: The process monitoring mechanism should continuously perform activity-based monitoring throughout the entire processes of RMD at runtime during the design session.

FR-M02: The process monitoring mechanism should recognize doubtful (unexpected) events in the design sessions in quasi-real time.

FR-M03: The process monitoring mechanism should identify an obstacle in the actual flow of design actions based on the procedural network with higher than 45% of justified objective decisions.

FR-M04: The decision support mechanism should capture at least three relationships of design actions included the logical, temporal, and methodological relationships.

FR-M05: The decision support mechanism should computationally model design actions and interconnect into a procedural network with lower than 5% of incorrect relationships.

FR-M06: The decision support mechanism should propose a feasible recommendation in human-understandable format.

Structural requirements

SR-M01: All computational mechanisms should consist of the lowest possible number of architecting modules.

SR-M02: All interrelationships among modules should be created based on the lowest possible number of dependencies.

SR-M03: The structural adaptation of mechanisms should be done with the lowest possible efforts when modification of algorithms is needed.

Computational requirements

CR-M01: All computational mechanisms are supposed to be operationalized by using minimum computational resources.

CR-M02: The process monitoring mechanism should be able to implement near zero-time processing.

CR-M03: The decision support mechanism should generate a recom mendation for the designer in less than 10 seconds.

Knowledge management requirements

KR-M01: The knowledge belonging to the decision support mechanism should be shared and jointly utilized by multiple modules

KR-M02: For the decision support mechanism, formal knowledge should be constructed either in a model-based or in a sematic-based format.

Interaction requirements

IR-M01: The flow of information throughout the mechanisms should be harmonized in time as requested by the modules with the least possible number of interactions.

IR-M02: The process monitoring mechanism should communicate directly with the designer using the lowest possible number of interactions.

2.7 Assessment of the requirements for the active recommender framework

2.7.1 Approach to assessing of requirements

Before operationalization in conceptualization and design, the collected set of requirements has to be assessed with regard to their demands, implications, and characteristics. The latter includes properties such as (i) consistency requirements are supposed to be not contradictory), (ii) completeness (all relevant requirements should be included in the requirements model), (iii) feasibility (the demands conveyed by requirements should be feasible both technically and economically), (iv) understandability (the description of the requirements should fulfill quality standards), and (v) reusability (the requirements are supposed to be reusable in other contexts and future projects). These are to be considered when assessing if the above-discussed requirements are relevant and do not conflict with each other. The different aspects of assessment imply the need for using different techniques. To assess the consistency, the complete body of requirements should be examined.

The individual requirements can be assessed from a theoretical point of view, or from a practical point of view, or from both concurrently. Typically, the former assessment is based on analysis of documents. In our work, the implications of the major findings of the literature study have been used as the basis of deriving requirements, but this does not guarantee the fulfillment of the expectations for proper requirements. The practical

assessment can be completed by testing the implications in an application context. Actually, such assessment can be done at syntactic level, i.e., by checking for compliance with standards and guidelines. However, in the case of assessment of the system-level and the mechanism-level requirements, the consistency resides at a semantic level (i.e., in the practical meaning). To facilitate the exploration of semantic relationships, we apply a formal representation to the set of requirements. The representation in the form of a semantic net facilitates not only checking the contradictions of requirements, but also the interpretation of the requirement relationships with regard to their technical feasibility.

2.7.2 Assessment of system-level requirements

Fourteen system-level requirements were specified for the ARF. Their sematic relationships of the requirements have been identified, as shown in Figure 2.14. We found that ten of the system-level requirements can be met technically (e.g., their expectations can be satisfied at the conceptualization of the ARF). The direction of the arrow indicates the causality of the requirements. For example, the requirement IRF02 has an influence on the requirements FR-F01. It means that if the ARF would interact with a designer in real-time, it should communicate with him by the lowest number of interactions. In addition, it is possible that a requirement has multiple relationships to others, and vise-versa. At the system-level, it should ensure that two functional requirements are fulfilled, including: (i) FR-F01 – the ARF should interact with the designer in real-time; and (ii) FR-F02 – the ARF should provide contextsensitive recommendation at runtime.

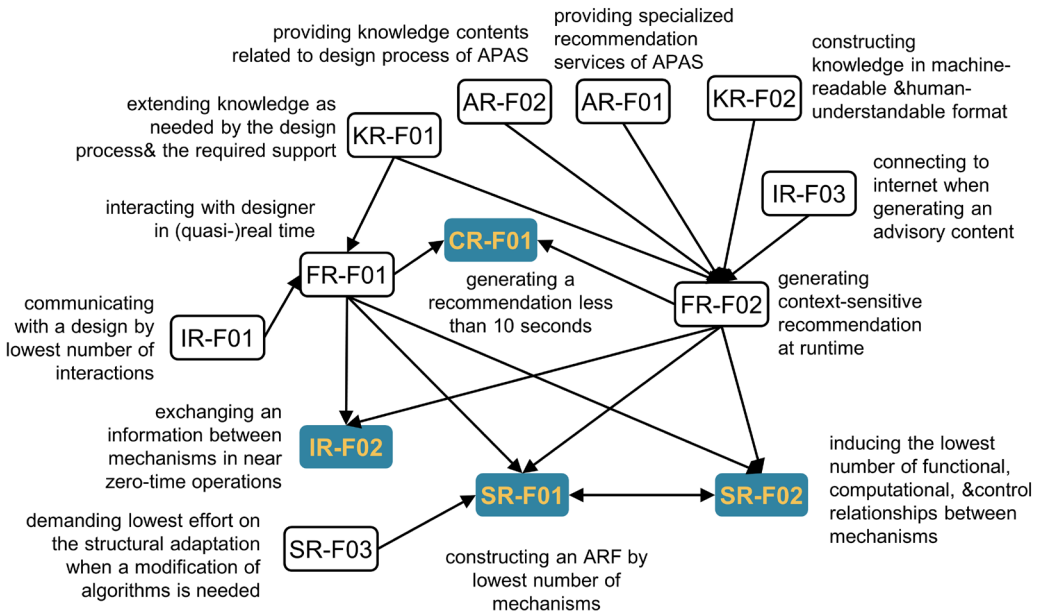


Figure 2.14: Semantic relationships of the system-level requirements

According to their relationships in the semantic diagram, they have influence on four non-functional requirements (see the color boxes in Figure 2.14), namely (i) IR-F02 – the information exchange should happen in a near zero-time operation, (ii) CR-F01 – the processing time of recommendation generating should be no longer than 10 seconds, (iii) SR-F01 – the system-level architecture of the ARF should be constructed of the lowest possible number of mechanisms, and (iv) SR-F02 – the mechanisms should induce the lowest possible number of functional, computational, and control relationships and dependencies with each other. All of them are presented at the terminal nodes. They have no influences on other requirements. Therefore, it could be claimed that if these requirements are fulfilled, all system-level requirements would be satisfied.

However, three sets of requirements seem to be in conflict with each other, as shown in Figure 2.15. They cannot be satisfied simultaneously. Therefore, a compromise should be found in terms of their goal (expectations). In the case of a particular set of requirements, SR-F03 – providing structural adaptation with the lowest possible effort and SR-F04 – allowing modification of algorithms with the lowest possible effort, it is difficult, or even impossible, to meet the requirement CR-F02 – maintaining the largest number of standard preprogrammed algorithms as well as to meet the requirement KR-F01 (extending the system-level knowledge as needed by the design processes and the required support). In the meantime, the requirements KR-F01 and IR-F01 (communicatingwith the designer by lowest possible number of interactions) are inconsistent. Two of them have direct relationships with the functional requirements FR-F01. This implies an opportunistic solution with multi-objective decision-making.

Considering the requirement KR-F01, it is the expectation for the scalability of the ARF which implies its learning capability. If there is the need for new knowledge as requested by either the design processes or the service support, the system knowledge of the ARF should be extendable. The knowledge can be extended by two ways: (i) the external modification – the knowledge engineer updates the new knowledge; and (ii) the internal modification – the ARF obtains context information from the designer and converts them into the system knowledge. The latter requires the increasing number of interactions with the designer and the adaptation of the architectural structure to perform this process.Thus, it is impossible to concurrently meet these requirements. To deal with this challenge and to avoid the uncontrollable complexity in the conceptualization,

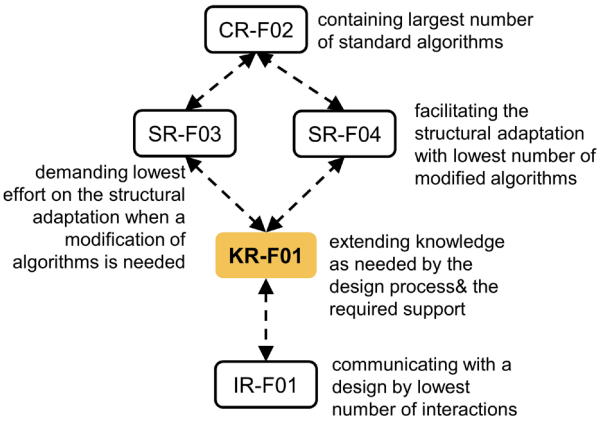


Figure 2.15: Trade-off issue in the case of inconsistent system-level requirements

implementation, and validation of the ARF, we assume that the knowledge is added by the knowledge engineers with minimum efforts for the functional, structural, and computational modifications, when it is needed.

To this end, the relationships of the requirements can be simplified as shown in Figure 2.16. Four key requirements should be tested to validate the system-level functionality of the whole ARF as shown in the green boxes. It can be assumed if these requirements are satisfied, the system-level functionality will be achieved. All the rest of requirements are satisfied without the needs of testing.

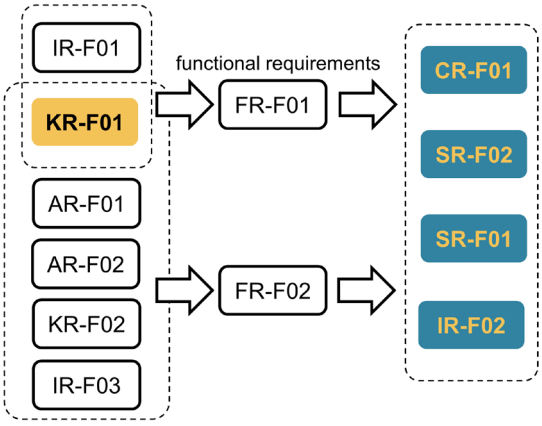


Figure 2.16: Simplified relationships of system-level requirements

2.7.3 Assessment of the mechanism-level requirements

There are eighteen identified requirements in total at the mechanism level. As shown in Figure 2.17, the semantic map is constructed by the relationships of sixteen requirements. Five requirements are presented at the terminate nodes including: (i) FRM03 – the process monitoring mechanism should identify an obstacle with a higher than 45% of justified objective decisions, (ii) SR-M01 – all computational mechanisms should consist of the lowest possible number of architecting modules, (iii) IR-M01 – the flow of information throughout the mechanisms should be harmonized in time as requested by the modules with the least possible number of interactions, (iv) CR-M02 – the process monitoring mechanism should be able to implement near zero-time processing, and (v) CR-M03 – the decision support mechanism should generate a recommendation for the designer in less than 10 seconds. Based on the interpretation of the semantic relationships, it assumes that if these requirements are satisfied, the expectations of the mechanism-level functionality will be achieved.

However, it seems that two groups of inconsistent requirements occurred. One of them is the group formed by the requirements FR-M03 and IR-M02 as shown in Figure 2.18 The

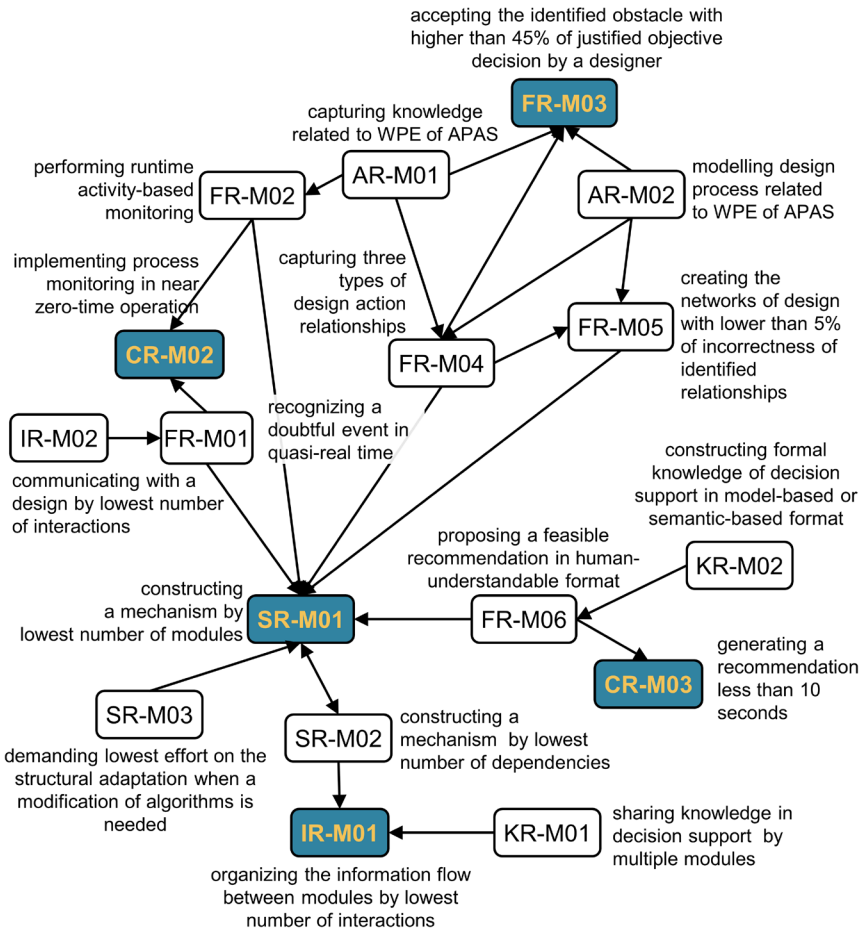


Figure 2.17: Semantic relationships of the mechanism-level requirements

justified objective decision on the acceptance of the identified obstacle may not reach a higher than 45%, if the lowest number of interactions between the ARF and the designer is happening. It might be the case that the ARF requests the further information for an investigation of the actual design flow.

According to the semantic map, the FR-M03 is considered as the key functional requirement, thus the requirement IR-M02 should be adjustable. Just as with the inconsistency of the requirements CR-M01 and CR-M02, they might not be concurrently satisfied. In some cases, the operationalization of the process monitoring in near zero-time consumes a huge number of computational resources. To meet the key requirement CR-M02, the CR-M01 should be compromised. Based on this analysis, five key requirements should be tested to ensure that the conceptualization of the ARF meet the expectation at the mechanism level as shown in the green boxes in Figure 2.19. Two requirements should be compromised

including IR-M02 and CR-M01 in order to meet all requirements at the mechanism level.

2.8 Conclusions

As discussed by many researchers, the paradigm of cyber-physical systems is rapidly evolving, and the domains of investigations, implementations, and applications are proliferating quickly. This is the reason why thinking in generations of CPSs was proposed in [222]. It can be seen that while CPSs are showing more 'system intellect' in their operation, their control regime must be more sophisticated, and they should be equipped with many self-* characteristics.

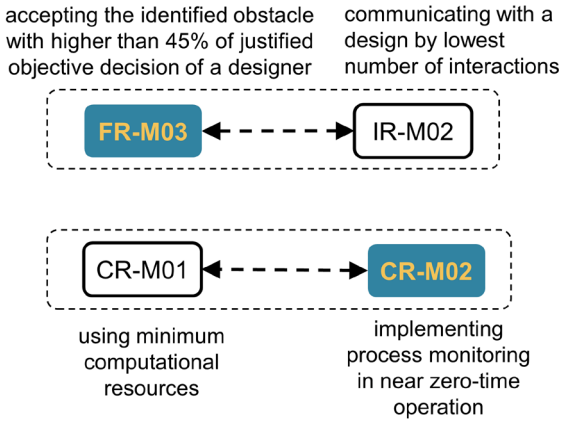


Figure 2.18: Trade-off issue in the case of inconsistent mechanism-level requirements

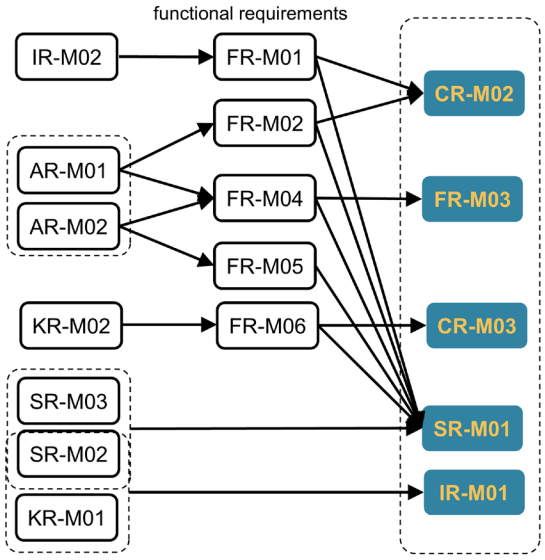


Figure 2.19: Simplified relationships of mechanism-level requirements

S-CPSs present many system-level operational characteristics, as opposed to the component operation driven aggregative manifestation of system characteristics. They go beyond what can be analyzed and designed based solely on reductionism and the traditional model-based approach. These statements are becoming our research challenge regarding how to develop smart CPSs with the capabilities of self-awareness and self-adaptation. Compositional conceptualization and design of S-CPSs need new principles (e.g., system level synergy) and a different (top-down specification) approach. The study was completed by using mixed qualitative and quantitative methods. The publications related to CPSs and system smartness represented the broader and the narrower contexts of the study. The domain of discourse included the domain of system engineering frameworks in the contexts of designing system-level reasoning and its enablers. The major requirements were explored concerning the development of a novel ARF. The knowledge aggregation was concludedas follows:

2.8.1 Conclusions concerning reasoning mechanism development

Reasoning about emerging conditions and their effect on system performance creates a complexity that cannot be tackled by predefined reasoning methods. This complexity is caused not only by demands for real-time computational requirements or by the need to cope with incomplete information, but also by the problem of finding optimal reasoning and adaptation strategies matching the nature of the emerging situation. It requires runtime composition of reasoning strategies and adaptive use of reasoning methods. The challenge for designers of a reasoning mechanism is to narrow down the solution space of composition of reasoning mechanisms that provide synergetic operation of S-CPS. The conclusions concerning RMD are:

- It is difficult, if not possible, to apply a single reasoning method to tackle complex reasoning problems that S-CPSs are typically facing. As S-CPSs operate under unpredictable, emerging conditions, their ability to runtime adapt to changing conditions in a safe and predictable way is essential for their robust operation.
- Synthetic computational approaches have the ability to compose reasoning methods at runtime. They, however, implement a low-level smartness by straight-forward composition of methods that are only activated if given conditions are fulfilled. Without a rigorous unifying framework, synthesis reasoning and an integration of the analysis results based on analytical computational approaches remain ad hoc.
- Compositionality regarding reasoning mechanisms manifests in different levels of abstraction that are: (i) on the system level, it achieves a synergy of knowledge through the entire reasoning processes that is needed for multitask problem solving; and (ii) on the component level, systemcomponents should be interoperated in compositional manner.
- This requires a multi-aspect framework that can integrate system-level reasoning on various abstraction levels ranging from defining system objectives to concreate implementation of adaptation at runtime.

2.8.2 Conclusions concerning active recommender framework development

Typical SEFs for specifying manifestations of non-compositional systems can be clustered as: (i) generic, (ii) conceptual, (iii) logical, (iv) architectural, (v) functional, (vi) component-based, (vii) model-based, (viii) temporal, (ix) contextual, and (xi) composite frameworks. Though many SEFs have been developed for composable systems, they cannot be transferred directly to compositional systems. 2G-CPSs assume SEFs that facilitate synergistic definition, synthesis, and adaptation of all system components. Besides smart reasoning, the target SEFs should support the implementation of other system- level features of S-CPSs such as dependability, security, and openness. Within the framework, we scope our attention to the reasoning part of S-CPSs, which is able to create system-level smartness. It consists of the compositional tasks of in S-CPSs that provide logical computational processes including creating semantic knowledge structures, inferring within reasoningmechanisms, building situation awareness, enabling dependent decision-making, and performing system adaptation. The conclusions concerning the content development of the ARF are as follows:

- The results of the literature study showed that no SEF was developed so far, which would cover all enablers of system level-reasoning (i.e., system knowledge, situation awareness, context sensitive reasoning, decision-making, and system adaptation). There are two possible trajectories for the development of such a framework. One is the development of a new concept from scratch. Another one is to integrate the relevant parts of existing frameworks into one holistic framework. However, in the latter case, the functionally relevant parts may not be simply interconnected. These parts should be reconsidered and interpreted on a higher level of abstraction, and that way they may be integrated into a novel designsupport framework.
- Due to the complexity of system-level reasoning, the ARF cannot be a singleaspect, monolithic and universal arrangements of interrelated things, but instead should be multi-aspect composite structures (constructs) that show a wide variety depending on the purpose of operations and the application contexts.
- The existing SEFs are operationalized in a static manner. Most of them cannot capture the changes in the system behaviors, thus they cannot provide the solutions at runtime operation. We concluded that the traditional SEFs cannot support a development of smart-reasoning mechanisms effectively.
- Two system-level capabilities are needed to equip a framework with an active manner: (i) real-time process monitoring, and (ii) runtime decision support. Therefore, the essence of the ARF is the combination of the process monitoring functionality with context aware recommendation services to support the development of reasoning mechanisms focusing on the enables of system-level reasoning.
- A synergistic fusion of the procedural reasoning process (e.g., event recognition, building awareness, context-based reasoning, decision-making, and functional and adaptation planning) of S-CPSs is needed for a development of ASRMs.

- Considering the computational algorithms included in ARSMs as knowledge elements, the ARF should handle these knowledge elements and realize their design process and tasks. Without this domain specific knowledge, the ARF cannot recognize the actual state of design process. Therefore, in the lack of sufficient knowledge, the possibility that the ARF will propose an incorrect recommendation to the designer may be large.
- The ARF should facilitate the system designer with the following services: (i) recognizing the changes in the design process, (ii) communicating with a designer during the execution of the design process, (iii) providing the design guideline at the higher level of abstraction, (iv) guiding how to select the right computational method with proper design action at lower-level of design operation, (v) providing an example of the best coupled design actions, (vi) comparing alternatives for the integration of multiple methods, and (vii) giving a recommendation about the feasible solutions varying on the application contexts.

2.8.3 Conclusions concerning the requirements for an active recommender framework development

The exploration of requirements for the ARF had been done in two levels: (i) systemlevel, and (ii) mechanism-level. The individual requirements were derived from the implications of the findings of the literature study. However, we observed that some of them had relationships to each other. It required an analysis of the semantic meanings of their relationships in the whole. Therefore, semantic maps were constructed to investigate the relationships of the requirements on different levels of abstraction. From the point of view of the development process of the ARF, the conclusions concerning the requirements are as follows:

- Two functional system-level requirements are related to: (i) process monitoring; and (ii) decision support functionality.
- It is feasible to test the system-level functionality of the ARF with four key system-level requirements: (i) the construction of system-level framework should be done with the lowest possible number of mechanisms, (ii) the mechanisms should induce the lowest possible number of functional, computational, and control relationships and dependencies with each other, (iii) the information exchange should happen in a near zero-time operation, and (iv) the processing time of recommendation generating should be no longer than 10 seconds.
- Regarding the mechanism-level requirements, five of them were considered as the key requirements: (i) the process monitoring mechanism should identify an obstacle with a higher than 45% of justified objective decisions, (ii) all computational mechanisms should consist of the lowest possible number of architecting modules, (iii) the flow of information throughout the mechanisms should be harmonized in time as requested by the modules with the least possible number of interactions, (iv) the process monitoring mechanism should be able to implement near zero-time processing, and (v) the decision support mechanism should generate a recommendation for the designer in less than 10 seconds.

The abovementioned requirements described the conceptual idea of the ARF, as follows:

- The findings in Section 2.4.5 imply that two system-level functionalities are needed to equip the framework with an active design support capability: (i) realtime process monitoring, and (ii) run time recommendation provision.The latter is integral part of decision support. From the system-level requirement.SR-F01 follows that the ARF should be constructed with the lowest possible number of architectural elements. This optimum can be achieved in two possible ways: (i) putting all elements together in one mechanism, and (ii) mapping one service functionality to one mechanism.
- At the mechanism-level, five functional requirements have influences on the architectural modules. Thus, if a function is mapped to a module in a one-to-one manner, then five modules are formed: (i) activity-based monitoring, (ii) unexpected event recognition, (iii) design actions and their relationship identification, (iv) a procedural network of design action construction, and (v) recommendation generation.
- With this number of mechanisms and modules, it is possible to keep the lowest number of functional, computational, and control dependencies between and within the mechanisms and to exchange the information flow with near zero operation time.
- With a view to keeping the designer's attention on the design task, it can be conceived that the computational process of recommendation generation can be done within 10 seconds.
- The last requirement that should be considered is the performance of the process monitoring mechanism and the decision made by the designer. If the ARF recognizes an unexpected event and identifies the obstacles based on the procedural network of design actions with a high percentage of correctness, and the designer accepts the offer with a high number of probabilities, this implies the quality of procedural network and the improvement opportunities for the recommendation generation.

References

[1] Van Eck, N. J., & Waltman, L. (2010). Software survey: VOSviewer, a computer program for bibliometric mapping. *Scientometrics*, *84*(2), 523–538.

[2] Esterle, L. & Grosu, R. (2016). Cyber-physical systems: challenge of the 21st century. *Ei Elektrotechnik Und Informationstechnik, 133*(7), 299–303.

[3] Broy, M., Cengarle, M. V., & Geisberger, E. (2012). Cyber-Physical Systems: Imminent Challenges. in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 1–28).

[4] Pourtalebi, S., Horváth, I., & Opiyo, E. (2013). Multi-aspect study of mass customization in the context of cyber-physical consumer durables. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 55911, p. V004T05A006). American Society of Mechanical Engineers.

[5] Krupitzer, C., Roth, F. M., Van Syckel, S., Schiele, G., &Becker, C. (2015). A survey on engineering approaches for self-adaptive systems. *Pervasive and mobile*

computing, 17, 184–206.
[6] Poovendran, R., (2010). Cyber-physical systems: Close encounters between two parallel worlds. In *Proceedings of the IEEE*, (pp. 1363–1366).
[7] Díaz, J., Pérez, J., Pérez, J., & Garbajosa, J. (2016). Conceptualizing a framework for cyber-physical systems of systems development and deployment. In *Proceedings of the 10th European Conference on Software Architecture Workshops - ECSAW '16* (pp.1-7).
[8] Smirnov, A., Levashova, T., Shilov, N., & Sandkuhl, K. (2014). Ontology for cyberphysical-social systems self-organization. In *Conference of Open Innovation Association, FRUCT* (pp. 101-107).
[9] Salehie, M., & Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transaction on Autonomous and Adaptive System, 4*(2), 1-42, 2009.
[10] Lee, K., & Chirikjian, G. S. (2007). Robotic self-replication. *IEEE Robotics & automation magazine, 14*(4), 34-43.
[11] Wang, S., Zhang, C., & Li, D. (2016). A big data centric integrated framework and typical system configurations for smart factory. In *Industrial IoT Technologies and Applications* (pp. 12-23). Springe International Publishing AG, Cambridge.
[12] Dumitrache, L. (2011). Cyber-physical systems - new challenges for science and technology. *Journal of control engineering and applied informatics*, 3(*3*), 3-4.
[13] Salerno, J., Hinman, M., & Boulware, D. (2004). Building a framework for situation awareness. In *Proceedings of the Seventh International Conference on Information Fusion* (pp. 219–226).
[14] Sharma, A. B., Ivančić, F., NiculescuMizil, A., Chen, H., & Jiang, G. (2014).Modeling and analytics for cyber-physical systems in the age of big data. *ACM SIGMETRICS Performance Evaluation Review, 41*(4), 74–77.
[15] Sene, A., Kamsu-Foguem, B., & Rumeau, P. (2015). Telemedicine framework using case-based reasoning with evidences. *Computer Methods and Programs in Biomedicine, 121*(1), 21–35.
[16] Leitão, P., Colombo, A. W., & Karnouskos, S. (2016). Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computer in Industry*, 81, 11–25.
[17] Engell, S. (2014). Cyber-physical systems of systems - Definition and core research and innovation areas. Retrieve from https://www.cpsos.eu/wp-contentuploads/2015/07/CPSoS-Scope-paper-vOct-26-2014.pdf.
[18] Macías-Escrivá, F. D., Haber, R., del Toro, R., & Hernandez, V., (2016). Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications., 40*(18), 7267–7279.
[19] Mainzer, K. (2015). The emergence of self-conscious systems: From symbolic AI to embodied robotics. In *Philosophy, Computing and Information Science* (pp. 57–66). Taylor and Francis.
[20] Metzler, T. & Shea, K. (2010). Cognitive products: definition and framework. In *Design* (pp. 865–874).
[21] Håkansson, A., Hartung, R., & Moradian, E. (2015). Reasoning strategies in smart

cyber-physical systems. *Procedia Computer Science., 60*, 1575–1584.
[22] Mostéfaoui, S. K., & Hirsbrunner, B. (2003). Towards a context-based service composition framework. In *Proceedings of the International Conference on Web Services* (pp. 42–45).
[23] Raducanu, B., & Vitrià, J. (2008). Learning to learn: From smart machines to intelligent machines. *Pattern Recognition Letters, 29*(8), 1024–1032.
[24] Horváth, I. (2021). Connectors of smart design and smart systems. *AI EDAM, 35*(2), 132–150.
[25] Saarinen E., & Hämäläinen, R. P. (2010). The originality of systems intelligence. *Essays on Systems Intelligence*., 9–28.
[26] Daniel, F., Matera, M., & Pozzi, G. (2008). Managing runtime adaptivity through active rules: the bellerofonte framework, *Journal of Web Engineering, 7*(3), 179–199.
[27] Daun, M., Brings, J., Bandyszak, T., Bohn, P., & Weyer, T. (2015). Collaborating multiple system instances of smart cyber-physical systems:a problem situation, solution idea, and remaining research challenges. In *the 1st International Workshop on Software Engineering for Smart Cyber-Physical Systems* (pp. 48–51). IEEE/ACM.
[28] Zhang X. Y., & Le Zhou, C. (2013). From biological consciousness to machine consciousness: an approach to make smarter machines. *International Journal of Automation and Computing, 10*(6), 498–505.
[29] Scheidl, D.I. R. (2016). Actuators and sensors for smart systems. In *Proceedings of the 10th International Fluid Power Conference* (pp. 367–384).
[30] Lanting C., & Lionetto, A. (2015). Smart systems and cyber physical systems paradigms in an IoT and Industrie/y4.0 context. In *Proceedings of 2nd International Electronic Conference on Sensors and Applications*.
[31] Okon, S., & Asagba, P. (2014). Self-organization and self-healing: Rationale and strategies for designing and developing a dependable software system. *International Journal of Innovative Research in Computer and Communication Engineering, 2*(4), 3687–3698.
[32] Wang, Y. (2009). On cognitive computing. *International Journal of Software Science and Computation Intelligence, 3*(1), 1–15.
[33] Zhuge, H. (2011). Semantic linking through spaces for cyber-physical-socio intelligence: A methodology. *Artificial Intelligence, 175*(5), 988–1019.
[34] S. Makridakis, S. (2017). The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. *Futures, 90*, 46–60.
[35] Zhou, P., Zuo, D., Hou, K. M., & Zhang, Z. (2017). A decentralized compositional framework for dependable decision process in self-managed cyber physical systems. *Sensors (Switzerland), 17*(11), 1–33.
[36] Skyttner, L. (1996). General systems theory: Origin and hallmarks. *Kybernetes, 25*(6), 16–22.
[37] Rajkumar, R., Lee, I., Sha, L., & Stankovic, J. (2010). Cyber-physical systems: The next computing revolution. In *Proceedings of Design Automation Conference* (pp. 731–736).
[38] Datta, P., Dey, S., Paul, H. S., & Mukherjee, A. (2014). ANGELS: A framework for mobile grids. In *Proceedings of the International Conference on Applications and*

Innovations in Mobile Computing (pp. 15–20). IEEE.

[39] Stamer, D., Zimmermann, O., & Sandkuhl, K. (2016). What Is a Framework? - A Systematic Literature Review in the Field of Information Systems. In *International Conference on Business Informatics Research* (pp. 145–158)

[40] Aguilar, J., Valdiviezo-dı, P., Riofrio, G., Valdiviezo-Díaz, P., & Riofrio, G. (2017). A general framework for intelligent recommender systems. *Applied Computing and Informatics, 13*(2), 147–160.

[41] Cicirelli, F., Fortino, G., Guerrieri, A., Spezzano, G., & Vinci, A. (2016). A metamodel framework for the design and analysis of smart cyber-physical environments. In *the 20th International Conference on Computer Supported Cooperative Work in Design* (pp. 687–692). IEEE.

[42] Anwar, M. W., Rashid, M., Azam, F., Kashif, M., & Butt, W.H. (2019). A model driven framework for design and verification of embedded systems through System Verilog. *Design Automation for Embedded Systems., 23*(3–4), 179–223.

[43] Levin, M.S. (2011). Four-layer framework for combinatorial optimization problems domain. *Advance in Engineering Software*, 42(12), 1089–1098.

[44] Sztipanovits, J., Bapty, T., Neema, S., Howard, L., & Jackson, E. (2014).physical systems. In *From Programs to Systems. The Systems Perspective in Computing* (pp. 235-248). Springer, Berlin, Heidelberg.

[45] Clark, K., Hengst, B., Pagnucco, M., Rajaratnam, D., Robinson, P., Sammut, C., & Thielscher, M. (2016). A Framework for Integrating Symbolic and Sub-Symbolic Representations. In *Proceeding of the International Joint Conference on Artificial Intelligence* (pp. 2486-2492).

[46] Blersch, M., Landhäuer, M., &Mayer, T. (2018). Semi-automatic generation of active ontologies from web forms for intelligent assistants. In *Proceedings of the International Conference on Software Engineering* (pp. 28–34).

[47] Marques, P. D., Silva, A. J., Henriques, E. M., & Magee, C. L. (2014). A descriptive framework of the design process from a dual cognitive-engineering perspective. *International Journal of Design Creativity and Innovation*, 2(*3*), 142–164.

[48] Long, L., & Kelley, T. (2009). The requirements and possibilities of creating conscious systems. In *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference* (p. 1949).

[49] U. Demiryurek, F. Banaei-kashani, & C. Shahabi, C. (2009). TransDec: A data-driven framework for decision-Making in transportation systems. *Transport Research Forum*, 1–15.

[50] Manzoni, S., Sartori, F., & Vizzari, G. (2005). Towards a general framework for substitutional adaptation in case-based reasoning. In *Congress of the Italian Association for Artificial Intelligence* (pp. 331-342). Springer, Berlin, Heidelberg.

[51] Tanik, U. J., & Begley, A. (2014). An adaptive cyber-physical system framework for cyber-physical systems design automation. In *Applied cyber-physical systems* (pp. 125-140). Springer, New York, NY.

[52] Zhang, L., & He, J. (2011). A formal framework for aspect-oriented specification of cyber physical systems. In *Proceedings of the International Conference on Hybrid Information Technology* (pp. 391-398). Springer, Berlin, Heidelberg.

[53] Lieto, A., Radicioni, D., Rho, V., & Mensa, E. (2017). Towards a unifying framework for conceptual represention and reasoning in cognitive systems.*Intelligenza Artificiale, 11*(2), 139-153

[54] Pavlić, M., Han, Z. D., & Jakupović, A. (2015). Question answering with a conceptual framework for knowledge-based system development "Node of Knowledge". *Expert systems with applications, 42*(12), 5264-5286.

[55] Kim, M., Stehr, M. O., & Talcott, C. (2013). A distributed logic for networked cyber physical systems. *Science of Computer Programming, 78*(12), 2453-2467.

[56] Rabe, F., & Sojakova, K. (2013). Logical relations for a logical framework. *ACM Transactions on Computational Logic, 14*(4), 1-34.

[57] de Roo, A., Sözer, H., Bergmans, L., & Akşit, M. (2013). MOO: An architectural framework for runtime optimization of multiple system objectives in embedded control software. *Journal of Systems and Software, 86*(10), 2502-2519.

[58] Sözer, H., Tekinerdoğan, B., & Akşit, M. (2009). FLORA: A framework for decomposing software architecture to introduce local recovery. *Software: Practice and Experience, 39*(10), 869-889.

[59] Rogers, Y., & Muller, H. (2006). A framework for designing sensor-based interaction to promote exploration and reflection in play. *Journal of Human-Computer Studies, 64*(1), 1-14.

[60] Jeong, S., Baek, Y., & Son, S. H. (2020). Component-based interactive framework for intelligent transportation cyber-physical syst*ems. Sensors, 20*(1), 264.

[61] Jung, M. Y., Deguet, A., & Kazanzides, P. (2010). A component-based architecture for flexible integration of robotic systems. In *Proceedings of International Conference on Intelligent Robots and Systems* (pp.16107-6112). IEEE.

[62] Nayak, A., Reyes Levalle, R., Lee, S., & Nof, S. Y. (2016). Resource sharing in cyberp hysical systems: modelling framework and case studies. *International Journal of Production Research, 54*(23), 6969-6983.

[63] Petnga, L., & Austin, M. (2016). An ontological framework for knowledge modeling and decision support in cyber-physical systems. *Advanced Engineering Informatics, 30*(1), 77-94.

[64] Napp, N., & Klavins, E. (2011). A compositional framework for programming stochastically interacting robots. *The International Journal of Robotics Research, 30*(6), 713-729.

[65] Horváth, I., Li, Y., Rusák, Z., van der Vegte, W. F., & Zhang, G. (2017). Dynamic computation of time-varying spatial contexts. *Journal of Computing and Information Science in Engineering, 17*(1).

[66] Jearanaiwongkul, W., Anutariya, C., & Andres, F. (2019). A semantic-based framework for rice plant disease management. *New Generation Computing*, *37*(4), 499-523.

[67] Sakr, S., & Elgammal, A. (2016). Towards a comprehensive data analytics framework for smart healthcare services. *Big Data Research, 4*, 44-58.

[68] Shih, C. S., Hsiu, P. C., Chang, Y. H., & Kuo, T. W. (2016). Framework designs to enhance reliable and timely services of disaster management systems. In *International Conference on Computer-Aided Design* (pp.1-8). IEEE.

[69] Romdhane, R., Bremond, F., & Thonnat, M. (2010). A framework dealing with uncertainty for complex event recognition. In *Proceedings of the 7th international conference on advanced video and signal based surveillance* (pp. 392-399). IEEE.
[70] Kappé, T., Arbab, F., & Talcott, C. (2016). A compositional framework for preference-aware agents. In *Electronic Proceedings in Theoretical Computer Science*, (Vol. 232, pp. 21-35). Open Publishing Association.
[71] Feng, S., Quivira, F., & Schirner, G. (2016). Framework for rapid development of embedded human-in-the-loop cyber-physical systems. In *Proceedings of the 16th international conference on bioinformatics and bioengineering* (pp. 208-215). IEEE.
[72] Bridgens, B., & Lilley, D. (2017). Understanding material change: Design for appropriate product lifetimes. In *PLATE: Product Lifetimes and The Environment* (pp. 54-59). IOS Press.
[73] Chaki, S., Ouaknine, J., Yorav, K., & Clarke, E. (2003). Automated compositional abstraction refinement for concurrent C programs: A two-level approach.Electronic Notes in *Theoretical Computer Science, 89*(3), 417-432.
[74] Tripakis, S. (2016). Compositionality in the science of system design. *Proceedings of the IEEE*, 104(5), 960-972.
[75] Alonso, D., Sanchez-Ledesma, F., Sanchez, P., Pastor, J. A., & Alvarez, B. (2014).Models and frameworks: a synergistic association for developing component-based applications. *The Scientific World Journal, 14*, 1–17.
[76] Rahimian, V., & Ramsin, R. (2008). Designing an agile methodology for mobile software development: A hybrid method engineering approach. In *Proceedings of the 2nd International Conference on Research Challenges in Information Science* (pp. 337-342). IEEE.
[77] Sokolova, M. V., & Caballero, A. F. (2012). Design and Implementation of the DeciMaS Framework. In *Decision Making in Complex Systems* (pp.47-88). Springer, Berlin, Heidelberg.
[78] Jannaschk, K., & Polomski, T. (2010). A data mining design framework- a preview. In *East European Conference on Advances in Databases and Information Systems* (pp. 571-574). Springer, Berlin, Heidelberg.
[79] Sander, I., Jantsch, A., & Attarzadeh-Niaki, S. H. (2017). ForSyDe: System design using a functional language and models of computation. In *Handbook of Hardware/ Software Codesign* (pp. 99–140).
[80] Ståhl, D., & Bosch, J. (2017). Cinders: The continuous integration and delivery architecture framework. *Information and Software Technology, 83*, 76-93.
[81] Kang, W., Lee, J., Jeong, Y.-S., & Park, J. (2015). VCC-SSF: service-oriented Security framework for vehicular cloud computing. *Sustainability, 7*(2), 2028–2044.
[82] Prabawa, P., & Choi, D. H., (2020). Multi-agent framework for service restoration in distribution systems with distributed generators and static/mobile energy storage systems. *IEEE Access, 8*, 51736–51752.
[83] Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., & Steenkiste, P. (2004).Rainbow: architecture-based self-adaptation with reusable infrastructure. *Computer, 37*(10), 46–54.
[84] Stanojević, V., Vlajić, S., Milić, M., & Ognjanović, M. (2011). Guidelines for

framework development process. In *Proceedings of the 7th Central and Eastern European Software Engineering Conference* (pp. 1-9). IEEE.

[85] Crnkovic, I., Sentilles, S., Vulgarakis, A., & Chaudron, M. R. (2010). A classification framework for software component models. *IEEE Transactions on Software Engineering, 37*(5), 593-615.

[86] PArv, B., Motogna, S.-C., Czibula, I.-G., & Laz, C.-L. (2011). ComDeValCo framework: designing software components and systems using MDD, executable models, and TDD. *Acta Universitatis Sapientiae Informatica, 3*(1), 48–75.

[87] Sifakis, J. (2005). A framework for component-based construction. In *Proceedings of the 3rd International Conference on Software Engineering and Formal Methods* (pp. 293–299). IEEE.

[88] Alberola, J. M., Búrdalo, L., Julián, V., Terrasa, A., & García-Fornes, A. (2014). An adaptive framework for monitoring agent organizations. *Information Systems Frontiers, 16*(2), 239-256.

[89] Pahal, N., Jain, P., Saxena, R., Srivastava, A., Chaudhury, S., & Lall, B. (2019).Context-Aware Reasoning Framework for Multi-user Recom mendations in Smart Home. In *International Conference on Pattern Recognition and Machine Intelligence* (pp. 302-310). Springer, Cham.

[90] Ali, R., Afzal, M., Sadiq, M., Hussain, M., Ali, T., Lee, S., & Khattak, A. (2018). Knowledge-based reasoning and recommendation framework for intelligent decision making. *Expert Systems, 35*(2), e12242.

[91] Hughes, A. J., Barthorpe, R. J., Dervilis, N., Farrar, C. R., & Worden, K. (2021). A probabilistic risk-based decision framework for structural health monitoring *Mechanical Systems and Signal Processing, 150*, 107339.

[92] Wu, B. F., Chen, Y. H., Yeh, C. H., & Li, Y. F. (2013). Reasoning-based framework for driving safety monitoring using driving event recognition. *IEEE Transactions on Intelligent Transportation Systems, 14*(3), 1231-1241.

[93] Di Francescomarino, C., Dumas, M., Maggi, F. M., & Teinemaa, I. (2019). Clustering-based predictive process monitoring. *IEEE Transactions on Services Computing, 12*(06), 896-909.

[94] Guan, L., Chen, H., & Lin, L. (2021). A multi-agent-based self-healing framework considering fault tolerance and automatic restoration for distribution networks. *IEEE Access, 9*, 21522–21531.

[95] Mshali, H. H., Lemlouma, T., & Magoni, D. (2015). Context-aware adaptive framework for e-health monitoring. In *International Conference on Data Science and Data Intensive Systems* (pp. 276-283). IEEE.

[96] Seyoung, P., Mye, S., Haeran, J., & Lee, H. (2016). Situation reasoning framework for the Internet of Things environments using deep learning results. In *International Conference on Knowledge Engineering and Applications* (pp. 133–138). IEEE.

[97] Thaduri, A., Kumar, U., & Verma, A. K., (2017). Computational intelligence framework for context-aware decision making. *International Journal of System Assurance Engineering and Management, 8*, 2146–2157.

[98] Wu, D., Liu, S., Zhang, L., Terpenny, J., Gao, R. X., Kurfess, T., & Guzzo, J. A. (2017). A fog computing-based framework for process monitoring and prognosis in

cyber-manufacturing. *Journal of Manufacturing Systems, 43*, 25-34.
[99] Du, Z., Tang, J., & Ding, Y. (2019). POLAR++: Active One-shot Personalized Article Recommendation. *IEEE Transactions on Knowledge and Data Engineering, 33*(6), 2709-2722.
[100] Wan, J., Yan, H., Suo, H., & Li, F. (2011). Advances in cyber-physical systems research. *KSII Transactions on Internet and Information Systems, 5*(11), 1891-1908.
[101] Petersson, H., Motte, D., Eriksson, M., & Bjärnemo, R. (2013). Integration of computer aided design analysis into the engineering design process for use by engineering designers. In *Proceedings of International Mechanical Engineering Congress and Exposition* (Vol. 56413, pp. V012T13A002). ASME.
[102] Steinbauer G., & Wotawa, F. (2013). Model-Based Reasoning for Self-Adaptive Systems – Theory and Practice. In *Assurances for Self-Adaptive Systems* (pp. 187–213). Springer Berlin Heidelberg.
[103] Zhang, H., Li, F., Wang, J., Wang, Z., Shi, L., Zhao, J., ... & Szczerbicki, E. (2017). Adding intelligence to cars using the neural knowledge DNA. *Cybernetics and Systems, 48*(3), 267-273.
[104] Khan, F., Eker, O. F., Sreenuch, T., & Tsourdos, A. (2014). Multi-domain modeling and simulation of an aircraft system for advanced vehicle-level reasoning research and development. *International Journal of Advanced Computer Science and Applications, 5*(4), 86-96.
[105] Lu, T., Zhao, J., Zhao, L., Li, Y., & Zhang, X. (2015). Towards a framework for assuring cyber physical system security. *International Journal of Security and Its Applications, 9*(3), 25-40.
[106] Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *IEEE computational intelligence magazine, 5*(4), 13-18.
[107] Crowder, J. A., Carbone, J. N., & Friess, S. A. (2014). *Artificial cognition architectures*. New York: Springer.
[108] Riegler, A. (2002). When is a cognitive system embodied?. *Cognitive Systems Research, 3*(3), 339–348.
[109] Petnga, L., & Austin, M. (2013). Ontologies of time and time-based reasoning for MBSE of cyber-physical systems. *Procedia Computer Science, 16*, 403-412.
[110] Tenorth M., & Beetz, M. (2013). KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *International Journal of Robotics and Research, 32*(5), 566–590.
[111] Kunze, L., Hawes, N., Duckett, T., Hanheide, M., & Krajník, T. (2018).Artificial intelligence for long-term robot autonomy: A survey. *IEEE Robotics and Automation Letters, 3*(4), 4023-4030.
[112] Rajeswari, P. V. N., & Prasad, T. V. (2012). Hybrid Systems for Knowledge Representation in Artificial Intelligence. *International Journal Advance Research in Artificial Intelligence, 1*(8), 31–36.
[113] Almeida, A., & Lopez-de-Ipina, D. (2012). A Distributed Reasoning Engine Ecosystem for Semantic Context-Management in Smart Environments. *Sensors, 12*(8), 10208–10227.
[114] Li, X., Martinez, J.-F., & Rubio, G. (2015). Context Aware Middleware Architectures:

Survey and Challenges. *Sensors, 15*(8), 20570–20607.
[115] Li, X., Martinez, J.-F., & Rubio, G. (2017). Towards a Hybrid Approach to Context Reasoning for Underwater Robots., *Applied Science, 7*(2), 183.
[116] Gomes, P. E., Marques, A., Costa, Â., Novais, P., & Neves, J. (2010). Patient monitoring under an ambient intelligence setting. In *Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence* (pp. 185-188). Springer, Berlin, Heidelberg.
[117] Gouin-Vallerand, C., Abdulrazak, B., Giroux, S., & Dey, A. K. (2013). A contextaware service provision system for smart environments based on the user interaction modalities. *Journal of Ambient Intelligence and Smart Environments*, 5(1), 47-64.
[118] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., & Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and mobile computing, 6*(2), 161-180.
[119] Gilman, E. (2015). *Exploring the use of rule-based reasoning in ubiquitous computing applications* (Doctoral dissertation, University of Oulu, Finland).
[120] Jakobson, G., Buford, J., & Lewis, L. (2006). A framework of cognitive situation modeling and recognition. In Military Communications conference (pp. 1-7). IEEE.
[121] Ye, J., Dobson, S., & McKeever, S. (2012). Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing, 8*(1), 36–66.
[122] Lewis, P. R., Chandra, A., Parsons, S., Robinson, E., Glette, K., Bahsoon, R., ... & Yao, X. (2011). A survey of self-awareness and its application in computing systems. In *Proceedings of the 5th Conference on Self-Adaptive and Self-Organizing Systems Workshops* (pp. 102-107). IEEE.
[123] Schlatow, J., Moostl, M., Ernst, R., Nolte, M., Jatzkowski, I., Maurer, M.,... & Herkersdorf, A. (2017). Self-awareness in autonomous automotive systems. In *Design, Automation & Test in Europe Conference & Exhibition* (pp.1050-1055). IEEE.
[124] Gurgen, L., Gunalp, O., Benazzouz, Y., & Gallissot, M. (2013). Self-aware cyber-physical systems and applications in smart buildings and cities. In *Design, Automation & Test in Europe Conference & Exhibition* (pp. 1149-1154). IEEE.
[125] Hudson, D. A., & Manning, C. D. (2018). Compositional attention networks for machine reasoning. In *International Conference on Learning Representations* (pp.1-20).
[126] Basu, C., Agrawal, A., Hazra, J., Kumar, A., Seetharam, D. P., Béland, J.,... & Lafond, C. (2014). Understanding events for wide-area situational awareness. In *Innovation Smart Grid Technologies* (pp. 1-5). IEEE.
[127] Sene, A., Kamsu-Foguem, B., & Rumeau, P. (2015). Telemedicine framework using case-based reasoning with evidences Computer Methods and Programs in *Biomedicine, 121*(1), 21-35.
[128] Subbaraj R., & Venkatraman, N. (2016). Reasoning in context aware computing – a review issn. *International Journal of Pharmacology and Technology*, 8(4), 5021–5032.
[129] Cimino, M. G., Lazzerini, B., Marcelloni, F., & Ciaramella, A. (2012). An adaptive rule-based approach for managing situation-awareness. *Expert Systems with Applications*, 39(12), 10796-10811

[130] Uusitalo, L., (2007). Advantages and challenges of Bayesian networks in environmental modelling. *Ecological Modelling, 203*(3-4), 312–318.
[131] Pan, I., & Bester, D. (2018). Fuzzy Bayesian Learning. *IEEE Transaction on Fuzzy Systems, 26*(3), 1719–1731.
[132] Yan, K.-Y. (2012). Fuzzy-probabilistic logic for common sense. In *Proceedings of the International Conference on Artificial General Intelligence* (pp. 372-379). Springer, Berlin, Heidelberg.
[133] Prentzas, J., & Ioannis, H. (2011). Case-based reasoning integrations: approaches and applications. *Case-based reasoning: process, suitability and applications*, 1–28.
[134] Hao, J. G., Bouzouane, A., Bouchard, B., & Gaboury, S. (2018). Activity inference engine for real-time cognitive assistance in smart environments. *Journal of Ambient Intelligence and Humanized Computing, 9*(3), 679–698.
[135] Bench-Capon, T. J. M., & Dunne, P. E. (2007). Argumentation in artificial intelligence. *Artificial Intelligence, 171*(10–15), 619–641.
[136] Caillou, P., Gaudou, B., Grignard, A., Truong, C. Q., & Taillandier, P. (2017). A simple-to-use BDI architecture for agent-based modeling and simulation. In *Advances in Social Simulation* (pp. 15-28). Springer, Cham.
[137] Elkhodary, A., Esfahani, N., & Malek, S. (2010). FUSION: a framework for engineering self-tuning self-adaptive software systems. In *Proceedings of the 18th ACM SIGSOFT international symposium on Foundations of software engineering* (pp. 7).
[138] Berka, P. (2011). NEST: A Compositional approach to rule-based and case-based reasoning. *Advance in Artificial Intelligence*, 1–15.
[139] Tsafnat, G., & Coiera, E. W. (2009). Computational reasoning across multiple models. *Journal of American Medical Informatics Association, 16*(6), 768–774.
[140] Yu, C., & Luo, Y. (2006). Research on the modeling system for decision-making problems based on knowledge. In *International Conference on Machine Learning and Cybernetics* (pp. 1960-1966).
[141] Li, B., Li, J., Tang, K., & Yao, X. (2015). Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys, 48*(1), 1-35.
[142] Senne, K. D., & Condon, G. R., (2007). Integrated Sensing and Decision Support. *Lincoln Laboratory Journal, 16*(2), 237–244.
[143] Ong, D. C. C., Khaddaj, S., & Bashroush, R. (2011). Logical reasoning and decision making. In *Proceedings of the 10th International Conference on Cybernetic Intelligent Systems* (pp. 26–31). IEEE.
[144] Baclawski, K., Chan, E. S., Gawlick, D., Ghoneimy, A., Gross, K. C., & Liu, Z. H. (2017). Self-adaptive dynamic decision making processes. In *Conference on Cognitive and Computational Aspects of Situation Management* (pp. 1-6). IEEE
[145] Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M.,...& Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In *Software Engineering for Self-adaptive Systems* (pp. 48-70). Springer, Berlin, Heidelberg.
[146] Berka, P. (2002). Adaptive features of machine learning methods. In *Proceedings of the 1st International IEEE Symposium Intelligent Systems* (Vol. 2, pp.

40–43). IEEE.
[147] Levin, S. A. (2002). Complex adaptive systems : exploring the known, the unkknown and the unknowable. *Bulletin of the American Mathematic Society, 40*(1), 3–19.
[148] Watson, R. A., Buckley, C. L., & Mills, R. (2011). Optimization in 'self-modeling' complex adaptive systems. *Complexity, 16*(5), 17–26.
[149] Cámara, J., Bellman, K. L., Kephart, J. O., Autili, M., Bencomo, N., Diaconescu, A.,...& Tivoli, M. (2017). Self-aware computing systems: Related concepts and research areas. In *Self-Aware Computing Systems* (pp. 17-49). Springer, Cham.
[150] Amara-Hachmi, N. (2006). An Ontology-based Model for Mobile Agents Adaptation in Pervasive Environments. In *International Conference on Computer Systems and Applications* (pp. 1106–1109). IEEE.
[151] Wang, X., Feng, Z., Huang, K., & Tan, W. (2017). An automatic self-adapttion framework for service-based process based on exception handling. *Concurrency and Computation: Practice and Experience, 29*(5), e3984.
[152] Papamarkos, G., Poulovassilis, A., & Wood, P.T. (2003). Event-Condition-Action rule languages for the semantic web. In *Proceedings of Conference on Semantic Web and Databases* (pp. 309-327).
[153] Goodwin, T. R. & Harabagiu, S. M. (2017). Knowledge representations and inference techniques for medical question answering. *ACM Transaction on Intelligent Systems and Technology*, 9(2), 1-26).
[154] Bui, K. H. N., Pham, X. H., Jung, J. J., Lee, O. J., & Hong, M. S. (2015). Contextbased traffic recommendation system. In *International Conference on Context Awareness Systems and Applications* (pp. 122-131). Springer, Cham.
[155] Wu, Z., Li, L., & Liu, H. (2020). Process Knowledge Recommendation System for Mechanical Product Design. *IEEE Access, 8*, 112795–112804.
[156] Konstan, J. A., & Riedl, J. (2012). Recommender systems: From algorithms to user experience. *User Modelling and User-adapted Interaction, 22*(1–2), 101–123.
[157] Aldrich, S. E. (2011). Recommender systems in commercial use. *AI Magazine, 32*(3), 28-34.
[158] Tang, J., Hu, X., & Liu, H. (2013). Social recommendation: a review. *Social Network Analysis and Mining, 3*(4), 1113-1133.
[159] Shokeen, J., & Rana, C. (2020). A study on features of social recommender systems. *Artificial Intelligence Review, 53*(2), 965–988.
[160] Li, Y., Liu, J., & Ren, J. (2019). Social recommendation model based on user interaction in complex social networks. *PLoS One, 14*(7), 1–17.
[161] Cena, F., Rapp, A., Musto, C., & Semeraro, G. (2020). Generating recommendations from multiple data sources: A methodological framework for system design and its application. *IEEE Access, 8*, 183430–183447.
[162] Dao, A.Q., Koltai, K., Cals, S.D., Brandt, S. L., Lachter, J., Matessa, M.,...& Johnson, W.W. (2015). Evaluation of a recommender system for single pilot operations. *Procedia Manufacturing, 3*, 3070-3077.
[163] Jannach, D., Jugovac, M., & Lerche, L. (2016). Supporting the design of machine learning workflows with a recommendation system. *ACM Transactions on Interactive*

Intelligence Systems, 6(1), 1-35.
[164] Bao, Y., & Jiang, X. (2016). An intelligent medicine recommender system framework. In *Proceedings of the 11th Conference on Industrial Electronics and Applications, ICIEA 2016* (pp. 1383–1388). IEEE.
[165] Jonnalagedda, N., & Gauch, S. (2013). Personalized news recommendation using twitter. In *Proceedings of the International Joint Conferences on Web Intelligence and Intelligent Agent Technologies* (Vol. 3, pp. 21-25). IEEE.
[166] Kartoglu, I. E., & Spratling, M. W. (2018). Two collaborative filtering recommender systems based on sparse dictionary coding *Knowledge and Information Systems*, 57(3), 709–720.
[167] Cacheda, F., Carneiro, V., Fernández, D., & Formoso, V. (2011). Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web*,5(1), 1-33.
[168] Ma, K. (2016). Content-based Recommender System for Movie Website. (Master's thesis, Examensarbete Inom Informations- OCH Kommun. Tek).
[169] Hussein, T., Linder, T., Gaulke, W., & Ziegler, J. (2014). Hybreed: A software framework for developing context-aware hybrid recommender systems. *User Modeling and User-Adapted Interaction, 24*(1-2), 121-174.
[170] Gomaa, W.H., & Fahmy, A.A. (2013). A survey of text similarity approaches. *International Journal of Computer Applications, 68*(13), 13-18.
[171] Cao, Y., Wang, X., He, X., Hu, Z., & Chua, T. S. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The World Wide Web Conference* (pp. 151-161).
[172] Liu, J., & Duan, L. (2021). A Survey on Knowledge Graph-Based Recommender Systems. In *Proceedings of the 5th Advanced Information Technology, Electronic and Automation Control Conference* (Vol. 5, pp. 2450-2453). IEEE
[173] Aguilar, J., Valdiviezo-Díaz, P., & Riofrio, G. (2017). A general framework for intelligent recommender systems. *Applied computing and informatics*, *13*(2), 147-160.
[174] Poczeta, K., Papageorgiou, E. I., & Gerogiannis, V. C. (2020). Fuzzy Cognitive Maps Optimization for Decision Making and Prediction. *Mathematics, 8*(11), 2059.
[175] Willmann, R. (2019). Product design with integrated holistic validation of economics of metal-based 3d-printing by the help of a recommender system. In *International Conference on Industrial Technology* (pp. 806-811). IEEE.
[176] Chen, G., & Ge, Z. (2020). Hierarchical Bayesian Network Modeling Framework for Large-Scale Process Monitoring and Decision Making. *IEEE Transactions on Control Systems and Technology, 28*(2), 671–679.
[177] Demelo, J. & Sedig, K. (2021). Forming cognitive maps of ontologies using interactive visualizations. *Multimodal Technology and Interaction, 5*(1), 1–39.
[178] Sheridan, P., Onsjö, M., Becerra, C., Jimenez, S., & Dueñas, G. (2019). An OntologyBased Recommender System with an Application to the Star Trek Television Franchise. *Future Internet, 11*(9), 182.
[179] Hu, C. J., Cheng, C. C., Wu, H. Y., & Chao, N. T. (2014). An Ontology Based

Recommendation Mechanism for Lighting System Design. In *International Symposium on Computer, Consumer and Control* (pp. 239-243). IEEE.

[180] Rodriguez, N. D. (2011). A Framework for Context-Aware Applications for Smart Spaces. In *International Symposium on Applications and the Internet* (pp.218–221). IEEE.

[181] Esheiba, L., Elgammal, A., Helal, I., & El-Sharkawi, M. E. (2021). A hybrid knowledge-based recommender for product-service systems mass customization. *Information*, 12(8), 296.

[182] Gabriel De Souza, P. M., Jannach, D., & Da Cunha, A. M. (2019). Contextual hybrid session-based news recommendation with recurrent neural networks.*IEEE Access*, 7, 169185-169203.

[183] Mahmoud, H., & Akkari, N. (2016). Shortest Path Calculation: A Comparative study for location-based recommender system. In *World Symposium on Computer Applications & Research* (pp. 1-5). IEEE.

[184] Jain, A., & Gupta, C. (2018). Fuzzy logic in recommender systems. In *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications* (pp. 255-273). Springer, Cham.

[185] Jie, H., ChangSheng, L., & ChengLi, L. (2019). Design and implementation of the cross-harmonic recommender system-based on spark. In *International Conference on Advanced Hybrid Information Processing* (pp. 461-474). Springer, Cham.

[186] Jallouli, M., Lajmi, S., & Amous, I. (2017). Designing recommender system: Conceptual framework and practical implementation. *Procedia Computer Science*, 112, 1701-1710.

[187] Glodek, J. (2013). Recommender System for a Knowledge Base. (Doctoral dissertation, Warsaw University of Technology).

[188] Mahmood, M. A., Al-Shammari, E. T., El-Bendary, N., Hassanien, A. E., & Hefny, H. A. (2013). Recommender system for ground-level Ozone predictions in Kuwait. In *Federated Conference on Computer Science and Information Systems* (pp. 107-110). IEEE.

[189] Laseno, F.U., & Hendradjaya, B. (2019). Knowledge-based filtering recommender system to propose design elements of serious game. In *International Conference on Electrical Engineering and Informatics* (pp. 158-163). IEEE.

[190] Taghipour, N., Kardan, A., & Ghidary, S.S. (2007). Usage-based web recommendations: A reinforcement learning approach. In *Proceedings of the ACM conference on Recommender systems* (pp. 113-120).

[191] Song, Q., Wang, G., & Wang, C. (2012). Automatic recommendation of classification algorithms based on data set characteristics. *Pattern recognition*, *45*(7), 2672-2689.

[192] Hussein, A. S., Omar, W. M., Li, X., & Amer Hatem, M. (2014). Smart collaboration framework for managing chronic disease using recommender system. *Health Systems, 3*(1), 12-17

[193] Linda, S., & Bharadwaj, K. K. (2018). A decision tree based context-aware recommender system. In *Proceedings of the International Conference on Intelligent Human Computer Interaction* (pp. 293-305). Springer, Cham.

[194] Shani, G., Heckerman, D., & Brafman, R. I. (2005) An MDP-based recommender system. *Journal of Machine Learning and Research, 6*, 1265–1295.
[195] Montaner, M., López, B., & De La Rosa, J. L. (2003). A taxonomy of recommender agents on the internet. *Artificial Intelligence Review, 19*(4), 285–330.
[196] Rana, C., & Jain, S. K. (2015). A study of the dynamic features of recommender systems. *Artificial Intelligence Review, 43*(1), 141–153.
[197] Unnikrishnan, G., Mathew, D., Jose, B. A., & Arvind, R. (2019). Hybrid Route Recommender System for Smarter Logistics. In *IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conferenceon Intelligent Data and Security (IDS)* (pp. 239-244). IEEE.
[198] Rabiu, I., Salim, N., Da'u, A., & Osman, A. (2020). Recommender system based on temporal models: a systematic review. *Applied Sciences, 10*(7), 2204.
[199] Sun, B., &Dong, L. (2017). Dynamic model adaptive to user interest drift-based on cluster and nearest neighbors. *IEEE Access, 5*, 1682–1691.
[200] Tareq, S. U., Noor, M. H., & Bepery, C. (2020). Framework of dynamic recommendation system for e-shopping. *International Journal of Information Technology, 12*(1), 135-140.
[201] Ning, X., Fan, Z., Burgun, E., Ren, Z., & Schleyer, T. (2021). Improving information retrieval from electronic health records using dynamic and multi-collaborative filtering. *Plos one, 16*(8), e0255467.
[202] Jannach, D., Manzoor, A., Cai, W., & Chen, L. (2021). A Survey on conversational recommender Systems. *ACM Computing Surveys, 54*(5), 1-36.
[203] Zhao, G., Zhao, J., Li, Y., Alt, C., Schwarzenberg, R., Hennig, L., ... & Xu, F. (2019). MOLI: Smart Conversation Agent for Mobile Customer Service. *Information, 10*(2), 63.
[204] Paul, A., Haque Latif, A., Amin Adnan, F., & Rahman, R. M. (2019). Focused domain contextual AI chatbot framework for resource poor languages. *Journal of Information and Telecommunication, 3*(2), 248-269.
[205] Brenas, J. H., Shin, E. K., & Shaban-Nejad, A. (2019). A hybrid recommender system to guide assessment and surveillance of adverse childhood experiences. In *Health Informatics Vision: From Data via Information to Knowledge* (pp. 332-335). IOS Press.
[206] Lin, K. S., & Ke, M. C. (2015). A virtual reality based recommender system for interior design prototype drawing retrieval. In *New trends in intelligent information and database systems* (pp. 141-150). Springer, Cham.
[207] Aciar, S., Zhang, D., Simoff, S. & Debenham, J. (2006). Recommender system based on consumer product reviews. In *International Conference on Web Intelligence* (pp. 719–723). IEEE.
[208] Hoeksema, J. (2000). Compositionality of meaning. In *Morphology: A Handbook on Inflection and Word-Formation* (pp. 851–857). Berlin: Walter de Gruyter
[209] Skyttner, L., (2006). *General systems theory: Problems, perspectives, practice*.
[210] Ghani, N., Hedges, J., Winschel, V., & Zahn, P. (2018). Compositional game theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science* (pp. 472–481).
[211] Muccini, H., Sharaf, M., & Weyns, D. (2016). Self-adaptation for cyber-physical systems:

A systematic literature review. In *Proceedings-11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2016* (pp. 75–81).

[212] Yu, Z., Zhou, L., Ma, Z., & El-Meligy, M.A. (2017). Trustworthiness modeling and analysis of cyber-physical manufacturing systems. *IEEE Access, 5*, 26076–26085.

[213] Rajhans, A., Bhave, A., Ruchkin, I., Krogh, B. H., Garlan, D., Platzer, A., & Schmerl, B. (2014). Supporting heterogeneity in cyber-physical systems architectures. *IEEE Transactions on Automatic Control, 59*(12), 3178-3193.

[214] Dragomir, I., Preoteasa, V. & Tripakis, S. (2016). Compositional semantics and analysis of hierarchical block diagrams. In *International Symposium on Model Checking Software* (pp. 38-56). Springer, Cham

[215] Schaefer, I., & Poetzsch-Heffter, A. (2008). Compositional reasoning in model-based verification of adaptive embedded systems. In *Proceedings of the 6th International Conference on Software Engineering and Formal Methods* (pp. 95-104). IEEE.

[216] Saitta, L. & Zucker, J.-D. (2013). Abstraction in different disciplines. In *Abstraction in Artificial Intelligence and Complex Systems* (pp. 11–47). Springer New York.

[217] Subagdja, B., & Tan, A. H. (2016). Interactive Teachable Cognitive Agents: Smart Building Blocks for Multiagent Systems. *IEEE Transactions on Systems, Man and Cybernetics, 46*(12), 1724–1735.

[218] Guo, K., Lu, Y., Gao, H., & Cao, R. (2018). Artificial intelligence-based semantic internet of things in a user-centric smart city. *Sensors (Switzerland), 18*(5), 1341.

[219] Majewski, M. & Kacalak, W. (2017). Smart control of lifting devices using patterns and antipatterns. In *Computer Science On-line Conference* (pp. 486-493).Springer, Cham.

[220] Ng, K. H., Du, Z., & Ng, G.W. (2017). DSO cognitive architecture: unified reasoning with integrative memory using global workspace theory. In *International Conference on Artificial General Intelligence* (pp. 44-53).

[221] Sarathy, V., & Scheutz, M. (2018). A logic-based computational framework for inferring cognitive affordances. *IEEE Transactions on Cognitive and Developmental Systems, 10*(1), 26–43.

[222] Horváth, I., Rusák, Z., & Li, Y. (2017). Order beyond chaos: introducing the notion of generation to characterize the continuously evolving implementations of cyber-physical systems. In *Proceedings of the ASME Design Engineering Technical Conference* (p. V001T02A015).

Chapter 3

Research cycle 2:
Conceptualization of a demonstrative part of the proposed active recommender framework

3.1 Objectives and methodological framing of the second research cycle

3.1.1 Objectives

In Chapter 2, the activities and results of the knowledge aggregation and requirement engineering completed in the first research cycle were discussed. This intelligence was used to form a robust idea about an active recommender framework (ARF) as an enabling tool to support the designers of application-specific reasoning mechanism (ASRM). Concerning a comprehensive conceptualization of the targeted ARF, the requirements were specified at two levels, namely, (i) on the level of the framework as a whole, and (ii) on the level of the computational mechanisms included in the framework. Two primary functional requirements for the entire framework were specified as follows. The ARF should: (i) simultaneously perform process monitoring and design support in real-time; and (ii) generate a recommendation based on the information processed by a designer and the dynamically changing context associated with the design processes.

On the level of computational mechanisms, six major functional requirements were defined as follows: (i) perform activity-related process monitoring throughout the various processes of reasoning mechanism development (RMD); (ii) recognize doubtful (unexpected) eventsit the design sessions in real-time; (iii) capture the logical, temporal, and methodological relationships of design actions; (iv) identify any obstacle in the design process as deviation from a target procedural network; (v) propose a feasible recommendation in human-readable format; and (vi) evaluate the designer's decision and the effects of the received recommendation.

The overall objective of the second research cycle was conceptualization of the computational enablers for a particular realization of an ARF. Conceptualization was challenging because the ARF was supposed to provide support not only to design decision-making, but also to the management of the design activities and to the development process of ASRMs. Two concrete goals were addressed in this research cycle. On the one hand, we intended to elaborate and operationalize a dedicated methodology to help the realization of the conceived functionalities of the ARF. On the other hand, we intended to specify the support functions that RMDs can expect from a partial implementation of this ARF. In order to achieve these goals, proper underpinning knowledge (theories) were needed including (i) the conceptual clarification of the notion of ARFs; (ii) the essence and implementation details of the facilitating methodology; and (iii) the expectations for the functionality of the ARF required when designing ASRM for concrete practical cases.

Based on these objectives, the guiding research question was formulated in the following way:

RQ: *What constituents enable a feasible and efficient (i) functionality, (ii) architecture, (iii) computational operation, and (iv) implementation of an ARF, which resolves procedural obstacles in a design process and provides decisional benefits for designers of ASRMs in the case of a family of S-CPSs?*

3.1.2 Research methodology

Design inclusive research (DIR) was applied as a methodological frame for conducting this research cycle. There were three phases of research actions: (i) explorative phase; (ii) constructive phase; and (iii) confirmative phase. In the explorative phase, the research actions started with a broad and systematic knowledge aggregation and continued with the ideation of what services an ARF can offer to a designer with the goal to support the logical and computational development of reasoning mechanisms. The technical input information for conceptualization was derived from the requirements which were stated and analyzed in the previous chapter, and this was extended with information by brainstorming and critical systemic thinking.

In general, conceptual-ization of an ARF is a complicated matter for the reason that (i) the process of designingASRMs and (ii) the specific functionality, interoperation, development process, and implementation methodology of the ARF should concurrently be taken into consideration. This is referred to as duality of development. Thus, the concept of ARF wasdeveloped based on assumptions about a particular family of ASRMs, its design process and actions, and the specific supportive role of an ARF.

In the constructive phase, concrete conceptualization and design actions were planned and completed. In the first place, the fundamentals of conceptualization of the ARF were discussed. The conceptualized mechanisms of the ARF included (i) process monitoring and (ii) design decision support. For the operational concept of the former mechanism,

event-based monitoring of the changes in the behavior of the designer was considered. The observation of a non-usual event in a design process is typified in five ways (type A-E) according to the contribution levels of the ARF to the design process. For the latter mechanism, non-usual event type B was selected as a fundamental for a generation of recommendations.

The ARF was conceptualized based on the proposed multi-perspective method. In the conceptualization process, the major activities were: (i) specification of functionality, (ii) system and component architecting, (iii) planning of the computational process, and (iv) specification of the algorithms and data constructs. In the last confirmative phase, the research actions focused on testing the feasibility of the concept, considering a specific part of an ARSM design process. The considered part of this ARSM design process concerned the working principle exploration (WPE) session of an automated parking assist system (APAS).This made it possible to conclude about and consolidate the findings on the conceptualization. Regarding the conclusions, the identification of requirements for the computational implementation of the ARF played a crucial role.

3.2 Generic assumptions concerning the active recommender frameworks

3.2.1 Assumptions with respect to complexity management

3.2.1.1 Assumptions concerning the manifestations of complexity in real-life cases

Smart CPSs are supposed to incorporate capabilities necessary for handling an unforeseen situation with incomplete information and limited control in the real-life problem. This means a smart CPS must be able to adapt its reasoning process to deal with dynamic situations in the target application. The complexity of a situation can be characterized by three factors: (i) the type of entities, (ii) the number of entities, and(iii) the number of relationships among the entities. If the local world comprises only few entities and a limited number of relationships, then one situation is probably sufficient to find an optimal solution for the application problem.

However, the complexity will increase in the real-life problems when the states of the entities and their relationships change over time. In addition, interplay among multiple situations may also happen in the local world. Therefore, we had to come to the conclusion that one situation might not be enough for finding an optimal working principle in the case of our actual real-life problem (working principle exploration for an automated parking assist system). The fact of the matter is that multiple situations should be considered, which are to be categorized into a group of similar situations. To handle this particular manifestation of complexity, the following generic assumptions were formulated:

Assumption 1: In a local world, a situation features a linearly (non-dynamically and non-exponentially) changing number of entities and relationships among them.

Assumption 2: A new situation is caused and defined by the change in the number of entities and/or their relationships.

Assumption 3: In a regular case, consideration of one situation is sufficient to end up with an optimal working principle.

3.2.1.2 Assumptions concerning the handling of application complexity

An automated parking assist system (APAS) was selected as the target application and design context. The reasoning mechanisms of ARSMs perform multiple reasoning processes. Due to the complexity of the APAS as application and the abundance of the related context information, we had to impose strong scoping. Towards this end, we decided to focus only on the *working principle exploration* (WPE) for an APAS. In essence, a working principle is defined as a set of rules or conditions that underpin the completion of a work or a task at hand. In the case of the APAS, the working principle depends on the arrangement of the actual parking area. The included objects, their eometric measures, their distances and orientations may be different. All of these cannot be precisely considered. The working principle also depends on the type of parking situation, i.e., if it is parallel, slanted, or perpendicular. For each of them a dedicated/optimal working principle should be found based on an approximate arrangement of a parking case. This means that one working principle can offer multiple alternative solutions, but optimization is not really necessary. A selection of the most appropriate ('good enough') working principles can be done based on the parking criteria. A collision-free and fast parking is a common criterion at evaluating a motion path trajectory for parking. It should be determined as a ('good enough')rather than as a case-dependent absolute solution.

Assumption 4: A specific useful working principle should be found for each concrete parking situation, but it can be sub-optimal.

Assumption 5: The selection of the best matching working principle should be based on a good enough approximation of the arrangement and features of a parking case.

Assumption 6: The selected working principle should be the first appropriate (good enough) for safe and fast parking

3.2.1.3 Assumptions concerning the handling of implementation complexity

Engineering systems are typically made up of a large number of inter-related subsystems, modules, and components. The complexity of the systems should be considered from six main sources: (i) the number of components; (ii) the functionality of the components, (iii) the functional interrelationships of components; (iv) the architectural interrelationships of components; (v) numbers of their interrelationships, and (vi) the operational scenario of the system and its components. The degree of complexity of an ARF should also be evaluated based on these factors. The design process of the whole of the ARF is a rather complex one

and it is a challenge to capture it as a single design problem. The strategy of divide-and-conquer is to be applied. This necessitates the decomposition of the design process of the ARF into self-contained process element. The above considerations were the basis of the assumptions that were made concerning the implementation complexity.

Assumption 7: The use of minimal number of constituents reduces the implementation complexity of the ARF in a natural manner.

Assumption 8: If pre-produced constituents can be used in an unchanged or a (slightly) modified form, and if their interfaces are proper, then it also can simplify the implementation complexity.

Assumption 9: The internal dependences of the constituents should be kept minimal in order to reduce operational and implementation complexity.

Assumption 10: Standard system development methods such as modular construction and component-based design should be applied to reduce implementation complexity.

3.2.1.4 Assumptions concerning the methodological approach to conceptualization of the active recommender framework

Conceptualization of the ARF includes a huge set of activities, which needs a systematized (systemic and systematic) approach. In order to consider all relevant intelligence and constraints, a top-down approach was used. This simultaneously considered two perspectives, namely (i) the identification the potential needs for support servicesbased on the analysis of the schematized design scenario, and (ii) the specification (decomposition) of the functionality and the architecture of the ARF.

To make the decomposition systematic, we used the concept of 'multi-layers' in the specification process of the ARF. The following layers were considered: (i) determination of the required services, (ii) specification of the functionalities, (iii) allocation of system components (architecting), (iv) development of algorithms and data constructs, (v) planning of the computational and interaction process, and (vi) validation (testing). These layers and their arrangement are shown in Figure 3.1. The introduced decomposition to activity layers facilitated the integration of the computational functions, the software components, the process flow, the algorithms/data constructs, and the external interactions.

A function is an abstraction of an operation, activity, process, or action performed by the constituents of the ARF to achieve a particular objective within a prescribed set of requirements. As customary, the functional specification started out from the main functions and specified various levels of sub-functions, which mainly represented computational actions that the ARF should accomplished. The functional decomposition was completed by the specification of the elementary functions that could be implemented by various algorithms. The constituents of the ARF are composed according to a multi-level, logical (functional) and architectural structure,which provides the framework. The process of the concomitant

architectural structuring of the software went through the definition of the computational mechanisms, the modules, the units, and the algorithms.

There were several generic assumptions conceived concerning the methodological approach to the conceptualization of ARF. The most significant ones were formulated as follows:

Assumption 11: The whole conceptualization is driven by the idea of service-provisioning systems.

Assumption 12: Due to the complexity and the novelty, a path finding and iterative approach has to be methodologically preferred instead of a linear or a waterfall-type procedural scheme.

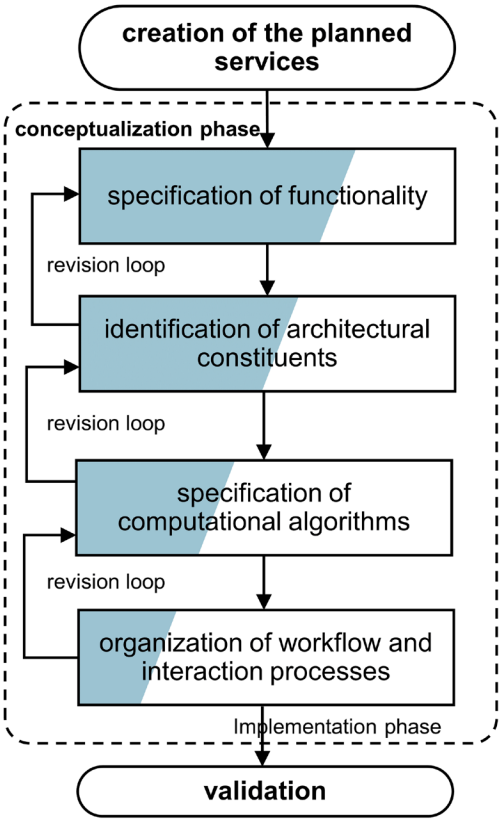


Figure 3.1: Methodological approach to conceptualization and implementation of the ARF

Assumption 13: The conceptualization should go through the stages of (i) specification of the functional concept, (ii) elaboration of the architecting concept, (iii) specification of the algorithms concept, and (iv) organization of the workflow and interaction concepts.

Assumption 14: The implementation should concretize the conceptual elements as computational elements.

Assumption 15: The compatibility and interoperability of algorithms should be tested (validated) in the context of application

3.2.2 Assumptions concerning the reasoning mechanism development

Typically, a reasoning mechanism executes a complicated inference process that involvesmultiple logical operations on logical expressions/statements to draw conclusions [1]. In the case of smart CPSs, included in the reasoning mechanisms, the computational algorithms process the input data and derive new knowledge based on the pre-programmed inferring/reasoning logic in a particular context for a given purpose. As mentioned above, an APAS is selected as the target application (design context). Its reasoning mechanism performs those procedural reasoning processes, which include multiple feedback loops. Due to this innate complicatedness, the design process of the ASRM should be decomposed into multiple design sessions. These considerations formed the basis of the assumptions concerning the ASRM development.

Assumption 16: Conceptualization of ASRMs for the APAS includes four design sessions, namely (i) problem specification; (ii) situation analysis; (iii) working principle exploration; and (iv) decision logic generation.

Assumption 17: The reasoning mechanisms of the APAS belong to a class or subclass of reasoning mechanisms for S-CPSs.

Assumption 18: The construction of a reasoning mechanism for a CPS generally includes multiple feedback loops, which assume information exchange between the phases of designing.

Assumption 19: Conceptualization of APAS needs to consider seven computational processes, namely (i) continuous sensing, (ii) recognizing an event, (iii) inferring a situation, (iv) exploring working principles, (v) decision-making, (vi) adapting to a new situation, and (vii) actuating.

3.2.3 Assumptions concerning the design process and design actions

As briefly mentioned above, designing of a reasoning mechanism for smart CPSs is a complicated process because of the natural complexity of reasoning mechanisms. It is not only the result of the set of the past (historic) design actions, but also a contingent process of evolution. In general, a design process is series of activities to find a solution to a problem. Changes in the design scenario and the design contexts imply changes in design processes. More specifically, in the RMD context, the design process is a purposeful arrangement of design actions to solve a design problem. It is structured as a finite set of design actions that procedurally can be arranged sequentially or parallel, depending on the needs of input data from the successive design actions. As element of a design process, a design action is a focused, but lasting effort towards a virtual elaboration of an artefact or knowledge. A design action can be identified at any point in the design process, where a set of input data is converted into knowledge or a new content. The possible arrangements and relationships among design actions create multiple design solutions. We argue that without specific knowledge and contextual information about the design process, the ARF is not

able to observe changes in the state of design actions.

These considerations led to the principal assumptions concerning the design process and design actions:

Assumption 20: ARF should have its own knowledge repository to store various domain specific bodies of knowledge about the design process and the design actions in the WPE session.

Assumption 21: Real-life design processes can be formalized based on the problem solving logic.

Assumption 22: Decision logic has a dual nature in the sense that choosing a design action is inseparable from choosing a design solution.

3.2.4 Assumptions concerning the active recommender framework

Evidentially the main role of a designer is to apply scientific knowledge to find the right solution to design problems in the course of RMD. Due to the algorithmic or big data complexity of reasoning mechanisms, it can be foreseen that designers may face a knowledge deficit while designing these mechanisms. In addition, finding an advantageous design solution may be made complicated by other issues such as (i) clarity of the definition of the problems to be solved, (ii) uncertainty of the design procedures, (iii) lack of analysis capabilities needed for the design process, and/or (iv) the choice of decisions on multiple objectives or multiple alternatives. In some cases, a designer may need the assistance of an expert analyst when unexpected difficulties occur during the analysis processes, and when interpreting the results. The considerations mentioned below implied the following assumptions concerning the ARF:

Assumption 23: The ARF should have sufficient synthetic knowledge to perform the process monitoring tasks and to generate context sensitive recommendations.

Assumption 24: The ARF should be able to provide services for (i) process monitoring, (ii) process management, (iii) recommendation generation and should do it continuously in harmony with the individual phases of RM designing scenario.

The overall conceptualization is driven by the notion of service-provisioning systems. The ARF provides recommendation services to support a designer in solving a design problem. The ARF needs to be built around a specific working scenario that has been derived from a hypothesized process model of designing a family of not purely logical or artificial intelligence-based reasoning mechanisms. As a means of depicting the support provided by the ARF the concept of (recommendation) services was utilized. It is shown graphically in Figure 3.2.

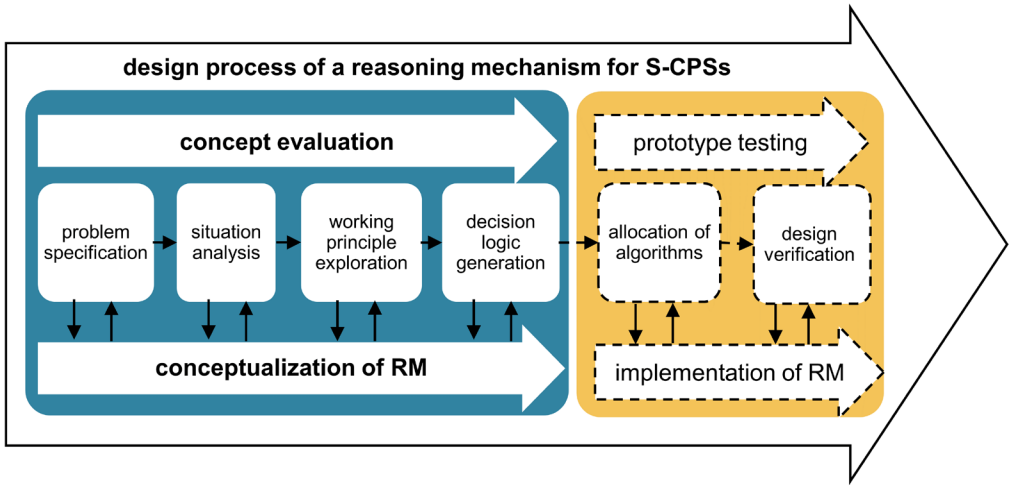


Figure 3.2: The schematized process of designing ASRMs

3.2.5 Generic and specific services provided by the proposed active recommender framework

As an intellectualized software system, the ARF is a composition of various mechanisms,which provide service packages - dedicated to the various stages of the design process - for the designers of ASRMs. In our case the ASRMs are those constituting the intellectualized constituents of the APAS software system. An allembracingrepresentation of the fundamental concepts underlying the developed ARF is shown in Figure 3.3. The left upper quadrant of this figure specifies the service packages that are associated with the concerned stages of the systematized RMD process as well as the design process dependent service packages. The right upper quadrant shows the component services included in these service packages and those stages of the implementation of the RM that are not supported by the ARF at all. The lower quadrant specifies the service packages that include design process independent services, while the right lower quadrant shows the component services included in these service packages. This conceptualization provided the basis for the architectural specification, in which there was one mechanism assigned to each service package, and a respective number of functional modules were assigned to each module.

One set of services of the ARF is related to process monitoring. This captures information on (i) the actions of the designer(s), (ii) the application of computer aided design tools, and/or (iii) the use of communication means. Another set of services concerns the support of actual design problem solving activities and the related decision making. The services for the other stages can be developed using the same principles.

Based on the above, the advisory services were sorted into two categories, namely: (i) process related recommendations, which deal with information and knowledge deficits within the information process, and (ii) content related recommendations, which

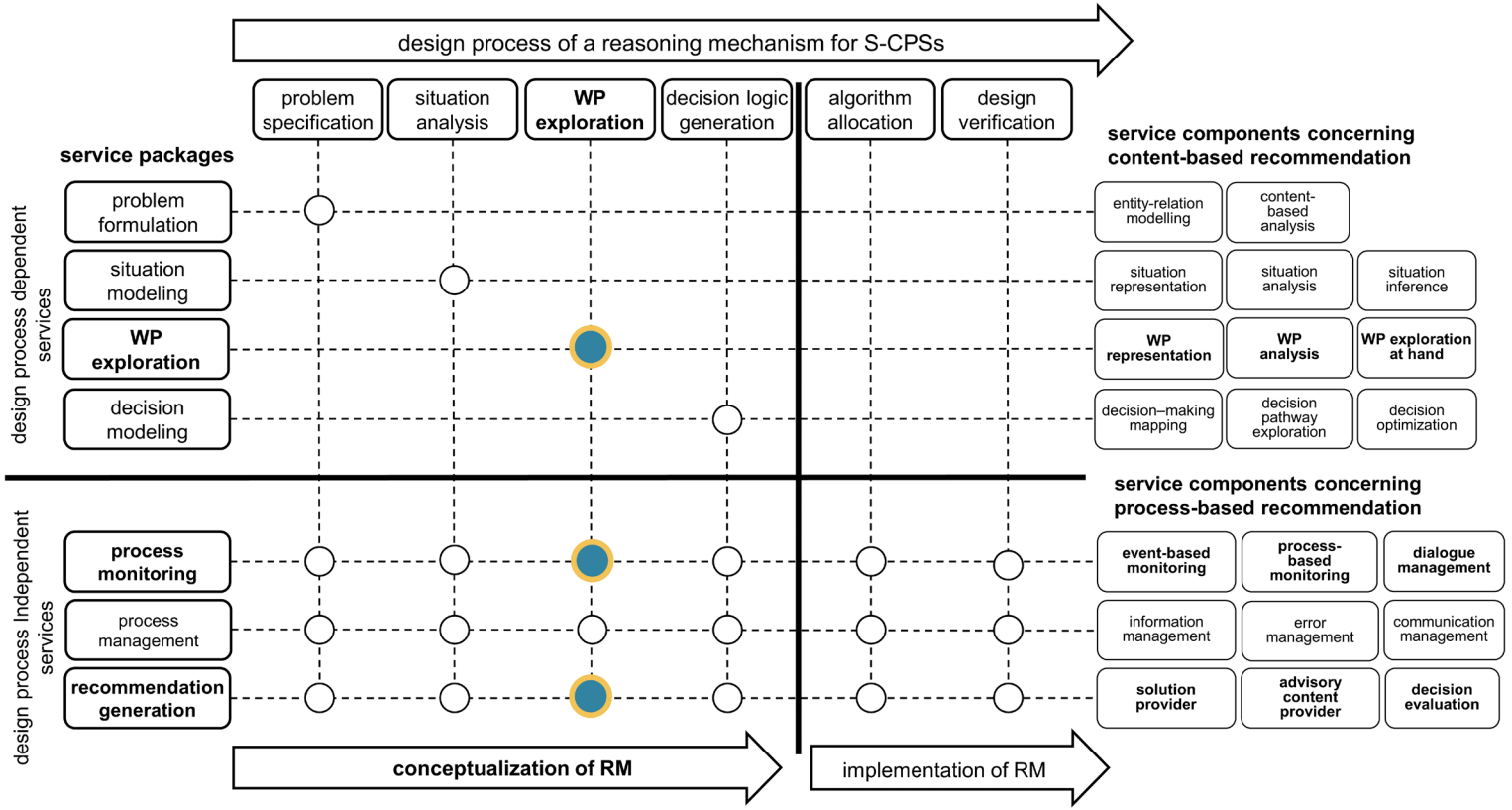


Figure 3.3: Relationships among the stages of the ASRM design process and the provided service packages and component services provided by the ARF

concern the development of the technical contents of ASRM algorithms. Content-based services were defined to support the development of contents for the identified design sessions, which are: (i)problem formulation (PF), (ii) situation modelling (SitM), (iii) working principle exploration (WPE), and (iv) decision modelling (DeM).

A problem formation is a process of identifying relevant information including objectives, constraints, assumptions, and contextual information and systematically organizing this information in the design space to determine a design problem. All of possible parameters should be considered as potential design variables. This allows for analyzing different possibilities to see the problem in a different context. A situation model is a logical representation of the interrelationship of the entities included in the defined problem. A situation can be inferred by the integrating and abstracting information about multiple events according to certain defined rules. In the situation modelling session, the ARF provides content-based recommendation for situation representation, situation analysis, and situation inference.

For example, to represent a situation, the ARF provides a set of rules concerning how to separate a continuous stream of multiple situations of the real-life problem into a set of separated single situations, or to advise the designer on this segmentation. When the parking situation is sufficiently known, the next design session concentrates on the tasks of WPE. In order to solve the WPE problem, several procedural rules are to be operationalized. In general, it is possible to find multiple working principles for a particular situation. Thus, to find an optimal working principle, the possibilities are to be converted into a decision model, which can then be used in the computer supported decision making session. In general, decision-making is a cognitive process of finding the best option from multiple choices.

The decision model was constructed in the form of a decision logic diagram. It can be seen as a network of decision points and actions. This representation allows the reasoning mechanisms to solve the particular problem. The optimal path of the network is selected as the best parking principle (strategy) in the course of making decision. In the circumscribed process, all conceptual elements will be converted into computational elements with a view to possible implementation of the reasoning mechanism.

For the second categories of services, which are process-based services, three packages were conceived. These are: (i) process monitoring, (ii) process management, and (iii) recommendation generation. Process monitoring includes three service components, namely: (a) event-based monitoring, (b) design process-based monitoring, and (c) dialogue management. Process management deals with capturing the information flow related to using the computer-aided software tool (more precisely, from the perspective of software execution). It offers three service components including (a) information management, (b) error management, and (c) communication management. Recommendation generation concerns three service components that are (a) solution provider, (b) advisory content provider, and (c) decision evaluation. In the conceptualization of an ARF, the focus of our

research was placed on those specific process monitoring and recommendation generation services, which are applied in the WPE session of the APAS development.

3.2.6 Goal of conceptualization

In the form of the ARF, we propose a knowledge-intensive system equipped with specialized process monitoring and design-support capabilities. As discussed above,these are the fundamental concepts of the ARF. The conceptualization was considered as the process of transforming the idea of the ARF into the testable concept. It should ensure that the technical specification satisfies all requirements. Thereby, the goal of conceptualization was to generate a testable blueprint of a limited, demonstrative part of the ARF which can support a particular stage of the development of ASRMs.To achieve this goal, two content domains were considered: (i) the development process,which included a high-level conceptualization of the ARF; and (ii) the technical contents, which are to demonstrate the concept in the application context.Domain knowledge about the design process of an APAS had to be explicitly specified. However, it had to deal not only with the provisioning of the technical contents, but also the changes in the design process. Capturing the procedural, content, and context changes make the knowledge contents dynamic.

The ARF operates as a smart assistant and, as such, it represents a specific subclass of context-aware recommender systems. The ARF has been conceptualized to be capable to communicate interactively with the designers and to participate in the execution of a concerned part of the design process whenever and wherever it is needed. The interoperation of two abovementioned mechanisms was needed in order to provide the required recommendation services. The process monitoring mechanism observes the design process in real-time and the decision support mechanism provides contextsensitive recommendation to solve the design problem at runtime. To demonstrate how the proposed concept works, it was tested in the application context. Towards this end, the WPE design session of the design process of an APAS was elaborated upon based on the set-up design scenarios.

3.3 Setting up a design scenario for an application- specific reasoning mechanism

3.3.1 Automated parking assist system – A practical case requiring application-specific reasoning

An automated parking assist system (ASAP) is a sub-class of driver assistant systems (e.g., collision avoidance assist system, maneuvering assist system, and adaptive cruise assist systems). It aims to offer the comfort and safety of parking for drivers. The range of the implementation of automatic operations of an APAS vary from information systems (e.g., solely distance control or parking space measuring) to fully autonomous parking assists (automated steering and speed control). To safely park a vehicle in a confined space, an APAS must perform numerous complicated tasks subject to various constraints including environment detection, steering, acceleration, braking, and gear shifting while moving the

car. It generally employs acoustic and/or radar distance sensors that detect the presence or absence of other vehicles and obstacles. This is a necessary condition to find a parking space and to complete safe parking maneuver. In line with the latest trends in vision-based technology, it can employ imaging sensors that are able to visualize the parking environment (for the driver/passenger).

The reasoning procedure for parking of an APAS starts with detecting obstacles in theenvironment of the concerned vehicle and finding a suitable parking space. Should the best parking space be found, it assesses the best approach of getting into that space by considering variables such as (i) the position and orientation of the vehicle, (ii) the direction of the parking lot, (iii) the planned motion path, and (iv) the capabilities of the vehicle to be parked. The reference motion path should be planned before performing the parking maneuvers. Then, the system navigates forward to reach a ready-to-park position. Finally, the parking maneuvers are executed by continuously estimating the changes in the state of the vehicle and by tracking the planned motion path as accurately as possible. These tasks are finished by interconnected sub-systems which are - as an example - shown in Figure 3.4.

The planning is the most crucial task in the reasoning process of an APAS. Initially, APASs were developed as semi-automated systems to park the vehicle for parallel parking, but recently systems for perpendicular and angled parking became available commercially. Many methods to tackle the motion path planning problem have been presented in the literature. They can typically be divided into two categories based on [3]: (i) stabilization of the vehicle to a target point, and (ii) planning a feasible path that connects initial and goal configurations. Most of the conventional methods carry out the path planning in two phases

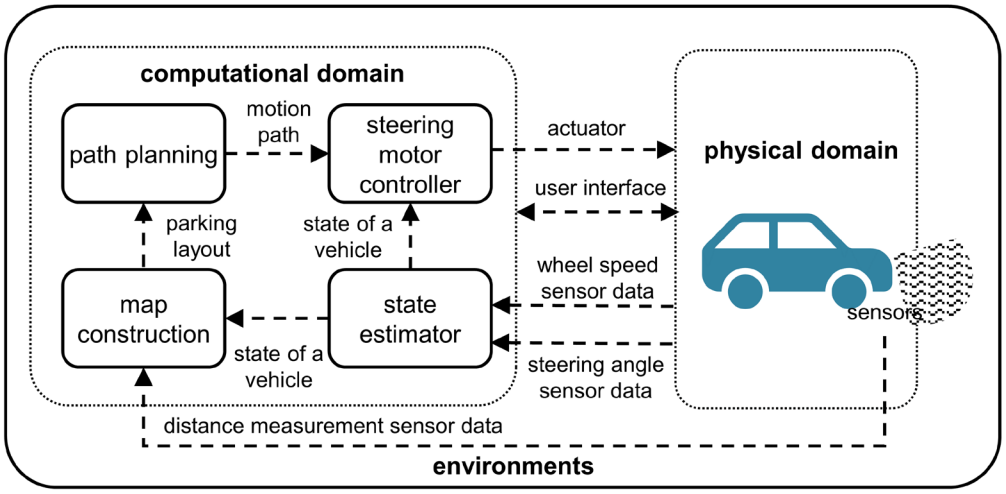


Figure 3.4: Components of an automated parking assist system (adapted from [2])

[4]: (i) planning a geometric collision-free path without taking non-holonomic constraints of the vehicle into account, and (ii) performing sub-divisions on the path until all end points can be linked to their neighbors by an admissible collision-free path using a local planner.

The challenge in the development of the reasoning mechanism of an APAS is that it typically consists of multiple processes in different levels of thinking which works not only in geometric domain but also works in logical domain. Within the latter domain, a designer has to deal with intangible elements such as knowledge, algorithms and rationality based on the logic. A smart assistant for complex real-life task needs specialized reasoning which manifest in multiple various ways. A contextualized reasoning mechanism for an APAS is one of the examples. The following sub-sections set up a design scenario in particular WPE design session to demonstrate the concept and feasibility of the proposed ARF.

3.3.2 Procedural and computational implementation of working principle exploration

The stage chosen for technical elaboration in this research cycle is working principle exploration (WPE). In other words, it means finding the most effective motion trajectory for the car to be parked, considering a low level of maneuvering activities by the car and the maximum safety and reliability that can be achieved at all. There are several reasons behind this choice. The first reason why this stage was selected for detailing is that it is rich in design activities, which in turn raise the need for a sophisticated process monitoring. The second reason is that this stage implies the need for several services from the ARF to support process management. Actually, a remarkable complexity is faced when we consider the total number of support services needed to complete this stage and the number of computational functions needed to implement the related activities. The third reason is that it involves many decision points, which are purpose and context dependent.

In the process of developing a multi-mechanism solution for automated car parking, a designer must find or develop relatively most effective working principle for a given parking situation. For example, if forward parking is selected as a working principle, it means that there are various sets of maneuverings that satisfy the parking. The term '*working principle exploration*' (WPE) is used to denote the procedure of construction of the motion plan and segmentation of the maneuvering activities. WPE considers the actual parking opportunity and situation, and provides information about the best trajectory (motion path of the car) to be followed and the physical parameters of maneuvering (speed, acceleration, ranges, stopping points, etc.).

For conducting the WPE design session, it is assumed that the parking situation has been analyzed and understood from spatial, temporal, maneuvering, technical, etc. points of view in the session focusing on situation analysis. A so-called 'situation blueprint' is generated, which provides information about the arrangement of the physical entities in the vicinity of the parking spot, including the layout of parking zone, the natural objects, the number

of neighboring parking cars, the spatial arrangement of parking cars, the geometry of the space that can be used for parking, the position of the car to be parked, the geometric sizes of the car to be parked, and the type (maneuvering capabilities) of the car to be parked. As well as, it is assumed to have information about the position and nature of possible (incidental) objects, if any.

Based on this input information and the expected output of WPE session, a designer can establish its own reasoning process. It should be kept in mind that this reasoning process will be implemented in the ASRM for an APAS. Thus, it should confirm that the process which works (semi-) manually at the time of design will work automatically and function correctly at runtime. Concerning the support of the designer by availing knowledge for modelling a reasoning process, our assumption is that an optimal motion path for parking is found based on the comparison of morphological structures of a current parking plan and past parking cases. As shown in Figure 3.5, the reasoning process consists of transformations of information in six sub-processes. It is seen as a reference model which corresponds to a design process.

3.3.3 Specification of the design tasks for working principle exploration

Basically, the major design activity of the reasoning mechanism development is the construction of algorithms for different computational problem-solving purposes. In the multi-stage process of reasoning mechanism design and in the process of implementation, the specific algorithms are integrated. A design process itself is regarded as a sequence of thinking-type and doing-type design actions, which are computationally interpreted as transformative events. A design task represents an element of the design process that a designer should complete in order to realize the process. In other words, the realization of the reasoning process is decomposed into a set of reasoning sub-processes - each of which is further decomposed into a sequence of design tasks. Using symbolic expressions, the relationships of the reasoning (sub-)processes and the design tasks can be formally described as follows:

$$RsP_i := \{d_1, d_2, d_3, \ldots d_k\} \tag{3-1}$$

where: RsP_i is a sub-process i^{th} of reasoning process $RsP_i \in RP$, $i \in N_s$, N_s is total number of sub-processes, $d_1, d_2, d_3, \ldots d_k \in \mathfrak{D}_i$ is a sequence of design tasks belonging to RsP_i and k is total number of design tasks for a completion of RsP_i.

In the case of each reasoning sub-process, an essential design task is to construct the algorithms, which together can solve the problem of exploring the working principle (e.g., finding a motion path, or reducing the maneuvers) for parking in a real-life situation. The exploration of the working principle involves the completion of eight design tasks, which are shown in Figure 3.6, as elements of a workflow diagram. To accomplish any of these tasks requires one or more algorithms for the ARSM. These design tasks should be

completed at the time of design in order to assure that the reasoning process will work (automatically and without any deficiency) at runtime.

The description of design tasks are as follows:

- $\mathfrak{D}_{1.0}$ Construction of an algorithm for searching for and retrieving past parking cases from the repository of the APAS concerning the actual parking situation (ASRM-Algorithm A_{01})
- $\mathfrak{D}_{2.0}$ Construction of an algorithm for extracting the morphological (topological, geometric, attributive) information structures from the N-appropriate parking cases (ASRM-Algorithm A_{02}).
- $\mathfrak{D}_{3.0}$ Construction of an algorithm for comparing the topological sub-structures of the motion paths of the past parking cases and selecting the most advantageous topological sub-structures (ASRM - Algorithm A_{03}).
- $\mathfrak{D}_{4.0}$ Construction of an algorithm for calculation of the 2D spatial position and distances (between the entities) information structure (geometry) of the parking plan (ASRM-Algorithm A_{04})
- $\mathfrak{D}_{5.0}$ Construction of an algorithm for determining the geometric resemblance by comparing the relevant geometric sub-structures of the selected (topologically best matching) past parking cases with the geometric sub-structure of the parking plan, and selecting the most affine geometric sub-structure of the selected past parking cases (ASRM-Algorithm A_{05})

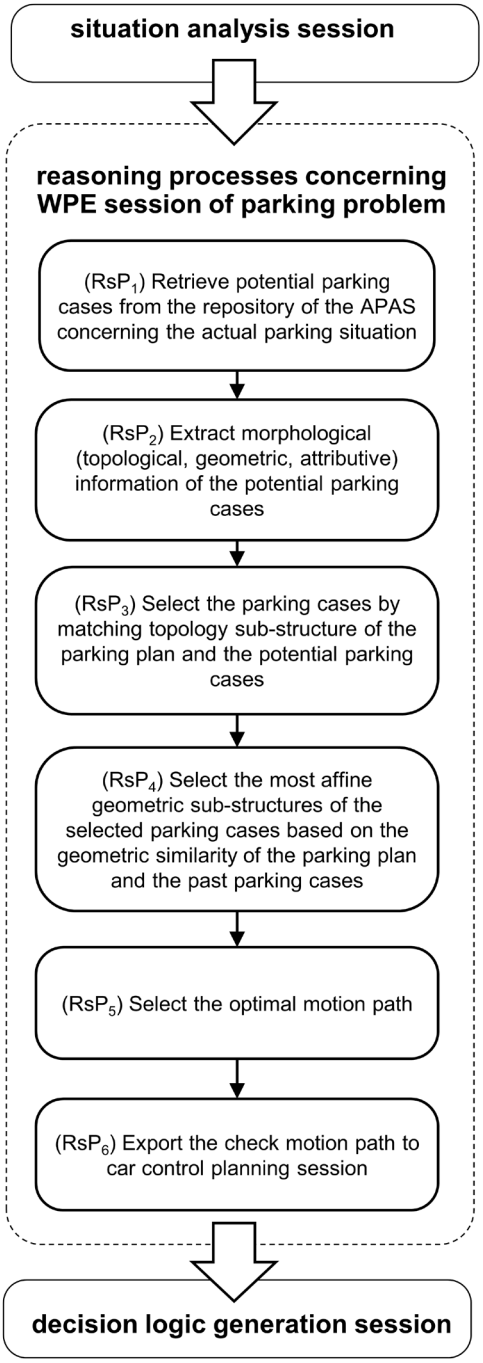


Figure 3.5: Reasoning process concerning WPE session of the APAS

$\mathfrak{D}_{6.0}$ Construction of an algorithm for extraction of the attributive information sub-structure (size information of the entities) of the parking plan (ASRM-Algorithm A_{06})

$\mathfrak{D}_{7.0}$ Construction of an algorithm for determining the attributive resemblance by comparing the relevant attributive sub-structures of the selected (geometrically best matching) past parking cases with the attributive sub-structure of the parking plan, and selecting the optimally-matching motion path of the reduced past parking cases (ASRM-Algorithm A_{07})

$\mathfrak{D}_{8.0}$ Construction of an algorithm for adapting the optimally-matching motion path of the reduced past parking cases to the parking plan by considering the attributive information sub-structure of the parking plan and providing a description of the adapted optimally-matching motion path cases (ASRM-Algorithm A_{08})

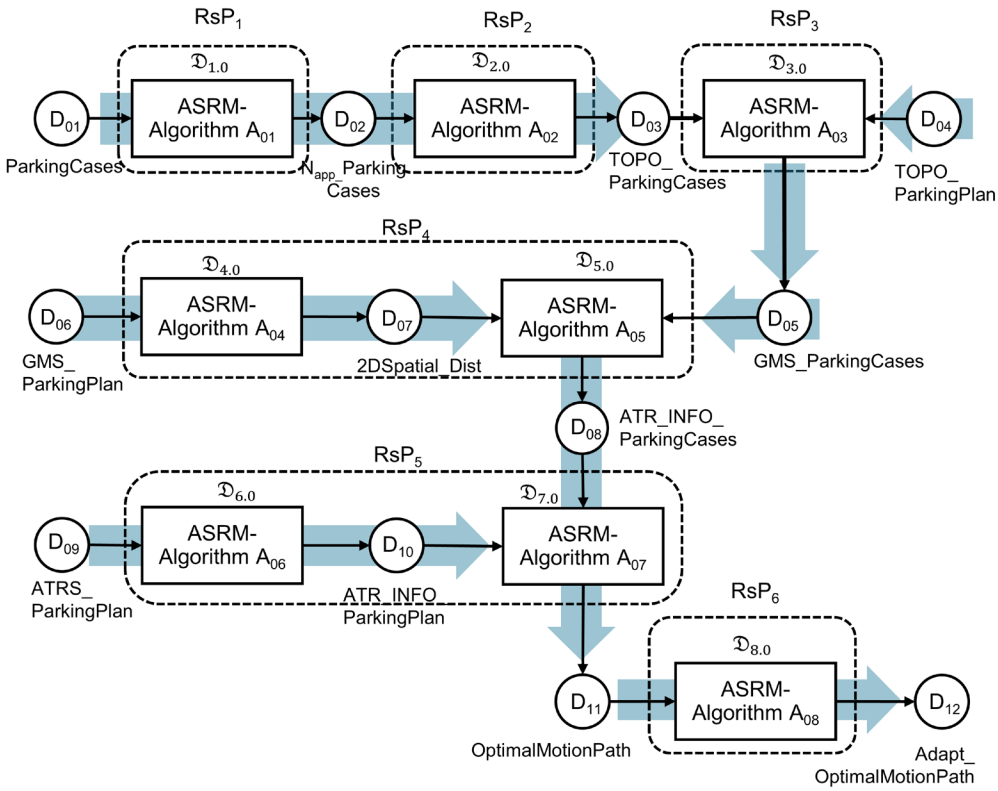


Figure 3.6: Workflow diagram identifying the design tasks needed to accomplish the exploration of a proper working principle for a parking problem

3.4 Fundamentals of conceptualization of the active recommender framework

3.4.1 On the duality of the active recommender framework development

From the system design engineering point of view, the duality of the ARF development means that, on the one hand, it supports the development of the application-specific reasoning mechanisms and algorithms by offering recommendations for making correct design decisions. On the other hand, it supports the design actions made by the designers with continuous process monitoring and resolving obstacles in the design process by making procedural recommendations. More specifically, the ARF is characterized by two main capabilities, process monitoring and decision-support capabilities which offers a context sensitive recommendation to a designer. Hence, the manifestation of the ARF is done by the interoperation of two mechanisms, process monitoring and decision-support mechanisms. The process monitoring mechanism is responsible for observing the behavior of a designer through the entire design process and recognizing when the designer has a problem. The decision-support mechanism provides knowledge and recommendations to resolve the problem.

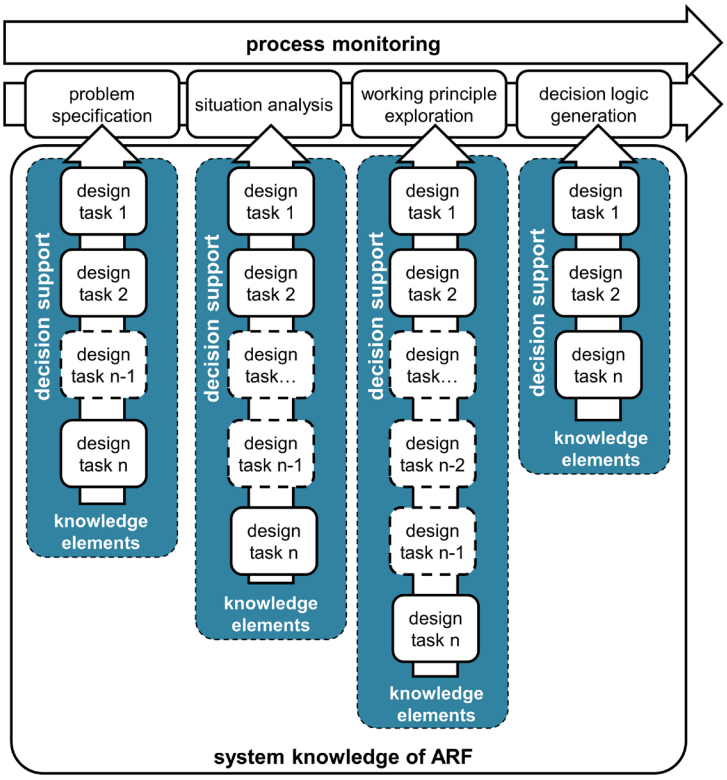


Figure 3.7: Duality of the ARF development

From the knowledge engineering point of view, it concentrates on creating knowledge for the ARF. In the context of ASRMs, the knowledge is about the enablers of system-level reasoning of S-CPSs. To develop an ARF, the domain-specific knowledge should be available when it is needed at any stages of designing ASRMs. This is considered as the duality of ARF development in term of the resembling knowledge of ARF and ARSMs. It means that knowledge in the ARF will be transformed into knowledge for the reasoning process of ASRMs.

Considering the design task, it is a knowledge element of ARSMs to construct the algorithms which together to solve the real-life problem. For the ARF, it is supposed to know what is needed for designing these algorithms and include these knowledge elements into the ARF. Figure 3.7 illustrates the duality of ARF development. The design process of ASRMs can be seen as the knowledge modelling for the development of ARF.

3.4.2 Event management related to design actions by the active recommender framework

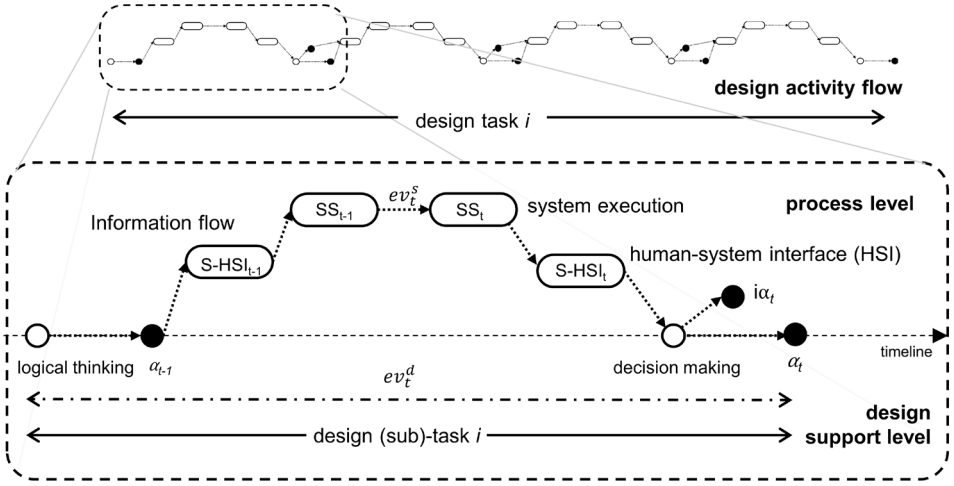
The major role of an ARF is to support a designer by making a recommendation when achieving the purpose of a design process is blocked (obstructed) for any procedural, informational, cognitive, or technical reason. To do this, the ARF observes an event which is the changes in the state of design process and reasons in context what a designer is doing at that moment. Theoretically, an event is derived by learning the differences of two subsequent states of an entity at different points in time and location with a combination of context information [5]. In order to monitor an event in the design process, two approaches are taken into a consideration. A simplified schematic diagram can be seen in Figure 3.8. One is a process-based monitoring which can be detected and monitored process flow within the observed system. Another is an activity-based monitoring which is observed in the changes of a designer's behavior.

A process-related event is captured by a change in the states of information flow in a system. For example, in the typical business process model, an event is detected by changes of the state of control flow and data flow [6]. This type of event can be inferred by collecting software execution data from log file [7]. Symbolically:

$$ev^s_t = \Delta(s^s_t , s^s_{(t+n)}) \tag{3-2}$$

where: ev^s_t represents a process-related event, which is detected by the deviation of state s^s_t and state $s^s_{(t+n)}$ in an interval of time t and $t+n$, and $n > 0$.

An activity-based event is detected and monitored by considering the changes in a designer's behavior during the design process. This type of event can be reflected in their body movements, for instance facial expressions, and eye movement [8]. When the changes in the two different states in behaviors are detected at a given moment in time, it could infer that an event in the design process is occurred. Symbolically:



Legend: ev_t^s = process-related event; ev_t^{act} = activity-based event; S-HSI = state of HSI; SS = state of system; α = design action; iα = design-independent action

Figure 3.8: Simplified schematic diagram representing the event-based monitoring throughout the interaction of designer's activities and system execution

$$ev^d_t = \Delta(s^d_t, s^d_{(t+n)}) \tag{3-3}$$

where: ev^d_t is an activity-based event, s^d_t and $s^d_{(t+n)}$ are designer behaviors, which are observable in an interval of time t and $t+n$, and n > 0.

For the conceptualized ARF, an event management deals with the latter approach because the event is inferred to the cognitive process of a designer during the design process. Since an event has been detected, inferring the event with a context of a design process of ASRM will give a semantic meaning related to a design action. Symbolically:

$$\alpha_t = infer(ev^d_t) \tag{3-4}$$

where: α_t is a design action, which is inferred according to an event ev^d_t.

To monitor the design process, we classify an event into two main types, namely: (i) a usual event, which refers to a regular completion of the design action as expected by the design protocol, and (ii) a non-usual event, which refers to the fact that the completion of a design action deviated from the expectation of the design process protocol. For example, when a designer cannot progress in the design process (is procedurally, informationally, cognitively, or technically blocked with a design action), an event happens which indicated that the state of the designer is changing differently from what would be expected based on the design process. If there is an unusual event captured in whatever way, then decision

on the type of recommendation should be made by the ARF and the actual content should be constructed accordingly. As an option, the designer busy with a design action may ask support from the ARF by making a note/query. This is an unusual event, which is recognized by a designer. The ARF can contribute to supporting the designer at performing the design process in various ways, which are discussed in the Section 3.4.3. The different ways of contribution by the ARF are regarded as the fundamental principles for conceptualization of the process monitoring mechanism of the ARF.

3.4.3 Typifying the ways of observation of non-usual events

The occurrence of an unusual event (a non-usual event – NUE) may become known in three ways: (i) based on the observation of the designer (Type A), (ii) based on an interaction between the ARF and the designer, and (iii) based on the investigation of the ARF (Type E). Since the designer and the ARF may have different (extents of) contribution to the observation and handling of the occurrence of an NUE related to the execution or results of a design action a NUE, three further cooperative types (Types B, C and D) have been distinguished. They are shown in Figure 3.9 and interpreted below.

Type A: An NUE is suspected or observed by the designer himself (that is, he recognizes that he cannot start (input issue), proceed (conduct issue), or complete (output issue) the design action at hand without the support of the ARF and initiates a dialogue with it towards a resolution of the situation.

Type B: An NUE is suspected or observed based on a smaller contribution of the ARF and a larger contribution of the designer (that is, the ARF observes some change in the designer's behaviors e.g., change of body posture or face expression) and initiates a dialogue with the designer towards a resolution of the situation.

Type C: An NUE is suspected or observed based on a balanced (equal) contribution of the ARF and the designer (that is, the ARF recognizes something odd (e.g., a decisional error) and notifies the designer about it, and if the designer acknowledges it, then it initiates a dialogue with the designer towards a resolution of the situation.

Type D: An NUE is suspected or observed based on a larger contribution of the ARF and a smaller contribution of the designer (that is, based on the reference protocol, the ARF observes a deviation from it and makes a recommendation based on the available knowledge (e.g., offering a list of design tools, design methods, or related resources, but the final decision is made by the designer.

Type E: An NUE is suspected or observed by the ARF, which automatically makes a decision or execute a design action on behalf of the designer, without his involvement.

3.4.4 Recognition of a non-usual event

The fundamental of event recognition of the conceptualized ARF is based on an observation of the facial expressions of the involved designer. This approach was underpinned by the assumption that the designer will express the certain pattern of emotions on his face when

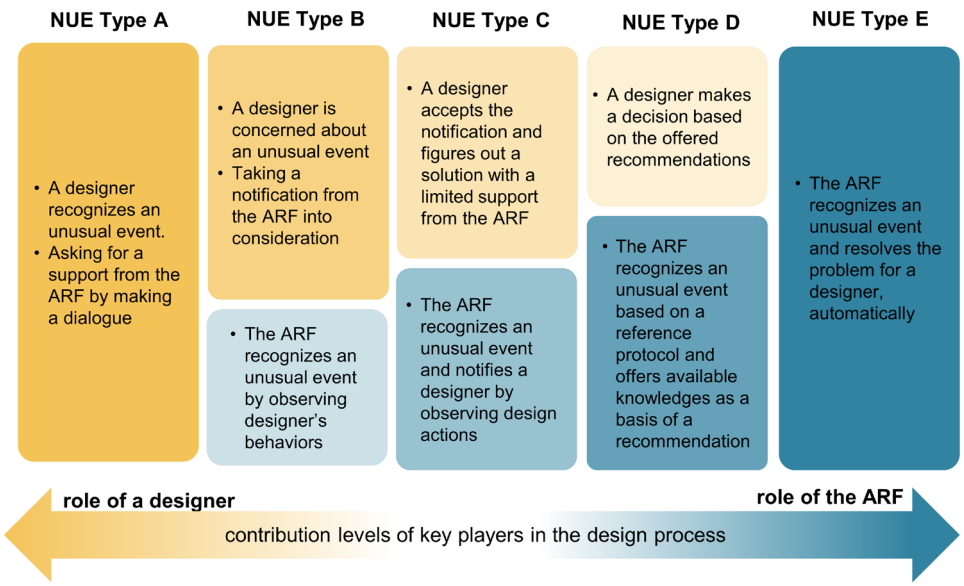


Figure 3.9: Contribution of the ARF and a designer in an execution of design process of RMD

he has a problem in the design process. There were several publications on this phenomenon in the literature [8]. The observation of the facial changes is the starting point for the ARF to interact with a designer. A recognition of facial expression is the process of identifying human mental status from the expressions and cognition has human on his face [9].

In general, the process of facial expression recognition consists of three main stages [10]: (i) face detection; (ii) facial feature landmark extraction; and (iii) facial expression recognition (FER) as shown in Figure 3.10. Face detection is the process that detecting human's face through images. It typically focuses on deviating features of human faces from images without recognizing an individual person. The most widely used algorithm for face detection is Viola and Jones Haar-cascade [11]. Recently, commercial software (e.g., Amazon Recognition, Face++, and Azura Cognition Services Face API) as well as open sources (e.g., Deepface, FaceNet, and InsightFace) have been developed for face detection.

Within the detected face, facial landmarks for instance the eyes and eye corners, brows, mouth corners, and nose tips are detected. The internal face model is justified in position, size, and scale in order to match to the actual face. Feature landmark extraction is the process of extracting key information from the detected face as a basis of facial recognition. The feature extraction is usually done by one of two approaches [12]: (i) processing the whole frontal face in order to collect information for classifications of facial expressions or, (ii) dividing the whole face image into subordinate sections and processing each of them to get information that can be used as classification input. Once a simplified face model is available,

the key features are selected including position and orientation information is fed as input for the feature classification. The machine learning classifiers are commonly used for this purpose e.g., multi-layer perceptron, SVM, Naïve Bayes, and k-NN.

Convolutional neural network (CNN) is the state of the art in the deep learning algorithms for FER [13]. To train the facial expression recognition algorithms, two types of input images are used. One type is the learning image using which classifiers are trained to recognize any pattern. Another one is input image on which the learned classifier is tested. The collection of images can be taken from various databases for instance Google Face dataset, CASIA-Webface, and Labelled face on the wild (LFW) database. The facial expressions are classified into eight categories, which include: (i) anger, (ii) disgust, (iii) fear, (iv) happiness, (v) contempt, (vi) neutral, (vii) sadness, and (viii) surprise. In addition, there has been a proliferation of commercial tools for FER solution available in the software market. In [14], the authors compared the performances of eight commercial software tools (for instance CrowdEmotion, MorphCast, and Human Observers).

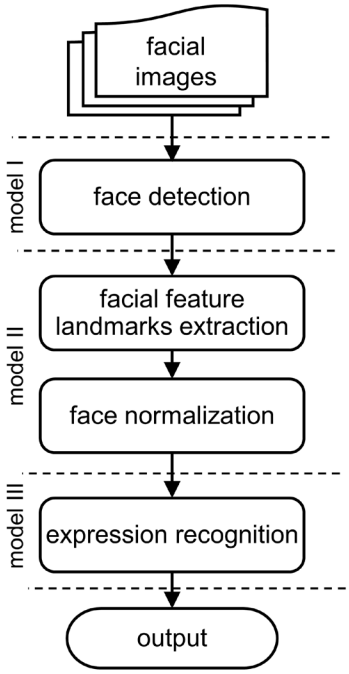


Figure 3.10: Facial expression recognition process [10]

3.4.5 Interaction of the active recommender framework and the designer in the targeted segment of the design process

The fundamental for a conceptualization of this part was using a human-machine interaction method through a dialogue. This aims at exploring the possibility of solving a problem in a design process which a designer may not have recognized by themselves. A dialogue is activated when an unusual event is detected. It starts with inviting a designer to a dialogue by offering help. If the designer accepts this offer, the ARF possess a question what design action a designer is working on. With domain-specific knowledge stored in the repository, the ARF contextualizes the available knowledge with the considered design process. When the current design action is recognized, the information including a set of questions, decision table, and contents of design action is retrieved. The set of questions is designed related to this particular design action which is about trying out the given search program. The dialogue will be organized to collect the current states of design action. The flow of questions is controlled based on pre-defined orders according to a certain design

action. The ARF communicates with a designer in textual or verbal forms (in the latter case, using natural language processing) and captures the patterns of replied answers by a designer.

Table 3.1: Simplified structure of a decision table

contents	pattern 1	pattern 2	pattern n
criteria 1	yes	no	yes
criteria 2	no	yes	yes
criteria 3	no	yes	no
criteria 4	yes	no	no
solution A	x		
solution B			x
solution C		x	

The goal of the interaction process between the ARF and the designer is to find out the solution to eliminate the obstacle at the current design action. The solutions were defined according to the patterns of decision criteria used in the historical cases as shown in Table 3.1. If the best match between the pattern of replied answers and the decision criteria in the decision table is found, the corresponding solution would be retrieved. However, it might be the case that the matched pattern could not be found. It can be assumed that the cause of obstacle would be occurred in one of the preceding design actions. This implies the need to investigate the procedural obstacle in the design process using the reference process protocol.

3.4.6 Principle definitions of a reference protocol and its constituents

The protocol of the design process (e.g., of all design actions) is supposed to be known by the ARF in the form of a computational model, or part thereof, called reference process protocol (RPP). On the one hand, the design protocol is seen as the basis for the design guideline by which the ARF provides information and instruction for the designers concerning the approach of dealing with the entire set of the design tasks and the included individual design tasks, at the beginning of the design session. This will not be elaborated in our work. On the other hand, the design protocol serves as a reference process protocol for the ARF to monitor the design actions done by the designer as well as the results of the completed design actions in the process of designing. If the observed execution of the design actions or the data processed by the executed design actions deviate from what is captured in the reference process protocol, the ARF may infer about the occurrence of an unusual event.

The principled definitions of a reference protocol and its constituents will be in this section as follows:

Definition A: *Reference process protocol (RPP)*

A reference process protocol is a prescriptive instrumental model of a design process, or part thereof, established by three constituents, formally:

$$RPP = \{TAM, PFM, DTM\} \tag{3-5}$$

where: TAM is a timed (design) actions model, PFM is a process flow model, and DTM is a decision tree model. From an information engineering point of view:

$$RPP = TAM \cup PFM \cup DTM \tag{3-6}$$

where: ∪ is symbolic union. The RPP is a network of design actions and a set of decision points. A protocol is a system of rules or procedures that allow two or more design actions creates their relationships. It captures the pattern of the design processes and their expected outcomes.

Definition B: *Timed action model (TAM)*

A timed action model is a finite non-zero set of design actions together with their input and output (parameters) variables. A TAM is represented as a split matrix, whose half matrices are arranged according to the temporal sequence of design actions, and captures the input parameters and output parameters (computational variables) with regard to each design action. Formally:

$$TAM = \{\alpha, t_s, t_e, m(\alpha P_{(in,m)}, \alpha P_{(out,m)})\} \tag{3-7}$$

where: DA is a finite, non-zero set of design actions, t_s is the start point in time of $\alpha_i \in DA$, t_e is the end point in time of α_i, m is a finite, non-zero set of the alternative methods of computational execution of α_i, $\alpha P_{(in,m)}$ is a finite, non-zero set of input design parameters related to a particular m, and $\alpha P_{(out,m)}$ is a finite, non-zero set of output design parameters related to a particular m.

Definition C: *Process flow model (PFM)*

A process flow model (PFM) is a state-transition model of design process, including a finite, non-zero set of the process flow elements representing the design actions as computational transitions and the input and output states of the design actions represented by the (evaluated) values of the design parameters (variables).

For the purpose of computation, a process flow model is represented as a petri net, which is a bipartite directed multigraph consisting of two different types of nodes formed by (i) the set of transitions, associated with design actions, T_α, and (ii) the set of states of design actions, S_α. With these dispositions, a process flow model (PFM) can be expressed symbolically as:

$$PFM = \{T_\alpha, S_\alpha, W, M\} \tag{3-8}$$

where: T_α is a finite, non-zero set of computational transitions related to design actions α, S_α is a finite, non-zero set of process states, W is a finite, non-zero set of relations among the elements of the set T_α, and the elements of the set S_α, represented as arcs of a bipartite directed multigraph process model, and M is a finite, non-zero set of markings. The PFM captures the pattern of the design actions and their expected outcomes.

Definition D: Decision tree model (*DTM*)

A decision tree model is defined as a computational means, which allows decision making on the relevance of a computational transition related to design action $\alpha \in DA$ and the computational method, $m \in \mathcal{M}$. Computationally, a DTM is a classifier algorithm for finding a proper method of a computational execution of design actions

$$DTM = \{w, M_{\omega}, R\} \tag{3-9}$$

where: α represents a design action, w is a finite set of decision variables, M is a finite set of potential computational methods for an execution of design action, and R is a finite set of decision rules.

3.4.7 Modelling the design activity flow by a reference process protocol

The network of design actions is a representative of a reference process protocol (RPP). In other words, it can be seen as a process model based on which a framework developer models (the part of) of the reasoning mechanism development process. In the design process of RMD, the procedural structure of the design process formed by the design actions can be captured in and modelled by the RPP. It provides a complete specification or design procedure to be followed to complete the design process. Based on the relationships of design actions in the RPP, multiple pathways can be created for a development of ASRMs.

In order to explore the possible design flows within the RPP, the causal relationships of design actions are constructed by Bayesian network (BN). It is a probabilistic graphical model which represents in a directed acyclic graph. A graph consists of a set of nodes and a set of directed edges between nodes. In context of the RPP, a node represents a design action. An edge connects nodes in the direction of influence to create a design flow. A BN plays a role in a decision support to select the best coupled design actions based on the probabilistic reasoning. It is possible that multiple usable methods are available to execute the considered design action. A decision tree model will be applied to select the best option. The probabilistic reasoning incorporates with decision tree models can infer the most informative design flow.

A graphical representation of conceptualization of an RPP is shown Figure 3.11. A rectangle box represents a process flow model. The circles and shaded/rounded rectangles in a process flow model represent a state and a transition, respectively. A dot line represents a composition relationship between two subsequent design actions. A black diamond represents a decision tree model for every decision point. A sequence of design actions in the network shows their temporal relationships. The variable p_i is a probabilistic relationship of design actions.

The design activity flow is represented by a sequence of design actions. Formally:

$$\mathbb{P} := \{\alpha_i, \alpha_j, \alpha_k, \ldots, \alpha_n\} \tag{3-10}$$

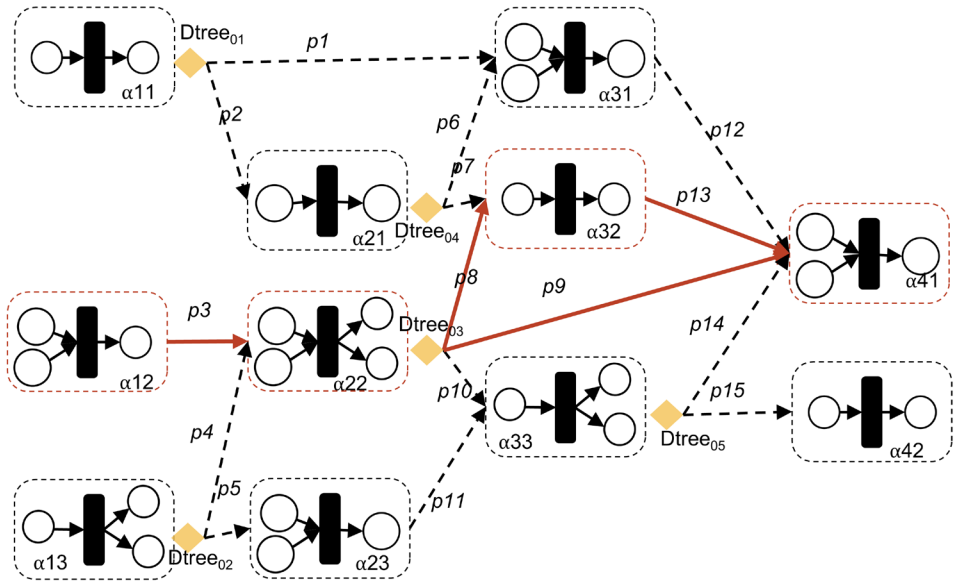


Figure 3.11: Conceptualization of a reference process protocol

where: $\mathbb{P}$ is a design activity flow, and $\alpha_i \in DA$.

3.4.8 Generation of recommendation in the case of type B observation of non-usual events

The goal of recommendation generation is to propose the most proper design activity flow which can rectify the conducted design actions and continue the design process when it is blocked. Two types of inference methods are applied for the generation of recommendations: (i) exact inference method which finds the perfect match between the pattern of designer's answers and the patterns of decision conditions which are pre-defined in the lookup table, and (ii) hybrid inference method which utilizes the RPP based on the incorporation of probabilistic reasoning and decision tree models. The generic workflow of the recommendation generation in case on type B observation of non-usual event is shown in Figure 3.12.

Since the ARF recognizes a non-usual event, the process of recommendation generation follows these steps:

Step 1: notify a non-usual event,
Step 2: identify the current design action,
Step 3: organize a dialogue,
Step 4: capture the pattern of answers replied by the designer,

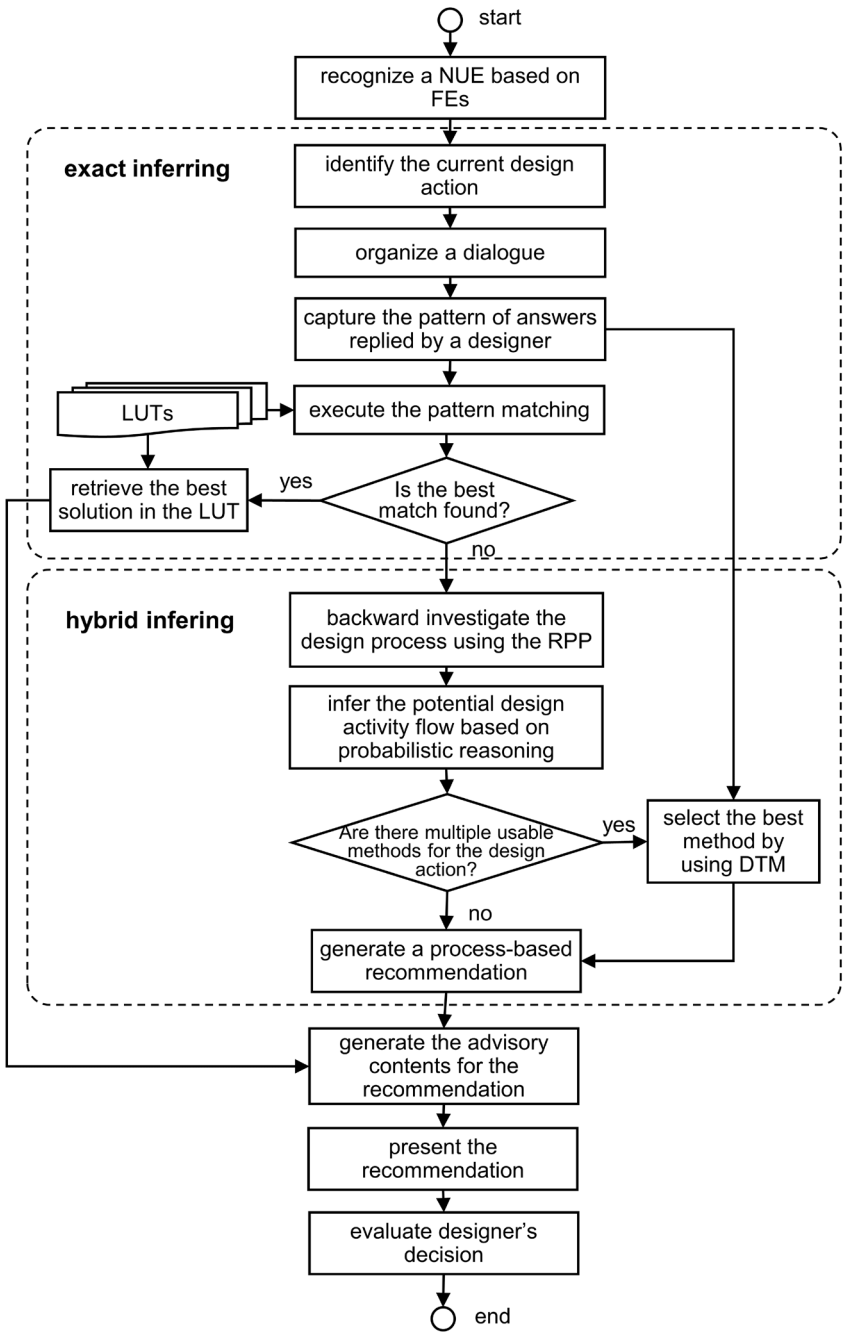


Figure 3.12: Generic workflow of the recommendation generation in the case of NUE type B

Step 5: execute the pattern matching of designer's answers and decision criteria in the lookup table, if the best match found, retrieve the best solution then go to step 10, otherwise go to the next step,

Step 6: investigate backward in the RPP to find the proper design action by using probabilistic reasoning,

Step 7: infer the potential design flow which includes (i) preceding design action, (ii) current design action, and (iii) next design action,

Step 8: select the proper method to execute the identified design action, if there are multiple choices of usable method then apply a decision tree to select the best one, otherwise go to the next step,

Step 9: generate the process-based recommendation,

Step 10: generate the advisory contents for the recommendation,

Step 11: present the recommendation,

Step 12: evaluate designer's decision on the proposed recommendation.

As a result, the process-based recommendation is constructed in form of the design activity flow which can be found in the RPP. Each of them involves the proper usable method to perform the design process. For example, the proposed process-based recommendation is shown by the sequence of design actions and their connections with the full lines in Figure 3.11.

3.5 Functional specification of the computational operations in the case of type B observation of non-usual events

The ARF is a multi-mechanism software system. The included mechanisms, which provide context-sensitive services for a designer, have specific functionalities. According to NUE type B, the computational mechanisms of the ARF are decomposed into six main functions included: (i) $F_{1.0}$ – to recognize NUE based on a designer's facial expressions; (ii) $F_{2.0}$ – to identify an obstacle at a certain design action using a dialogue; (iii) $F_{3.0}$ – to construct a reference process protocol; (iv) $F_{4.0}$ – to identify a procedural obstacle in a design process using a reference process protocol; (v) $F_{5.0}$ – to generate an advisory content; and (vi) $F_{6.0}$ – to evaluate the quality of recommendation. The decomposition of the sub-functions is shown in Figure 3.13. The requirements considered at the specification of functionalities were identified in Chapter 2. Below the specified functions are explained with more technical details.

3.5.1 Functional specification for recognition of non-usual event based on a designer's facial expression

The main function $F_{1.0}$ is to recognize a non-usual event based on the facial expression of the designer. The function was specified based on the requirements FR-M02 - the mechanism should recognize doubtful (unexpected) events in the design sessions in quasi-real time. To realize this function, the sub functions were specified according to the regular process of face recognition and the notification function.

This includes the following six sub-functions. The sub-function $F_{1.1}$ is to capture video image. It is an input sub-function, which activates a device and captures a designer's image from a video camera. The sub-function $F_{1.2}$ detects a designer's face in the video image. The captured image is preprocessed in order to highlight the face region.

Normalization and equalization were performed on the original images. Since the face region was detected, it is registered into the database. The sub-function $F_{1.3}$ is to extract facial landmark features as input for training a machine-learning algorithm. The sub-function $F_{1.4}$ applies the machine-learning model to classify the certain type of facial expressions. The sub-function $F_{1.5}$ learns the patterns of facial expressions and predicts an event. If the designer shows the pattern of facial expressions which are defined as a non-usual event, the sub-function $F_{1.6}$ concludes about a non-usual event and notify the designer about it.

3.5.3 Functional specification for a construction of reference process protocol

A reference process protocol is considered as a content-oriented model for an investigation of the intended design process. It can be seen as a network of design actions and decision points. A decision point is not only a terminal node of preceding action, but it is also a node initiates follow up actions. The objective of this function $F_{3.0}$ is to construct a reference process protocol with the composition of three constituent elements includes process flow models, a timed action model, and decision tree models. The sub-functions are related to the creation of these constituent elements and management of their knowledge.

The main function is decomposed into seven sub-functions. The sub-function $F_{3.1}$ is to create a data model representing design action. The model is a computational representation of design action called as a design entity. Each entity contains the specific information which is distinct from each other. After modelling the design actions, the sub-function $F_{3.2}$ establishes a repository to store the collect of design entities and related knowledge elements. The sub-function $F_{3.3}$ is to create a representative of process flow model. To identify the temporal relationships of the design entities, the sub-function $F_{3.4}$ creates a sequence of design tasks and assigns a design entity to the certain design task. Based on the relationships of design entities, the sub-function $F_{3.5}$ constructs a timed action model. The sub-function $F_{3.6}$ is to construct a decision tree model. Lastly, all three elements are consolidated by the sub-function $F_{3.7}$ to compose a reference process protocol.

3.5.4 Functional specification for an identification of procedural obstacle in a design process

The goal of this main function is to find the most appropriate sequence of design actions to generate the process-based recommendation. The conducted design actions can be rectified by the proposed design actions. The next design action is predicted to continue the design process. The reference process protocol is used to investigate the considered design process. To achieve this goal, the main function is decomposed into seven sub- functions. Once the current design action was identified, the sub-function $F_{4.1}$ aims at exploring the possible

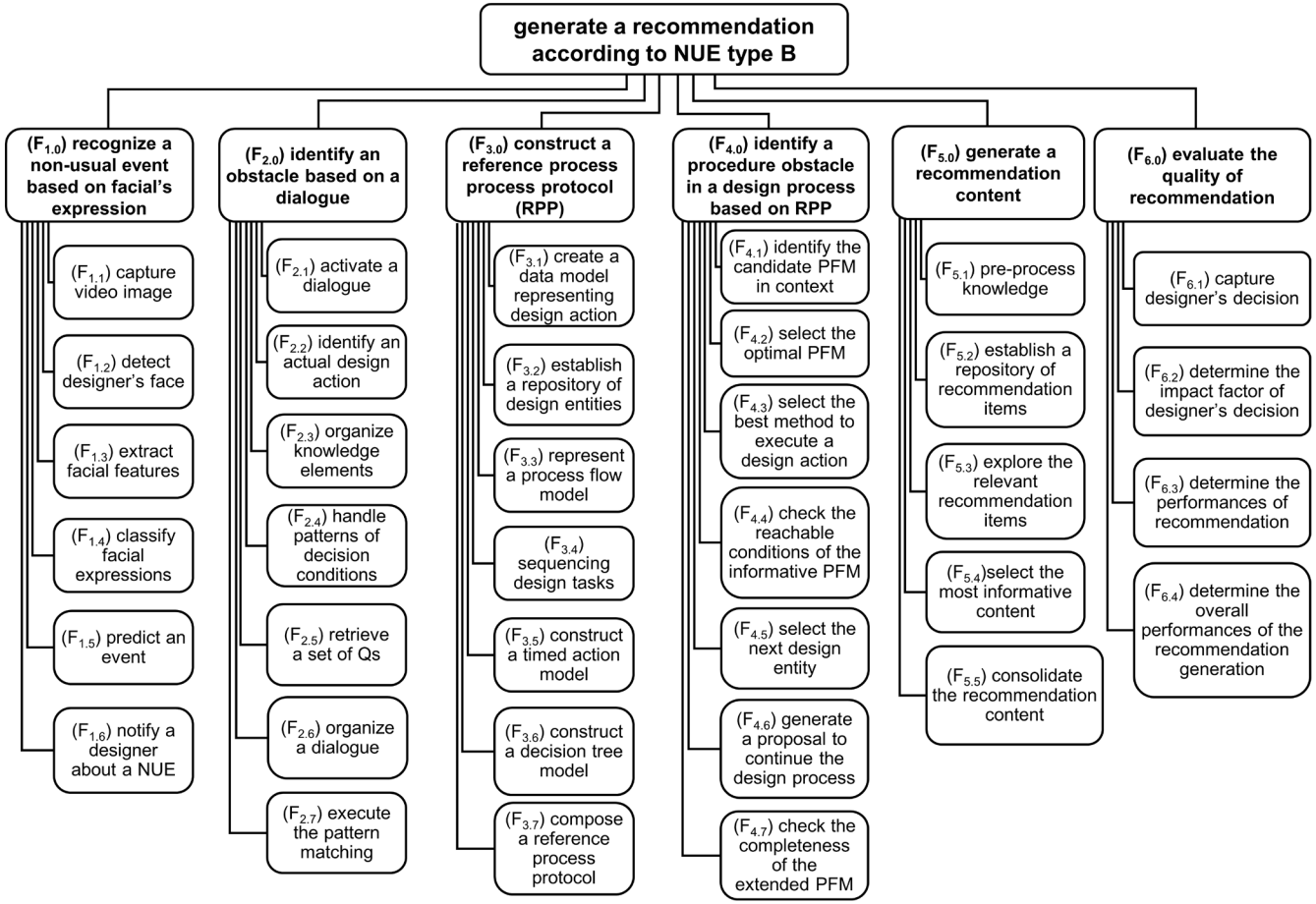


Figure 3.13: Functional decomposition of the ARF

sets of process flow models representing the design process in context. It steps back to investigate in the RPP to find the possible design actions which produce an input data for the current design action. The sub-function $F_{4.2}$ determines the probabilistic relationships of elements in the candidate PFMs and selects the optimal one. Here, the PFM representing the amended actual design flow is available.

The sub-function $F_{4.3}$ applies a decision tree model to select the proper method for each design action. The output of this function is the best combination of the current design entity and its preceding one. To ensure that all required input-output data is available for the coupled design entities, the sub-function $F_{4.4}$ checks the reachable conditions of states and transitions of the PFM. As a next step, the sub-function $F_{4.5}$ selects the next design action based on the probabilistic relations in the RPP. At this point, three design entities (i.e., the proposed current design action, its preceding and next ones) and their proper usable methods are specified. The sub-function $F_{4.6}$ generates a proposal to continue the design process. A proposal is a basis for the generation of process-based recommendation which includes the abovementioned elements. To ensure that all required data is available when it is needed, the sub-function $F_{4.7}$ checks the completeness of information flow throughout the recommended design flow.

3.5.5 Functional specification of generation of advisory content

The advisory content is a part of the content-based recommendation. It provides an informative document related to the proposed design flow and its elements. The generation of advisory content function aims at finding the most informative contents to support the exaction of the design process and presenting them in meaningful media. The fundamental concept for the generation of advisory contents is related to information retrieval. The main function is decomposed into seven sub-functions. The sub-function $F_{5.1}$ pre-processes knowledge to generate a recommendation item. It extracts the main contents from a knowledge source based on the identified terms, processes the most relevant contents, and converts them into a document as a recommendation item.

The sub-function $F_{5.2}$ establishes a repository of recommendation items. It is a specific location to store the recommendation items. Here, the collection of items is available. Since the proposal is generated, the sub-function $F_{5.3}$ explores the possible recommendation items to support the utilization of process-based recommendation. If there are multiple choices of recommendation items, the sub-function $F_{5.4}$ selects the best one which provides the most informative contents and navigates the designer to the knowledge source. Lastly, the sub-function $F_{5.5}$ consolidates all required contents and constructs them in the human-understandable format.

3.5.6 Functional specification for an evaluation of the quality of recommendation

The function for an evaluation of the quality of recommendation aims at examining

the quality of recommendation from the perspective of the designers. The process of recommendation generation works in a closed loop. The feedback from the historical activities of the designer will be used to recalculate the impacts on the recommendation item selection. The results from the evaluation of the designer's decision are also used to improve the performances of recommendation generation engine. Four sub-functions are specified. The sub-function $F_{6.1}$ captures the designer's responses on the recommendation. It is possible that the designer does not accept the proposed recommendation provided by the ARF. Two approaches are applied to capture the decision made by the designer: (i) providing the form-based user interface to directly obtain information from a designer; (ii) determining the deviation between the proposed item and the alternatives.

Based on two sources of information, the sub-function $F_{6.2}$ determines the impact factors of the designer's decision on the recommendation. The sub-function $F_{6.3}$ determines the performances of recommendation. Various metrics can be used e.g., perceived accuracy, novelty of recommendation, coverage of recommendation, and time-saving. To this end, the sub-function $F_{6.4}$ determines the overall performances of the recommendation generation. It takes both the impact factors from the designer side and the performances of recommendation engines into the consideration.

3.6 Allocation of functions to architectural constituents

3.6.1 Reasoning about the allocation of functions to architectural constituents

Software mechanisms are the highest-level architectural constituents. They implement several operations in order to provide services. A software mechanism includes multiple software modules. A module is an abstraction of interrelated software components that serve for a specific purpose. Every lower-level element works to accomplish some higher-level goals. A component includes a set of interrelated algorithms to realize its functions. This hierarchical decomposition makes the architecture of the ARF rigorously ordered and staged transition from the highest-level specification to lowest-level implementation. Designing an appropriate architectural structure of the ARF is important to meet the requirements for functional specification, interaction of designer, and allocation of computational components.

According to the requirements SR-F01, and SR-F02, the highest-level modules are constructed by one-to-one connection of the main functions of the ARF. They are either for process monitoring or for supporting decision-making related to the development of ASRM algorithms. Figure 3.14 shows the system-level architecting of the ARF, included the user interfaces. The process monitoring mechanism comprises three modules, namely: (i) module 1.0 – non-usual event detector (NUE-D); (ii) module 2.0 – dialogue-based obstacle identifier (DOI); and (iii) module 4.0 - reference protocol-based obstacle identifier (ROI). The design support mechanism has three modules: (i) module 3.0 – reference process protocol creator (RPC); (ii) module 5.0 – advisory content generator (ACG);

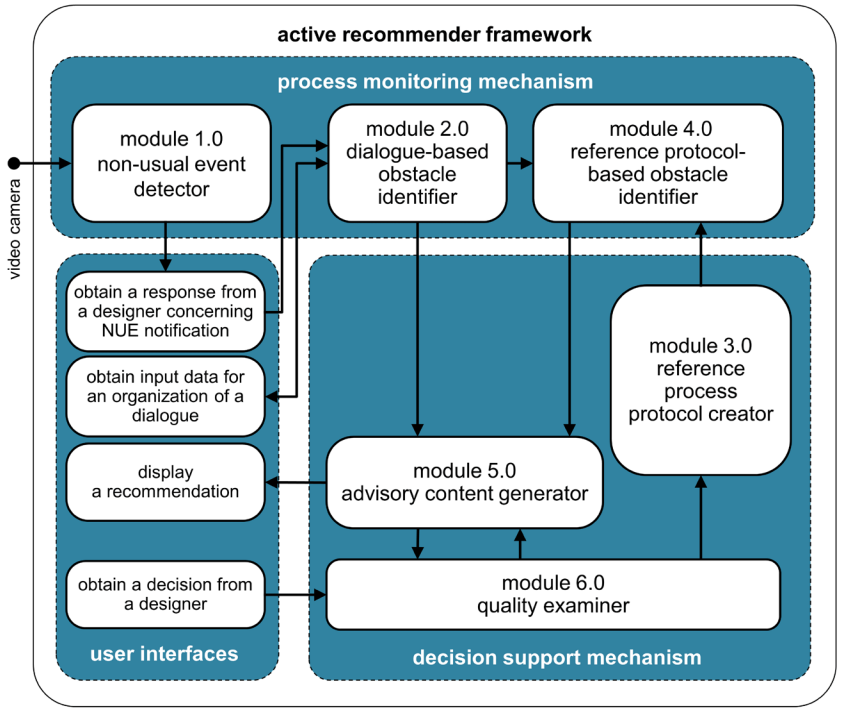


Figure 3.14: System-level architecting of the ARF for handling NUE type B

and (iii) module 6.0 – quality examiner (QE) module. The technical specifications of the architecting module were given in the following sections.

3.6.2 Architectural specifications of process monitoring mechanism

The overall architecture of the mechanism performing the process monitoring functionality is shown in Figure 3.15. It consists of three modules which were designed following the requirements SR-M01 and SR-M02. The NUE-D module implements the function F1.0 to notify a design when a non-usual event is recognized. The DOI module implements the function F2.0 that allows the ARF to gather some required information about the current design action and to investigate the current state of design action by creating a dialogue. The ROI module performs the function F4.0 to investigate the design process by using the RPP. The following sub-sections discuss the sub-module level.

3.6.2.1 Architectural specification of non-usual event detector module

The non-usual event detector (NUE-D) module observes the designer's facial expressions, monitors an event based on the patterns of facial expressions, and notifies a designer if a non-usual event is recognized. The NUE-D module is composed of four interrelated sub-

modules: (i) the face detector; (ii) the facial feature manager, (iii) the designer's behavior analyzer, and (iv) the event manager. The face detector sub-module connects to the video camera in order to obtain video images and capture the designer's face. It implements the computational operations which analyzes the area of face in the captured image and registers it the computable format. Two sub-functions are allocated to this sub-module including $F_{1.1}$ (capturing video images), and $F_{1.2}$ (detecting a designer's face).

The facial feature manager sub-module deals with the data and information about the features of faces and facial expressions including an extraction of facial landmark features,

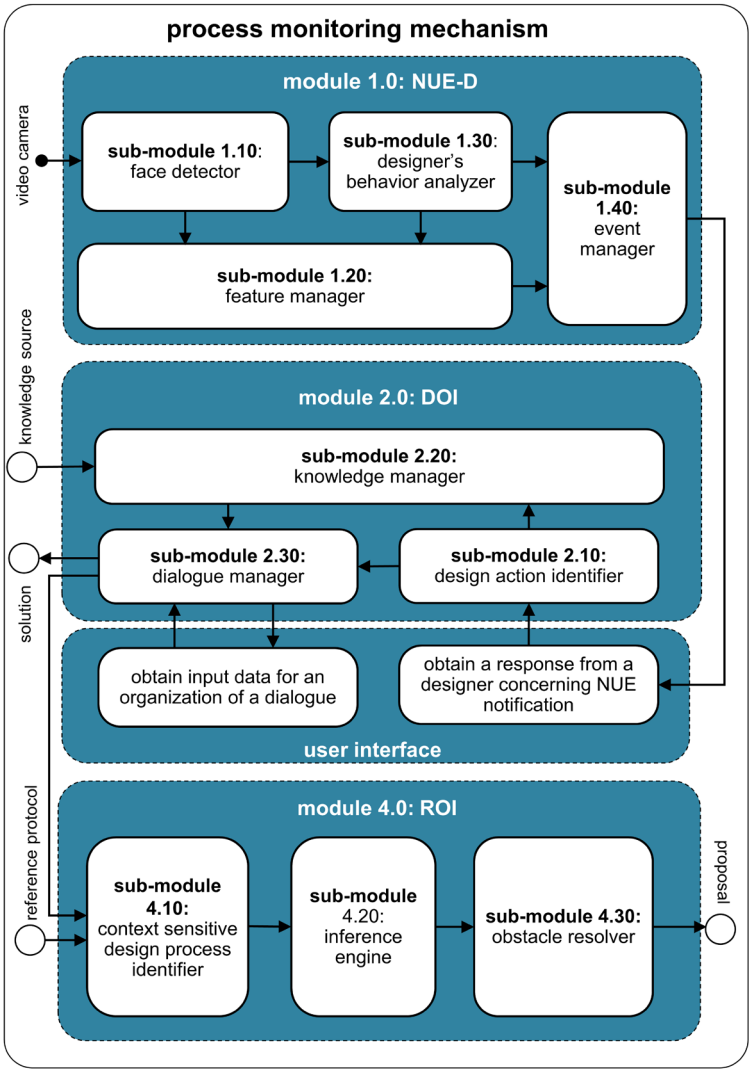


Figure 3.15: Architecture of process monitoring mechanism

a selection of the key features, and storing the features. This sub-module performs the sub-function $F_{1.3}$ and organizes the collections of facial expressions, and their features which are stored in the database. The designer's behavior analyzer sub-module is dedicated to two sub-functions $F_{1.4}$ (classifying the facial expressions), and $F_{1.5}$ (predicting an event). If an irregular pattern is found, the event manger sub-module receives the pattern and finds the best match in the database. This sub-module performs the sub-function $F_{1.6}$ to conclude an event and send a notification message to a designer.

3.6.2.2 Architectural specification of dialogue-based obstacle identifier module

The dialogue-based obstacle identifier (DOI) module comprised three interrelated sub-modules: (i) the design action identifier; (ii) the knowledge manager; (iii) the dialogue manager. The computation process of this module transforms the information from a designer to the best solution based on the exact inference. The design action identifier sub-module implements the computational components for a realization of two sub-functions, $F_{2.1}$ (to activate a dialogue) and $F_{2.2}$ (to identify an actual design action).

The knowledge manage sub-module is dedicated to three sub-functions including $F_{2.3}$ (to organize knowledge elements), $F_{2.4}$ (to handle the patterns of decision conditions), and $F_{2.5}$ (to retrieve the questions). This sub-module organizes the formal knowledge and context information related to the design action. It provides the knowledge elements as input data to the dialogue management sub-module. This sub-module uses the knowledge to contextualize the current state of design action. It organizes a dialogue and infers the best solution. Thereby, two sub-functions are allocated to this sub-module, which are $F_{2.6}$ (to organize a dialogue) and $F_{2.7}$ (to conclude the best solution). The expected output of the whole module is the proper usable method for the rectification of the actual design action.

3.6.2.3 Architectural specification of reference protocol-based obstacle identifier module

The reference protocol-based obstacle identifier (ROI) module performs the main function $F_{4.0}$. From the architecture point of view, this module contains the main functional and computational elements which are considered as the main contribution of our research. The expected output is the proposal for solving the procedural obstacle in the design process. Three interrelated sub-modules are organized to implement seven sub-functions. The context sensitive design process identifier sub-module is dedicated to two sub-functions including $F_{4.1}$ (to explore the possible set of preceding design entities), and $F_{4.2}$ (to identify the best representative of the actual design flow).

The inference engine sub-module performs the hybrid inference in order to find the most proper design entities and their usable methods. These elements are input for generating a proposal. Three sub-functions are assigned to this sub-module, which are the sub function $F_{4.3}$ (to select the usable method), $F_{4.4}$ (to check the reachable conditions of a PFM), and $F_{4.5}$ (to predict the next design entity). The obstacle resolver sub-module constructs the

elements necessary to create the most informative process flow model and to investigate the coverability of the model. This sub-module performs two sub-functions, which are: $F_{4.6}$ (to generate a proposal), and $F_{4.7}$ (to check the completeness of information flow through the model).

3.6.3 Architectural specification of decision support mechanism

The overall architecture of the decision support mechanism comprises three modules as shown in Figure 3.16. It aims at performing the decision support functionality. Each module is dedicated for the main function. The CRP module is dedicated to the function $F_{3.0}$ to construct a reference process protocol. The ACG module is allocated to the function $F_{5.0}$ to create an advisory content for the content-based recommendation. The QE module is dedicated to the function $F_{6.0}$ which is for the evaluation of the recommendation. The detailed descriptions of the sub-modules are given in the following sections.

3.6.3.1 Architectural specification of reference process protocol creator module

The reference process protocol creator (RPC) module is the core component of the decision support mechanism. It comprises four interrelated sub-modules: (i) design entity modeler; (ii) process flow model configurator; (iii) timed action model builder; and (iv) reference process protocol creator. The design entity modeler sub-module deals with the organization of knowledge elements related to a design action. It builds a design entity model for handling knowledge contents representing a design action. This sub-module serves for the realization of the sub-function $F_{3.1}$ (building a data model) and $F_{3.2}$ (establishing a repository of data models).

The process flow model configurator sub-module builds a state-transition model to represent a process flow model. It employs the structure of a Petri net model to classify the configuration of PFMs. The sub functions $F_{3.3}$ (to represent a process flow model is assigned to this sub-module. The timed action model builder sub-module plans to devise the two sub-function $F_{3.4}$ (sequencing design tasks) and $F_{3.5}$ (constructing a TAM). Lastly, the RPP representation creator sub-module is dedicated to two sub-functions including $F_{3.6}$ (to build a decision tree model) and $F_{3.7}$ (to compose of a reference process protocol). The expected output of the whole module is a representation of an RPP.

3.6.3.2 Architectural specification of advisory content generator module

The advisory content generator module is composed of three interrelated sub-modules: (i) recommendation item generator; (ii) recommendation item manager; and (iii) advisory content provider. The module consolidates the relevant contents for generating the comprehensive recommendation. The recommendation item sub-module is the computational operation of processing knowledge contents. It is dedicated to the sub-function $F_{5.1}$ (to create a recommendation item). The recommendation item manager sub-

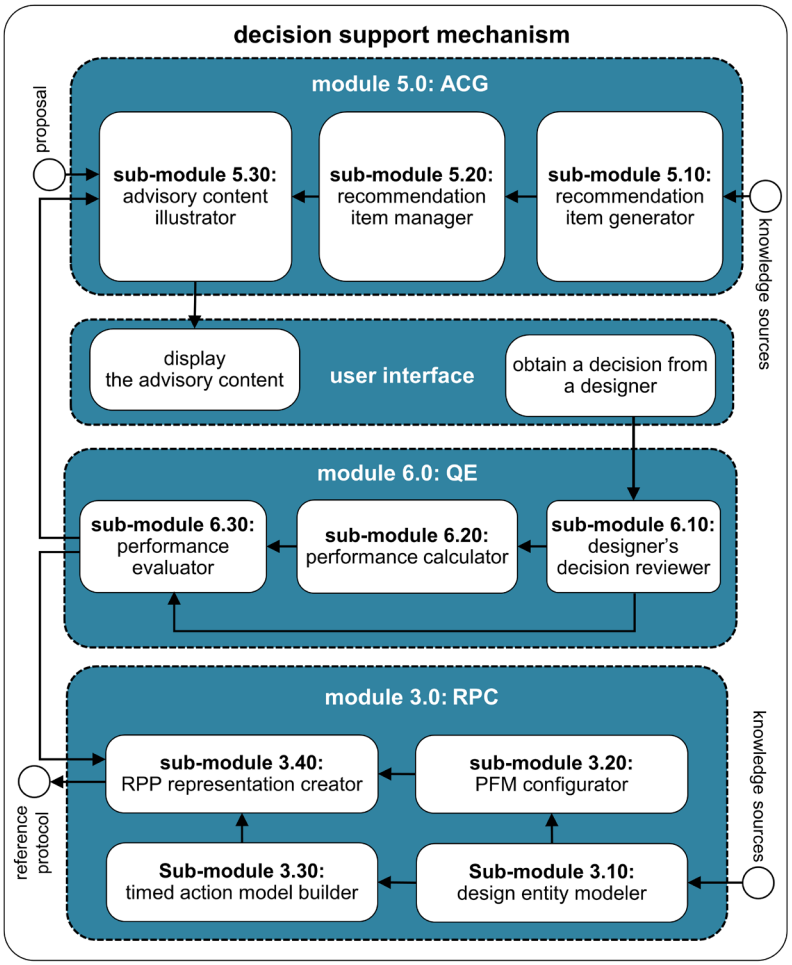


Figure 3.16: Architecture of decision support mechanism

module serves the sub-function $F_{5.2}$ (to establishes a repository of recommendation items). The advisory content provider sub-module finds the most information for the advisory contents and presents the comprehensive recommendation. Three interrelated sub-functions are planned to allocate to the sub-module, which are $F_{5.3}$ (to explore the possible recommendation items), $F_{5.4}$ (selects the best recommendation item), and $F_{5.5}$ (conclude the comprehensive recommendation).

3.6.3.3 Architectural specification of quality examiner module

The quality examiner module comprises three interrelated sub-modules: (i) designer's decision reviewer; (ii) performance calculator; and (iii) performance evaluator. The ultimate goal of this module is to obtain the feedback designer's decision on the quality of recommendation. The designer's decision reviewer sub-module intends to serve two

sub functions including $F_{6.1}$ (to capture the designer's responses), and $F_{6.2}$ (to determine the impact factors of the designer's decision). The performance calculator sub-module is reserved for the sub-function $F_{6.3}$. Lastly, the sub-function $F_{6.4}$ (evaluate the overall performance) is allocated to the performance evaluator sub-module.

3.7 Allocation of algorithms to the specified architectural constituents

The functionality of the modules has been realized either by adapting existing algorithms or by developing a set of new algorithms. Basically, an elementary function or a group of elementary functions is executed by an algorithm. In the case of complicated interrelated elementary functions, typically more than one algorithm or a purposeful composition of them is needed. It is an advantage that some existing algorithms can be reused without any change or can be adapted for differing functions. Not only structural integration of the algorithms, but also their harmonization in the time dimension is important. All specified algorithms, which are discussed in this section, concern and belong to the case of type B observation of NUEs.

3.7.1 Allocation of algorithms to the non-usual event detector module

Eight algorithms are required for the realization of the functionality for facial expression recognition (FER). They are listed in Table 3.2. These algorithms are interrelated to detect an unusual event in the design process. It could assume that a designer shows the identified facial expression during the execution of a design process. ARF Algorithm A1.01 captures video images from a video camera. ARF Algorithm A1.02 is a face detection algorithm. The Viola-Jones cascade classifier can be employed for this task. Algorithm A1.03 registers the detected face into a database. These three algorithms are interrelated. Thus, they are assigned into the sub-module1.10 (face detector). Two algorithms, A1.04, and A1.05 deal with the features. They are allocated to the sub-module (facial feature manager). The algorithm A1.04 is facial feature extraction algorithm. The output of feature extraction algorithm contains separable and classifiable vectors. Several algorithms for feature extraction can be applied for instance, Gabor filters, Susan algorithm, and *K*-mean clustering. The algorithm A1.05 is feature selection algorithm.

Table 3.2: Allocation of algorithms to the NUE-D module

FN.	required algorithms		sub-module
$F_{1.1}$	A1.01:	video image capturing	sM1.10
$F_{1.2}$	A1.02:	face detection	
	A1.03:	face registration	
$F_{1.3}$	A1.04:	facial feature extraction	sM1.20
	A1.05:	feature selection	
$F_{1.4}$	A1.06:	classification of facial expressions	sM1.30
$F_{1.5}$	A1.07:	recognition of the patterns of facial expressions	
$F_{1.6}$	A1.08:	Notify an NUE based on the pattern of facial expression	sM1.40

Two algorithms, A1.06 and A1.07, are interrelated in the process of recognition of facial expressions. They are devised to implement in the sub-module 1.30. The algorithm A1.06 is the ML-based feature classifier. The algorithm can be trained to follow one of the face recognition approaches which were mentioned in the literature. For validation the algorithm A1.06, several standard databases can be used (e.g., Cohn-Kanade dataset, T-FED, and ILF). They store a collection of images which represents eight primary different facial emotional expressions including neutral, anger, happiness, fear, contempt, surprise, sadness, and disgust. The algorithm A1.07 applies the learning algorithm to learn the pattern of facial expressions of a designer at runtime during an execution of design process.

When an irregular pattern of the facial expressions is detected, the algorithm A1.08 matches that pattern and the identified patterns which are stored in the database to recognize an indicator of a non-usual event. It is solely implemented in the sub-module 1.40 (event manager). A designer will receive a notification message according to the recognized NUE from this sub-module. However, if the best match cannot be found, then the notification is sent regularly to the designer and the service is offered. It is the automated verification of an unknown event. If the designer accepts the offer, the irregular pattern will be recorded and recognized as a non-usual event. Otherwise, it will be stored and classified as a suspicious event to be verified.

3.7.2 Allocation of algorithms to the dialogue-based obstacle identifier module

The realization of the main function $F_{2.0}$ required eight algorithms, as shown in Table 3.3. The Algorithm A2.01 activates the dialogue. It aims at inviting a design to provide the information about the design process. At the beginning, the algorithm provides information about the services and poses the very first question. A form-based user interface is used for this purpose. Algorithm A2.02 obtains context information from the designer. This information is contextualized with the available knowledge identify the current design action. These two algorithms are planned to implement in the sub-module 3.10 (design action identifier).

Three algorithms are related to the organization of knowledge elements including algorithm A2.03, A2.04, and A.2.05. They are allocated to the sub-module 3.20 (knowledge manager). The algorithm A2.03 constructs a knowledge repository of design entities. The knowledge elements contain the profile

Table 3.3: Allocation of algorithms to the DOI module

FN.	required algorithms		sub-module
$F_{2.1}$	A2.01:	activate a dialogue	sM2.10
$F_{2.2}$	A2.02:	identify a current design action	
$F_{2.3}$	A2.03:	construct a knowledge repository of design entities	sM2.20
$F_{2.4}$	A2.04:	construct a decision matrix	
$F_{2.5}$	A2.05:	retrieve a set of questions	
$F_{2.6}$	A2.06:	organize a dialogue	sM2.30
$F_{2.7}$	A2.07:	execute pattern matching	

of design entities, a collection of questions, and the lookup tables. The algorithm A2.04 converts a lookup table into the computable matrix and algorithm A2.05 retrieves the most relevant set of questions. The outputs of these three algorithms are used at the execution of the sub-module 3.30 (dialogue management). It is planned to implement two algorithms for the realization of $F_{2.6}$ and $F_{2.7}$, which are A2.06 (organize the dialogue) and A2.07 (execute pattern matching).

3.7.3 Allocation of algorithms to the reference process protocol creator module

The RPC module is dedicated to the main function $F_{3.0}$. For the realization of this functionality, the planned implementation of the module including thirteen interrelated algorithms as listed in Table 3.4. Two algorithms, A3.01 and A3.02 are allocated to the sub-module 3.10 (design entity modeler). The algorithm A3.01 builds a data model representing a design action. The algorithm A3.02 organizes the data models and their related knowledge elements. To build a process flow model, a design entity is a primary element for building a process flow model. The algorithm A3.03 is designed to perform the function $F_{3.3}$ (construct a PFM). For a development of algorithm A3.03, it could be modified based on the Petri net modelling algorithms. The algorithm A3.04 classifies the configuration of the process flow models. These two algorithms are assigned to the sub-module 3.20 (process flow model configurator).

When building a timed action model, two functions, $F_{3.4}$, and $F_{3.4}$, were taken into account. Thereby, five interrelated algorithms, A3.05, A3.06, A3.07, A3.08, and A3.09, were assigned to the sub-module 3.30 (timed action model builder). The algorithm A3.05 arranges the

Table 3.4: Allocation of algorithms to the RPC module

FN.	required algorithms		sub-module
$F_{3.1}$	A3.01:	modelling a design action	sM3.10
$F_{3.2}$	A3.02:	generation of a repository of design entities	
$F_{3.3}$	A3.03:	construction of the petri-net-like model	sM3.20
	A3.04:	net configuration identifier	
$F_{3.4}$	A3.05:	sequencing design tasks	sM3.30
	A3.06:	classification of entity2task	
$F_{3.5}$	A3.07:	construct a matrix to handle relations of d-entities	
	A3.08:	construct a matrix to handle composition relations of d-entities	
	A3.09:	timed action modelling	
$F_{3.6}$	A3.10:	train a decision tree classifier	sM3.40
	A3.11:	d-tree induction	
$F_{3.7}$	A3.12:	assemble a reference process protocol	
	A3.13:	graph construction to represent a reference process protocol	

sequence of the design tasks for a particular design process. The algorithm A3.06 classifies a design entity to the relevant design task. The algorithm A3.07 identifies the temporal relationships of design entities according to the sequence of design tasks. The algorithm A3.08 identifies the compositional relationships of design entities. The algorithm A3.09 constructs the timed action model based on the identified relationships of design entities.

The implementation of the reference process protocol creator sub-module needed four interrelated algorithms. The first two of them were algorithms A3.10 (decision tree induction algorithm), and A3.11 (training the classifier algorithm). The other two algorithms were A3.12 for assembling the elements of the reference protocol, and A3.13 for construction of the graph that represents the reference protocol.

3.7.4 Allocation of algorithms to the reference process-based procedural obstacle identified module

To realize the reference protocol-based obstacle identification functionality, ten algorithms were designed and integrated in the ROI module as presented in Table 3.5. The required algorithms were classified into three groups. Each of them is assigned to a sub-module. The first group, including the algorithms A4.01, A4.02, and A4.03, was implemented in the sub-module 4.10. They interoperate in order to select the candidate PFMs which best represent the actual design process. The algorithm A4.01 aims at exploring the possible preceding design actions. The algorithm A4.02 calculates the probability of the preceding design actions, which appeared in the actual design process. The algorithm A4.03 selects the PFM which best represents the actual design process.

In the next group, four algorithms, namely A4.04, A4.05, A4.06, and A4.07, are combined. They were assigned to the sub-module 4.20 (inference engine). Two inference approaches were incorporated to construct the most informative design flow and to select the next design action. The algorithm A4.04 selects the best usable method for the considered design action. The algorithm A4.05 evaluates the fulfilment of data

Table 3.5: Allocation of algorithms to the ROI module

FN.	required algorithms		sub-module
$F_{4.1}$	A4.01:	identify the n entities of the segment design process	sM4.10
$F_{4.2}$	A4.02:	calculation of joint distribution probability of PFM	
	A4.03:	select the most representative PFM in context	
$F_{4.3}$	A4.04:	select the best method for the PFM in context	sM4.20
$F_{4.4}$	A4.05:	reachability checker	
$F_{4.5}$	A4.06:	predict the next design action	
	A4.07:	assembling the extended PFM	
$F_{4.6}$	A4.08:	proposal generation	sM4.30
$F_{4.7}$	A4.09:	building the coverability tree for the extended PFM	
	A4.10:	coverability checker	

requirement throughout the proposed design flow. The algorithm A4.06 selects the most potential design action in order to continue the design process. The algorithm A4.07 consolidates all constituent elements to construct the extended PFM. This model is the basis of the proposal generation actions. The sub-module 4.30 encapsulates three algorithms, which generate a proposal and check the coverability condition of a PFM. The algorithm A4.08 is responsible for the former tasks, while the algorithms A4.09 and A.4.10 perform the latter task. The common coverability tree algorithm was modified to create the algorithm A4.09. With the intention to check the information flow through the PFM, simulation of the coverability tree is realized by the algorithm A4.10.

3.7.5 Allocation of algorithms to the advisory content generation module

As presented in Table 3.6, nine algorithms were required for the realization of the recommendation content generation functionality. They were allocated to three sub-modules. Four algorithms, including A5.01, A5.02, A5.03, and A5.04, were assigned to the sub-module 5.10 (recommendation item generator). Conceptually, the four algorithms are individually responsible for the four steps of converting the unstructured knowledge in a knowledge source into the informative contents. The algorithm A5.01 aims at pre-processing the knowledge obtained from the source into texts. The algorithm A5.02 finds the relevant contents by matching a query in text. The algorithm A5.03 generates a document that contains relevant information about the design actions. The algorithm A5.03 extracts the terms most frequently occurring in the document.

The task of the sub-module 5.20 (recommendation item manager) is syntactic organization of the recommendation items. Two algorithms were assigned to this sub-module, including the algorithm A5.05 (modelling the recommendation items) and the algorithm A5.06 (establishing a repository of recommendation items) Three sub-functions were dedicated to the process of finding the most informative content. Each of them was supposed to be realized by a particular algorithm. Thus, three algorithms were assigned to the sub-module 5.30 (advisory content provider). The algorithm A5.07 finds the best matching solution for a recommendation item. The algorithm A5.08 explores the most

Table 3.6: Allocation of algorithms to the ACG module

FN.	required algorithms		sub-module
$F_{5.1}$	A5.01:	convert knowledge sources to texts	sM5.10
	A5.02:	match a query in texts	
	A5.03:	generate a document	
	A5.04:	find top K words	
$F_{5.2}$	A5.05:	RecItem modelling	sM5.20
	A5.06:	RecItem repository generation	
$F_{5.3}$	A5.07:	calculate similarity of *K* terms & RecItem	sM5.30
$F_{5.4}$	A5.08:	top *N* rank of RecItem	
$F_{5.5}$	A5.09:	wrap up the comprehensive recommendation	

informative contents and ranks them by similarity scores. The algorithm A5.09 concludes about the use of a particular advisory content and presents the constructed recommendation.

3.7.6 Allocation of algorithms to the quality examiner module

Six required algorithms are specified for the realization of the evaluation of the designer's decision functionality as listed in Table 3.7. Three sub-modules are organized for the four sub-functions. For the implementation of the sub-module 6.10 (designer's decision reviewer), two algorithms are assigned, including algorithm A6.01, and A6.02. The algorithm A6.01 captures the decision of the designer. The algorithm A6.02 calculates the impact factors of designer' decision on the recommendation. The sub-module 6.20 (performance calculator) plans to implement two algorithms, which are algorithm A6.03 and A6.04. The algorithm A6.03 calculates the performance of recommendation. The algorithm and A6.04 perform cross-validation to measure the performances.

The sub-module 6.30 (performance evaluator) was prepared for embedding two algorithms, namely A6.05 and A6.06. The algorithm A6.05 determines the correlation between the 'performance' of the recommendation in terms of its impacts on the designer's decisions. Finally, all data was brought together to evaluate the overall performance by algorithm A6.06. The output of this algorithm is returned to the process of recommendation generation.

3.8 Presenting the operation of the conceptualized demonstrative part

3.8.1 Setting up a case of reasoning mechanism design for automated parking

The setting-up design scenario is that the parking problem is already defined in the problem formulation (PF) session. Specifically, the case is as follows: A car is going to parallel park between two cars on the side of the road. Several situations can happen during parking, for example, other cars may come in (and through) the scene, pedestrians may

Table 3.7: Allocation of algorithms to the ACG module

FN.	required algorithms		sub-module
$F_{6.1}$	A6.01:	capture a designer's decision	sM6.10
$F_{6.2}$	A6.02:	calculate the impact factors on the decision of designer	
$F_{6.3}$	A6.03:	calculate the performance measurements of recommendation	sM6.20
	A6.04:	cross validation of the performance measurements	
$F_{6.4}$	A6.05:	calculate the correlation of performances and designer's decision	sM6.30
	A6.06	evaluate the overall performances of recommendation	

move through the parking space and along the sideway, etc. Accordingly, to capture these, situation models are developed and stored in the knowledge repository. The situation models captured spatio-temporal information of the relevant entities involved in the parking case. This information (e.g., the identification number of entities, the entity properties, the location and orientation of the entities, and the relationship of entities) at a point in time is stored in the spatial reference feature (SFR) matrices as described in Li (2019) [15]. As shown in Figure 3.17.a-c, the parking situations in different moments of time are recorded in SFR matrices. The multiple SRF-matrices are composed and mapped onto the context information reference (CIR)-cube with regard to spatial and attributive context information, as shown in Figure 3.18.

Figure 3.17.a: Parking situation at the time *t*

Figure 3.17.b: Parking situation at the time *t+dt*

Regarding this parking scenario, there are three cars in the parking scene. Each SFR matrix records the distances between the target car and the other cars. These are measured by a set of sensors at a point in time t. The contents of all cells in the SFR matrices are used in the dynamic computation process. All relative distances can be computed for logical/semantic inferring the implications of dynamic contexts.

Figure 3.17.c: Parking situation at the time *t +(l-n)dt*

The reader should be informed here that the impacts of the contextual situations on the solution opportunities were determined at the situation analysis stage. At this point, we assume that a collection of situation models is already available in the knowledge repository. The assignment for a designer is to develop an algorithm which is able to select the best parking case in the repository to perform the parking maneuver in the real-life street parking problem.

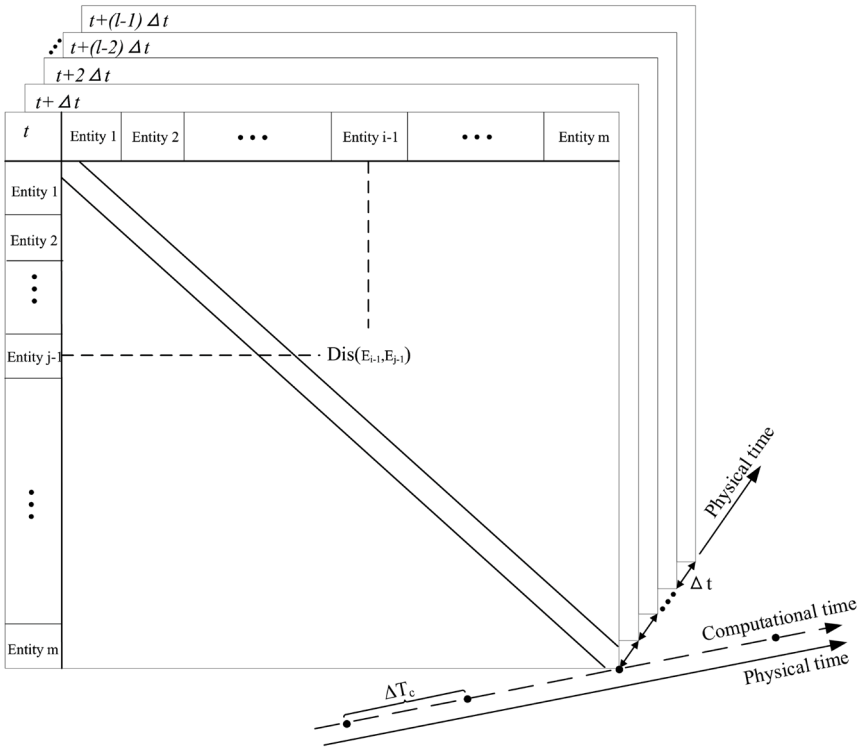


Figure 3.18: The theoretical model of the CIR cube for storing and inferring (based on [15])

3.8.2 Scoping the demonstrative example to retrieve the most appropriate parking case

In the sample case, the designer is engaged in the design process in the WPE session. The objective is to accomplish the design task formulated as below:

$\mathfrak{D}_{1.0}$ *'Construction of an ASRM-algorithm A01 for searching for and retrieving past parking cases according to actual situation from the repository of the APAS'.*

We assumed that the developed reference process protocol contains the relationships of the design actions. These relationships were considered also as the context of the development of a machine learning-based algorithm. The detailed description of design sub-tasks and related design actions are given below.

(i) *Data preparation*, which is a task to process raw data into a suitable format prior to using these data in processing and analysis. Design actions involved in this sub-task are, for instance, such as data (i) cleansing, (ii) blending, (iii) reshaping, (iv) reformatting, and (v) dimensionality reduction.

(ii) *Feature selection*, which is a task to select the most discriminatory features out of the available ones. Design actions to accomplish this sub-task are, for instance, such as (i) conducting statistical analysis, (ii) doing feature transformation, and (iii) conducting principle component analysis.

(iii) *Model training*, which is the process of applying an algorithmic model, built from a historical dataset, to explore the patterns of data and learning from the patterns to predict the dependent variable. Design actions for training the model are, for instance, such as (i) selecting a training algorithm, (ii) examining the training set, and (iii) attempting to find the finest model.

(iv) *Metric selection*, which is the task to choose the right metrics for evaluating a learning model. The selection of the metrics depends on several criteria, for instance, (i) the objective of the model, (ii) the statistical characteristics of dataset, and (iii) the expected performances of the model. Design actions are, for example, such as (i) identifying the objective, (ii) performing a statistical analysis, and (iii) developing an optimization model.

(v) *Model scoring*, which is the task to compute the metric for the evaluation of the performances of the trained model. Design actions involved in this task are, for example, such as (i) doing a statistical analysis, and (ii) applying the model with an optimization function.

(vi) *Model validation*, which is the task to test if the trained model is correct and suitable in the context of interest. The test is done by applying the model with a new dataset. Several methods could be applied to conduct a design action, for instance, (i) the fitting performances, (ii) ROV curve, (iii) hyper-parameter optimization, and (iv) statistical measures.

Concerning the development of the ML-based model, the list of design actions belonging to the considered design sub-tasks are listed in Table 3.8. Throughout the design process, the functionalities of the conducted process monitoring support the designer at executing the design task. Below, we further elaborate on how the conceptualized ARF works with this task in the WPE session.

3.8.3 Integration of the conceptualized part of the active recommender framework considering the interactions with the designer

The sequence diagram in Figure 3.19 shows the workflow of the ARF supporting the designer to perform the design process. The interaction of the ARF and the designer happens through user interfaces. In the first computational cycle, the process monitoring functionality is activated by the module 1.0 - NUE-D to monitor the designer's behavior. A video camera captures a designer's face and detects the changes of the facial expression. The recognition of an event is based on the available time-distributed information and knowledge. Whenever, a suspicious event was detected based on the recognition of recorded facial expressions during the execution of the concerned design action. It gives an

Table 3.8: Example for design actions of the development of ML-based algorithm

design sub-tasks	examples of design actions belonging to the sub-task
data preparation	cleansing data (e_{11}), dimensionality reduction (e_{12}), split data (e_{12}), blending variables (e_{14})
feature selection	select the attribute (e_{21}), generate new features (e_{22}), investigate the features (e_{23})
model training	train a classification model (e_{31}), train a regression model (e_{32}), train a clustering model (e_{33})
metric selection	perform a cross validation (e_{41}), analyses statistical testing (e_{42}), develop an optimization model to select the metrics (e_{43})
model scoring	calculate loss of the model (e_{51}), apply a simulation model (e_{52}), apply the trained model with an optimization function (e_{53})
model validation	hyper-parameter optimization (e_{61}), cross validation method (e_{62}), sequential feature selection (e_{63})

indication of the occurrence of a non-usual event (NUE). The possible causes of an NUE are from lack of information or a wrong assumption with may lead to wrong decision by the designer.

For example, suppose that the algorithm A1.07 detects an irregular pattern of facial expressions as shown in Figure 3.20. It sends the information about this pattern to the algorithm A1.08 in order to find the best match with the pre-defined patterns in the database. Once this pattern is recognized as a non-usual event, it will show a notification message to the designer and offer the service. If the designer accepts the offer, the dialogue is activated by the module 2.0 (DOI). At this point, the domain specific knowledge and context information about the design process of ASRM-algorithm A01 should be available. Conceptually, the collection of formal knowledge is stored in the knowledge repository. The dialogue poses the first question to identify the current design action in the third computation cycle. For example, the designer is working on the *feature selection* task, and struggling with '*select the attributes*'. The ARF should recognize this design action and retrieve a set of related questions.

In the fourth computational cycle, the multi-turn dialogue is organized to collect more information about the current state of design action. The number of interactions through the dialogue is equal to the total number of questions related to the identified design action. This process is a part of the recommendation generation using the exact inference approach. The detail description of this computational process will be discussed in the Section 3.8.4. Based on the exact inference, two possible alternatives are determined, either positive or negative results. If it is positive, the solution will be found. In this example, the solution is the proper method for executing the design action, 'select the attributes'. If the proper solution is not found, the investigation in the design process is required to find a procedural obstacle in the design process. The module 4.0-ROI is activated for the generation of

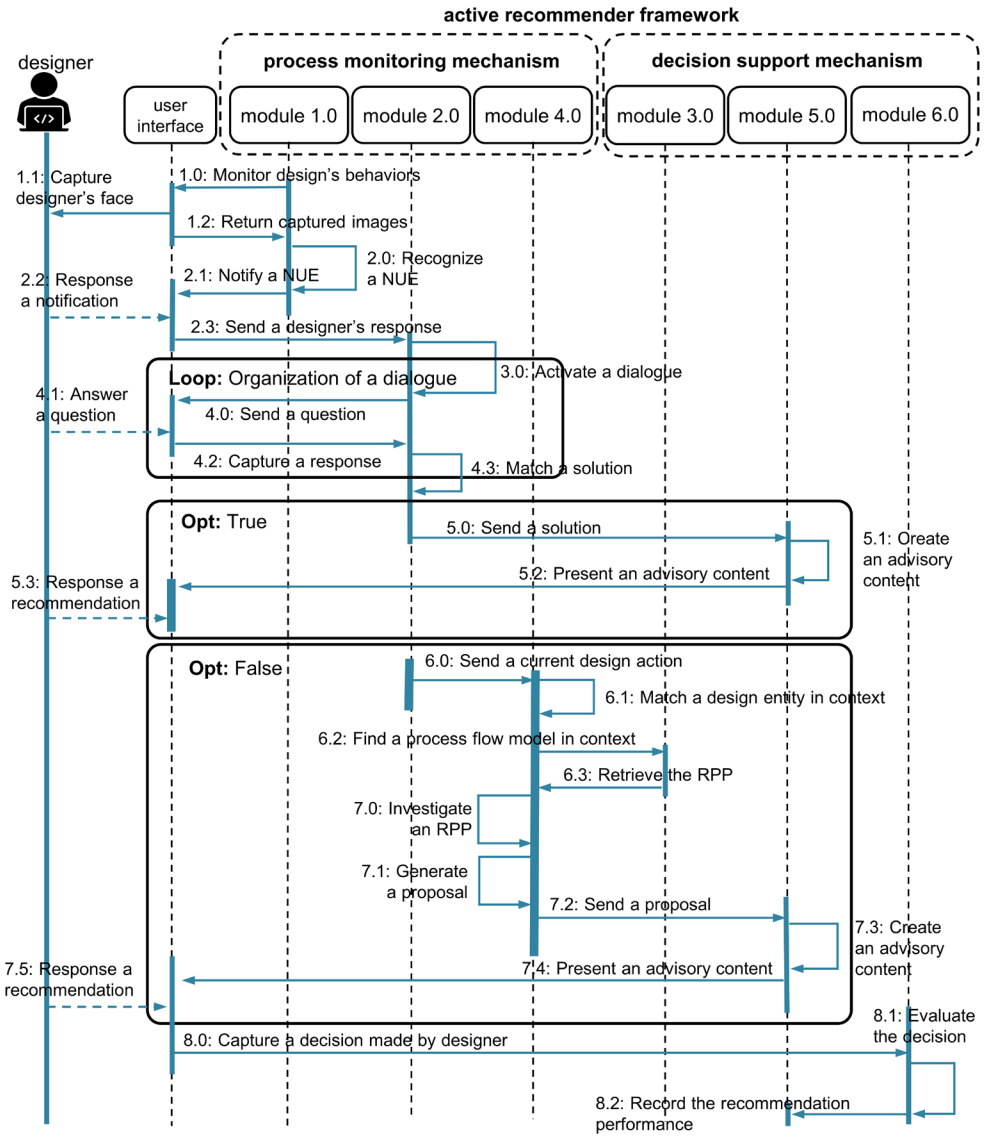


Figure 3.19: Sequence diagram representing the computational workflow of the conceptualized part of ARF

process-based recommendation. The descriptions of this computational process will be discussed in details in the Section 3.8.6.

To generate the content-based recommendation, the module 5.0 (ACG) receives two options of input data. They can be: (i) the solution produced by the module 2.0 (DOI), or (ii) the process-based recommendation generated by the module 4.0 (ROI). For the former

option, it is operated in the fifth computational cycle. The input data consists of the context information about the design action and the proposed method. The information includes the profile of the design action i.e., a set of input and output variables, textual description defining the design action, the usable method, and key terms indexing the method. The process of advisory content generation is based on the fundamental of text mining operations. It makes a query on key terms and find the most informative contents to support the execution of the design action. In the conceptualization of the module 5.0, the fundamental concept of the text similarity is applied. The key terms are extracted from knowledge in web pages. In the fifth computational cycle, the output of this process is the content-based recommendation as presented in Figure 3.21 that provides (i) the contents of design action, (ii) the proposed method, and (iii) the hyperlink that navigates the designer to the informative knowledge source.

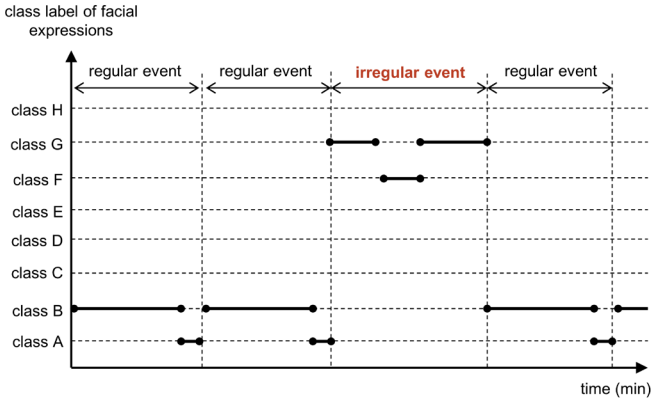


Figure 3.20: Graph representing the real-time monitoring of the patterns of facial expressions

For the second option, the operation is in the seventh computational cycle. The input data is the proposal provided by the module 4.0. It consists of three design actions and their proper methods which represent the most informative design flow. The contents of the recommendations for all three elements have the same structure as the content-based recommendation as abovementioned. In addition, the process-based recommendation shows the procedural relationships of these three design actions. As a result, the recommendations are presented to the designer. She may accept the recommendation or select another way to rectify the design process.

ID	DE2.1
Task	{'Feature selection'}
D-action	{'Select Attribute'}
Method	{'kendall's rank coefficient'}
Input	{'TrainingSet'}
Output	{'predictors' 'response'}
content	https://scikit-learn.org/stable/modules/model_evaluation.html#multilabel-ranking-metrics'

Figure 3.21: Structure of content-based recommendation

In the eighth computational cycle, the module 6.0 (QE) will capture the decision of thedesigner and use it as an input to evaluate the quality of the recommendations. The output of this process will be feedback to the module 3.0 in order to improve the performances

of recommendation generation in the next cycle of the services. This comes to the final stage of the recommendation generation. The designer is back to the design process and continues the next design action. It returns to the first computational cycle, the module 1.0 (NUE-D) continuously monitors the designer's behaviors and provides the services until the end of the design process.

3.8.4 Generation of recommendation using exact inference

The conceptualization of the process of recommendation generation using the exact inference aims at finding the solution to continue the certain design action. The computational workflow of the module 2.0 (DOI) is shown in Figure 3.22. It starts with receiving the input data from the designer to identify the current design action by the algorithm A2.02. In this demonstrative example, let us suppose that the designer is obstructed at '*selecting the attribute*'. The algorithm A2.05 will retrieve the set of questions related to this design action in the knowledge repository. The dialogue aims at investigating the characteristics of data set in order to provide the recommendation to select the proper usable method as shown in Table 3.9. According to the criteria in the lookup table, four main questions will be posed to the designer by the algorithm A2.06:

- Q1: Does your dataset have high data dimensionality? yes (if number of features > 20), no (otherwise)
- Q2: Does your dataset have the heterogeneity of features? Yes (if the features follow different distribution, no (otherwise)
- Q3: Does your dataset have the high correlation of features? yes (if two features have correlation > 80%, no (otherwise)
- Q4: Does your dataset contains an imbalanced data? yes (if the total number of different two classes is greater than 20%, no (otherwise)

The designer will respond with binary answers to these questions. The ARF captures the pattern of the responses of the designer. The set of decision criteria in the lookup table will be converted into the binary decision matrix by the algorithm A2.04. The pattern similarity is applied to find the perfect match of the patterns. The algorithm A2.07 is employed for this task. The solution is found if the sequence of binary value in two patterns is perfectly matched.

Let us suppose the designer replied the answers to the questions following this pattern {yes, yes, no, yes}. The combination of these answers describes the characteristics of the dataset as follows: the data set contains more than 20 features, each of them follows different statistical distributions, less than 20% of the features have correlations among them, and at least one feature contains imbalanced data. According to the knowledge in the lookup table, it concludes that the best method is '*Chi-square test*'. If the designer replies the answers which are diverse from the patterns of decision criteria in the lookup table, it means that no proper method suited for this dataset. This implies that the dataset should be modified in the previous design action. Thus, the investigation of the design process is needed to find

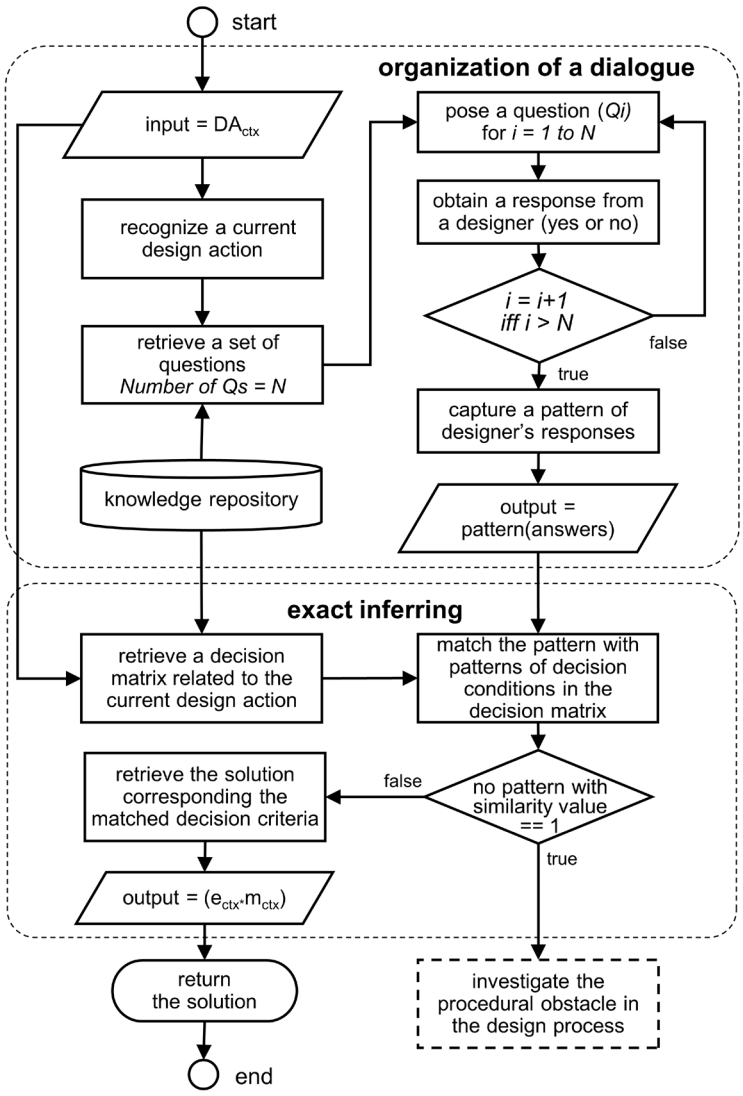


Figure 3.22: Computational workflow of recommendation generation through a dialogue

the preceding design actions and to offer the recommendation for the modification of the dataset.

3.8.5 Construction of the reference process protocol

To demonstrate the generation of a process-based recommendation, the reference process protocol (RPP) should be constructed to represent the design process in the target application context. This section describes the computational workflow for building a graph representing the RPP as shown in Figure 3.23. This workflow is a part of computational operation in

Table 3.9: Lookup table containing the decision conditions associated with the useable methodfor the design action, '*selecting the attribute*'

criteria	opt.1	opt.2	opt.3	opt.4	opt.5	opt.6
high data dimensionality	no	no	yes	yes	no	yes
heterogeneity of data	yes	yes	yes	yes	no	yes
high correlation of features	no	yes	no	no	yes	yes
imbalanced data	no	no	no	yes	yes	no
usable methods						
Pearson's correlation coefficient	**1**	0	0	0	0	0
ANOVA correlation coefficient	0	0	**1**	0	0	0
Kendall's rank coefficient	0	**1**	0	0	0	0
Information gain ratio	0	0	0	0	**1**	0
Chi-square test	0	0	0	**1**	0	0
Neighborhood component analysis	0	0	0	0	0	**1**

the RPC module. Four types of input data should be provided by the sub-module (3.10-3.30) included: (i) finite set of design entities, (ii) the timed action model, (iii) the decision tree models, and (iv) the historical data about the frequency of co-occurrences of design actions. We assume the design actions listed in Table 3.8 are members of the set of design entities for the construction of the RPP. A directed graph (G) representing an RPP is defined as $G = (N,L,F)$ where: N is a finite set of nodes $n_i \in N$ and L is a finite set of links $l_{ij} \in L$, which are ordered pair of elements of N, and $f_{ij} \in F$ is the weight connection of nodes n_i and n_j. It is the frequency of co-occurrence between entity e_i and e_j.

The procedure starts with finding the temporal relationships of design entities in the TAM. If the relationship is found, then assign the number of frequencies of co-occurrences between two entities. The next step creates a finite set of nodes without links. Assign each of all design entities to each node. If the frequencies of co-occurrences between entity e_i and e_j is greater than 1, connect node n_i and n_j with a link l_{ij} and label the link l_{ij} with f_{ij}. Otherwise find the next e_{j+1}. Repeat these steps until all entities are included in the graph. The direct graph is acyclic. Thus, it checks that no loop has occurred in the graph. If a loop is found, remove the link which opposes the sequence of design tasks. Here, the output is the directed graph representing the network of design entities. In the next part, it is to include the decision tree models into the graph. A decision tree model helps the designer to select the proper method for the considered design entities.

For all entities represented by the nodes of the graph G, if multiple choices of methods $m_i \in M$ are occurred at n_i, then find a decision tree model, which corresponds to design entities e_i. To do this, it could assume that the decision tree models are constructed and available in the knowledge repository. The output of the computational operation is shown as example in Figure 3.24. The RPP is supposed to be known by the ARF for the investigation of

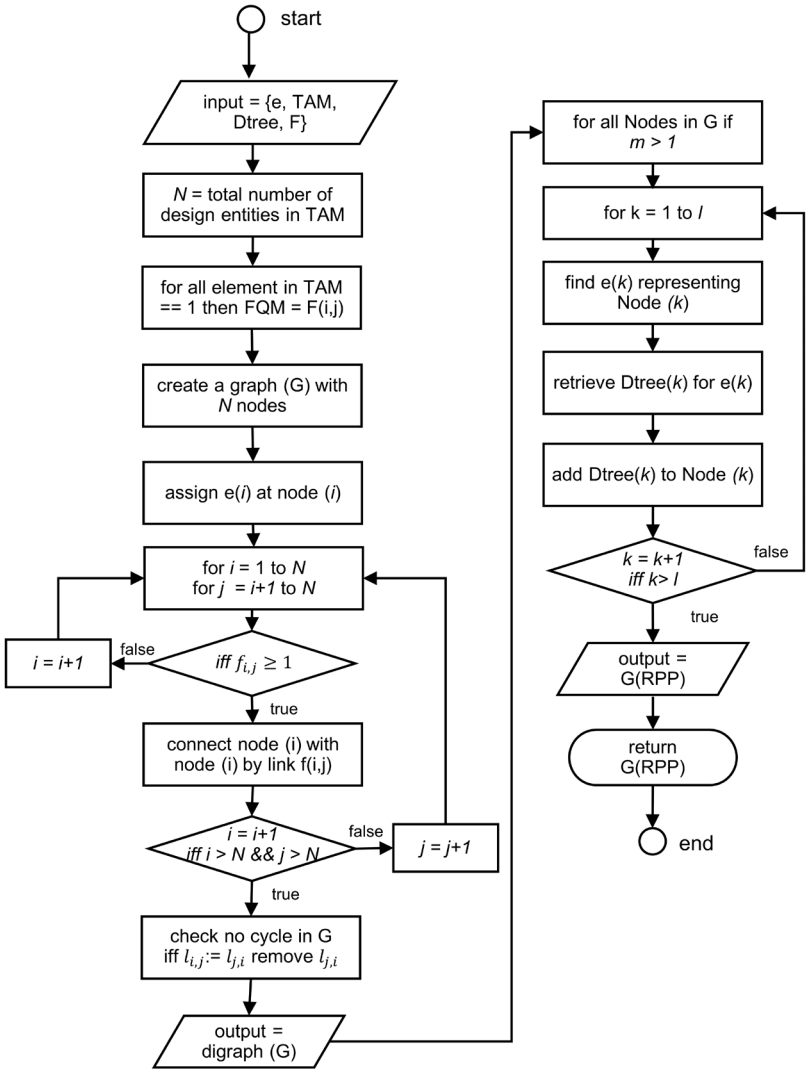


Figure 3.23: Computational workflow of a construction of graph representing RPP

procedural obstacle in the design process.

3.8.6 Generation of recommendation using the hybrid inference

Since the operation of exact inference in the DOI module gave the negative result, the function F4.0 is activated. The ROI module is operationalized to identify the procedural obstacle in the design process and propose the recommendation to resolve it. The computational workflow of this process is shown in Figure 3.25. It starts with identifying a design entity which represents the current design action in the reference protocol. Following the same

example in the previous section, the current design action is '*selecting the attribute* (e_{21})'.

It could be assumed that the cause of the obstacle in the current design action occurred at the preceding one. Then, the algorithm A4.01 uses the RPP to find the possible preceding design entities. Three options are considered as shown in Table 3.10. Each of them consists of three design entities that connects to the current design entity e_{21}. To select the most informative PFM, the probabilistic reasoning is applied by using the algorithm A4.02 and A4.03. As results, the second option of candidate PFMs gives the highest value of JDP. Three entities are included (e_{11}, e_{12}, e_{21}) in the process flow model.

Here, the preceding design entity '*dimensionality reduction* (e_{12})' is selected. The knowledge contents stored in the data model will be retrieved as necessary information for generating a proposal. In the next step, the proper method will be selected to rectify the current design action by using the algorithm A4.04. The design tree model is used for this purpose. To predict the proper method, the prediction variables are the pattern of designer's answers. If the designer already answered the questions concerning the design entity (e_{12}), that pattern will be retrieved from the knowledge repository. In case of no historical data concerning the particular design entity, the dialogue will be organized. As an example, suppose all required data for the prediction is available as shown in Table 3.11. As results, two design entities and their proper methods are composed of the informative PFM $\{(e_{ctx}, \hat{m}_{ctx}), (e_{ctx}, \hat{m}_{ctx})\}$. The algorithm A4.05 is deployed to check the completeness of the required data.

Next, the probabilistic reasoning will be used to find the next design entity in the RPP. The algorithm A4.06 finds the possible design actions presented at the nodes in the next step,

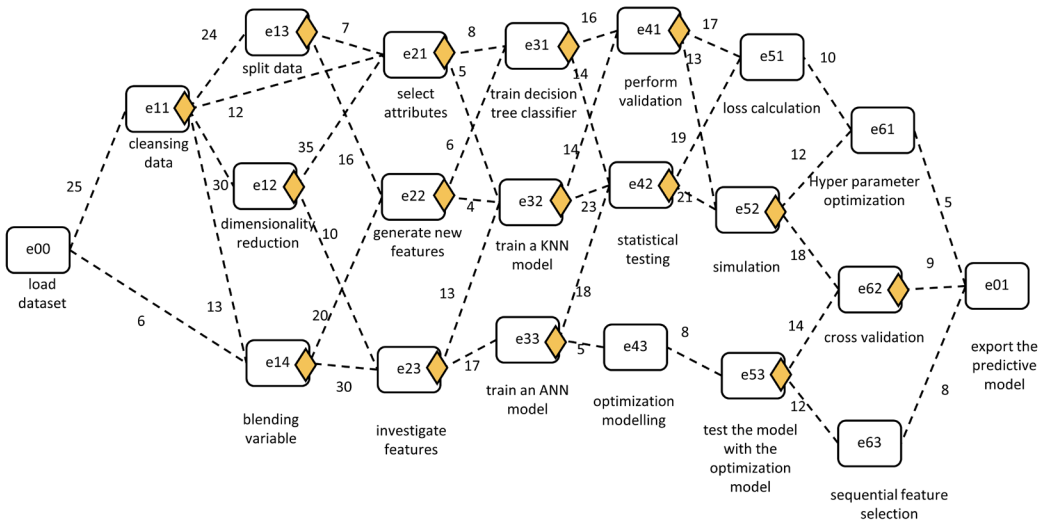


Figure 3.24: Graph representing the RPP for the demonstrative case

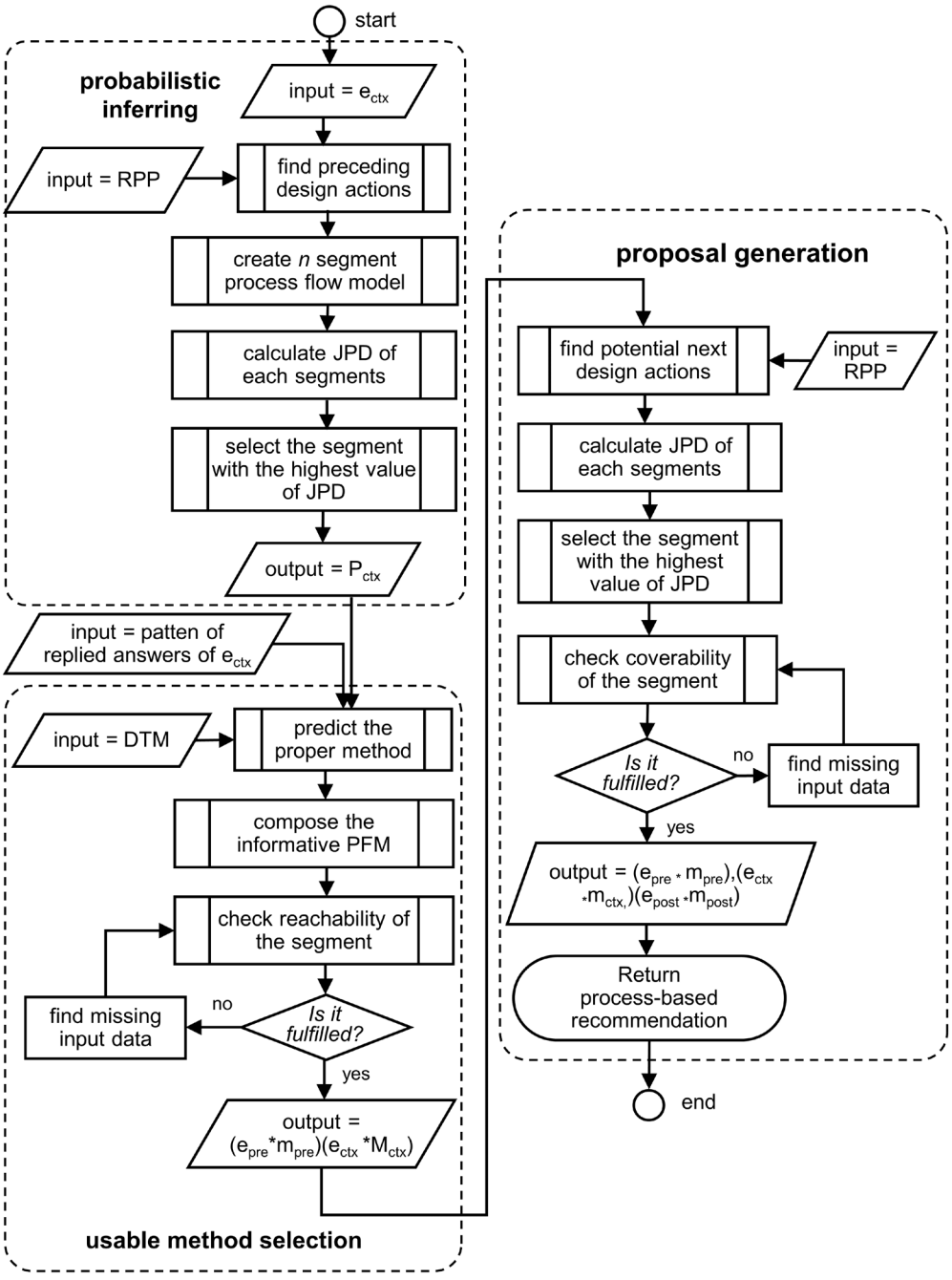


Figure 3.25: Computational workflow of recommendation generation using a hybrid inference

calculates the JPD based on two design entities and candidate next design entities, and select the best one with the highest value of JPD. In this example, *the train a classification model* (e_{31}) is selected for the next design action. The knowledge contents stored in the data model of the next design entities will be retrieved as contents for recommendation generation. Based on the hybrid inference, the process flow model including three design entities and their proper method $\{(e_{12}, \hat{m}_{ctx}), (e_{21}, \hat{m}_{ctx}), (e_{31}, m_{post})\}$ are proposed as the process-based recommendation.

Table 3.10: Calculation of JDP for candidate PFMs

candidate PFMs	joint distribution probability (JDP)
$e_0 \rightarrow e_{11} \rightarrow e_{21}$	25/(25+6) * 12/ (24+12+30+13) = 0.12
$e_{11} \rightarrow e_{12} \rightarrow e_{21}$	30/(24+12+30+13) * 35/(35+10) = **0.299**
$e_{11} \rightarrow e_{13} \rightarrow e_{21}$	24/(24+12+30+13) * 7/(7+16) = 0.094

As the last step, the algorithm A4.09 and A4.10 check the coverability of the proposal. If all required data is completed through the proposal. The content-based recommendation will be generated included the advisory contents in the ACG module. As a result of the computational operations of the demonstrative part of ARF, the process-based recommendation and content-based recommendation are combined in the comprehensive recommendation as shown in Figure 3.26.

3.9 Discussion of the findings

3.9.1 Implications of the findings with regards to the implementation of the demonstrative part

We proposed a novel concept of ARF for the development of ASRMs for a particular APAS. At the first step, the notion of the ARF is initiated with a set-up designing ASRMs scenario. Based on this initiative idea, the service packages provided by the ARF are proposed to support the setup design processes. The ARF can support the designer in the different ways of contribution in the design process. The conceptualization of the ARF was done according to the case of type B observation of non-usual events. Two mechanisms are devised to perform two essential functionalities, process-monitoring and decision-support. The activity-based monitoring is applied as the fundamental concept for the conceptualization of the process monitoring mechanism. The reference process protocol is proposed as the computational means for the conceptualization of

Table 3.11: Sample of prediction variables used for selecting the proper method for (e_{12})

decision variables of DTM	replied answers
high data dimensionality	no
heterogeneity of data	yes
high correlation of features	yes
imbalanced data	no
noise in dataset	yes
missing value	no
outlier	no
Learning algorithm (Classification or Regression)	C

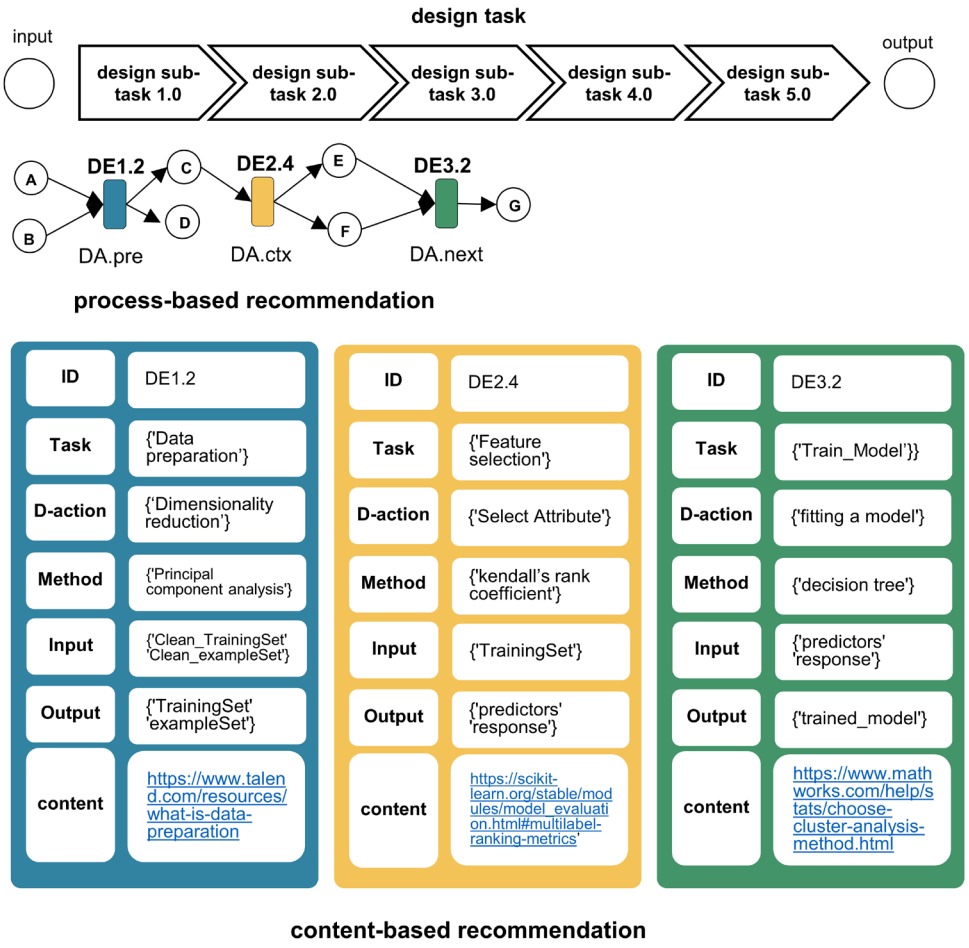


Figure 3.26: The comprehensive recommendations provided by the ARF

the decision support mechanism. Two inference approaches are employed for the generation of recommendation: (i) exact inference, and (ii) hybrid inference. The implications of the conceptualization of the ARF for the implementation of the demonstrative part are addressed as follows:

- Based on the methodological assumptions, a four-layer framework was proposed as the methodological basis of conceptualization of the ARF. The design task $\mathfrak{D}_{1.0}$ '*Construction of an ASRM-algorithm* A_{01}' will be used as the demonstrative case to test the system-level functionality. The concept of a particular WPE session will be brushed up in the implementation phase.
- Being forced by the complexity of the implementation of the ARF, we needed to prefer implementing a demonstrative part, rather than that of the whole ARF prototype.
- The selection of the demonstrative modules was based on their main contributions to

the conceptualized mechanisms. They were supposed to show the main characteristics of the implemented mechanism-level functionalities.

- Monitoring the process flow is needed to observe the design activities which possibly obstruct the design process. From the sequence diagram, the involvement of the designer in the decision-making process can be observed.
- The hybrid inference was regarded as the main contribution of the implementation of the process-based monitoring mechanism and generation of process-based recommendations. At least three sub-modules were considered for the implementation: (i) the dialogue manager, (ii) the context-sensitive design process identifier, and (iii) the inference engine.
- For the implementation of the decision support mechanism, the sub-modules related to the construction of the reference process were included in the demonstrative part. The chunks of knowledge for the construction of RPP are derived from the design process of the example use case.
- The algorithms are regarded as the lowest-level architectural elements of the ARF. They are planned to be included in multiple various computational components. The implementation of these components should be planned according to the conceptualization of the higher-level of architectural elements.

3.9.2 Identification of requirements for the implementation of the demonstrative part

Based on the implications for the implementation of the demonstrative part, the requirements for the demonstrative implementation are explored. They are categorized into two levels: (i) the module level, which is considered as the functional requirement, and (ii) the algorithm level which is considered as the computational requirement. The requirements were specified based on the considered sub-modules which were discussed in the previous section.

3.9.2.1 Functional requirements

FR01: the DOI module should recognize the actual design situation (state of design actions) based on the lowest possible number of answers

FR02: the DOI module should use the lowest possible number of decision criteria to find a solution for the problem at hand

FR03: the DOI module should reply a question posed by the designer within 0.1 second

FR04: the RPC module should include the relationships of design actions with a lower than 5% of incorrectness of their relations

FR05: the RPC module should construct an extendable network of design actions with a higher than 90% of coverage of the considered design process

FR06: The RPC module should model individual design actions or chains of design

actions which are extractable from the computational representation of the network of design actions.

FR07: the ROI module should be able to predict and provide proper corrective action (to resolve the obstacle in the design process) with a higher than 90% of reliability rate

FR08: the ROI module should be able to identify the conducted design action with a probability higher than 90% of accuracy rate

FR09: the ROI module is supposed to generate a recommendation at runtime with a rate of higher than 45% justified objective decisions in line with the actual context

FR10: the ROI module should provide a process-based recommendation for the designers concerning the avoidance of procedural hindrances with a higher than 90% of reliability rate

FR11: the ACG module should offer a content-related (decision making) recommendation in varying procedural contexts with a higher than 45% justified objective decisions in line with the actual context

3.9.2.2 Computational requirements

CR01: the algorithm A2.07 should provide a near zero-time response to a returned question

CR02: the algorithm A2.08 should select the best solution with a lower than 5% of incorrectness ratio

CR03: the algorithm A3.09 should construct a Timed action model with a lower than 5% of incorrect relationships between the concerned design actions

CR04: the algorithm A3.10 should predict the most usable method for a design action with a higher than 90% of accuracy rate

CR05: the algorithm A3.11 should construct a process flow model with a lower than 5% of incorrectness rate in terms of the number of mismatched elements

CR06: the algorithm A3.12 should include the design entities and decision tree models in a graph representing RPP with a lower than 5% of incorrectness rate in terms of the number of mismatched elements

CR07: the algorithm A4.01 should include the design entities in the process flow model with a higher than 90% of reliable rate in term of the relationships between the design entities

CR08: the algorithm A4.03 should retrieve the candidate PFMs in context with a higher than 75% of accuracy rate

CR09: the algorithm A4.06 should predict the next design action that includes in the proposed recommendation selected by a designer with a higher than 45% of justified subjective decisions.

CR10: the algorithm A4.07 should generate a process flow model within less than 0.1 seconds

CR11: the algorithm A4.08 should include the design entities in the proposal with a higher than 90% of reliable rate in term of the relationships between the design entities

CR12: the algorithm A5.07 should select the recommendation item for a certain design action with a less than 5% of incorrectness rate

The technical implementation of these requirements (i.e., architectural design, algorithmic programming, specification of resources, and design scenario for the testing case) will be discussed in Section 4.2.1 in Chapter 4.

References

[1] Patokorpi, E. (2006). *Role of abductive reasoning in digital interaction*. [Doctoral thesis, Åbo Akademic University].

[2] Szádeczky-Kardoss E., & Kiss, B. (2008). Path planning and tracking control for an automatic parking assist system. In *European Robotics Symposium* (pp. 175–184). Springer Berlin Heidelberg.

[3] Moon, J., Bae, I., & Kim, S. (2019). Automatic parking controller with a twin artificial neural network architecture. *Mathematics Problems in Engineering*.

[4] Choi, S., Boussard, C., & D'Andréa-Novel, B. (2011). Easy path planning and robust control for automatic parallel parking. *IFAC Proceedings, 44*(1), 656-661

[5] Seyoung, P., Mye, S., Haeran, J., & Lee, H. (2016). Situation reasoning framework for the Internet of Things environments using deep learning results. In *IEEE International Conference on Knowledge Engineering and Applications* (pp.133-138).

[6] Liu, C., Van Dongen, B., Assy, N. & Van Der Aalst, W. M. P. (2017). Component behavior discovery from software execution data. In *IEEE Symposium Series of Computational Intelligence* (pp.1-8). IEEE.

[7] Saadah S., & Wulandari, G. S. (2015). Anomaly detection from log files using data mining techniques. In *Information Science and Applications* (pp. 449-457). Springer, Berlin, Heidelberg.

[8] Sun G., & Yao, S. (2011). A new framework of studying the cognitive model of creative design. In *International Conference on Engineering Design*.

[9] Lu, Y., Wang, S. & Zhao, W. (2019). Facial expression recognition based on discrete separable shearlet transform and feature selection. *Algorithms, 12*(1), 11.

[10] Wang, Y., Ai, H., Wu, B., & Huang, C. (2004). Real time facial expression recognition with adaboost. In *Proceedings of the 17th International Conference on Pattern Recognition* (Vol. 3, pp. 926-929). IEEE.

[11] Dino, H. I., & Abdulrazzaq, M. B. (2020). Comparison of four classification algorithms for facial expression recognition. *Polytechnic Journal, 10*(1), 74–80.

[12] Kantharia, K. J., & Prajapati, G. I. (2015). Facial behavior recognition using soft computing techniques: A survey. In *Proceeding of the 5th International Conference on Advanced Computing & Communication Technologies* (pp. 30-34). IEEE.

[13] Balaban, S. (2015). Deep learning and face recognition: the state of the art. In *Biometric and Surveillance Technology for Human and Activity Identification XII*

(Vol. 9457, p. 94570B). International Society for Optics and Photonics.
[14] Dupré, D., Krumhuber, E. G., Küster, D., & McKeown, G. J. (2020). A performance comparison of eight commercially available automatic classifiers for facial affect recognition. *PLoS One, 15*(4), 1–17.
[15] Li, Y. (2019). *Utilizing dynamic context semantics in smart behavior of informing cyber-physical systems.* [Doctoral dissertation, Delft University of Technology]

Chapter 4

Research cycle 3:
Implementation of a demonstative part of the active recommender framework

4.1 Objectives and methodological framing of the third research cycle

4.1.1 Research and development objectives

The goal of the third research cycle was to realize the computational mechanisms of the ARF, which (i) support various stages of the design process of ASRMs in the case of particular S-CPSs; (ii) include multiple computational resources for handling the various stages of the design process; (iii) provide recommendations based on process monitoring and involvement in design problem solving/decision making; and (iv) gradually aggregate data, information and knowledge concerning the design processes and learn support opportunities. This chapter presents the implementation principles and procedures of the computational mechanisms of the demonstrative part of the ARF, which is dedicated to a non-usual event type B.

Based on the forerunning conceptualization of the ARF, we realized that the complexity of the computational mechanisms is one of the major challenges in the implementation phase. The algorithm-level functional implementation of the considered computational mechanisms required fifty-two interoperating algorithms. To cope with structural complexity, the architecture of the ARF was decomposed to self-contained sub-modules, with due attention to optimizing the functional relationships amongst the computational components. This was necessary since the number of components and their relationships contributed to the increase of complexity. Putting everything together, we have managed to draw up a realistic scope for a demonstrative implementation, which is intended to clarify

the main characteristics of the ARF, but also avoid any unmanageable complexity of the computational implementation.

To achieve the goals and deal with the challenges of the implementation, there were seven specific objectives:: (i) to operationalize the requirements for the implementation; (ii) to select the demonstrative parts for the implementation; (iii) to clarify the principles for the implementation; (iv) to specify usable resources in the programming environments; (iv) to specify the contents of the demonstrative modules; (v) to implement the demonstrative modules; (vi) to implement the contents of the demonstrative modules in the given contexts; and (vii) to validate the demonstrative implemented modules in the target application context. These objectives were considered while setting up an appropriate implementation scenario.

4.1.2 Methodological framing

The research cycle was framed methodologically according to the procedural structure of design inclusive research (DIR) [1]. The main contribution of the research cycle was the computational implementation of the chosen demonstrative part of the ARF. The inquiry activities were driven by the need for (i) knowledge aggregation, (ii) completion of the constructive (design) activities, and (iii) validation of the implementation. Accordingly, there were three procedural phases defined that started with the exploratory research actions, continued with the design actions, and concluded the confirmatory research actions. The ARF conceptualization knowledge was completed with implementation related knowledge.

The exploratory phase considered three kinds of issues concerning the implementation of the demonstrative part of the ARF, namely (i) strategic issues, (ii) tactical issues, and (iii) operational issues. The strategic issues concerned four research activities: (i) transformation of requirements; (ii) identification of the possible approaches to implementation of the demonstrative algorithms; (iii) determining the critical algorithms for the demonstrative implementation; and (iv) selection of computational resources for the working environment. The research activities concerning the tactical issues focused on the specification of the implementation principles for the targeted demonstrative modules. The operational issues were related to the programming resources available in programming environments, including the fundamental programming language, its built-in functions, applied toolboxes, and the library of application development functions.

In the constructive phase, software engineering activities were conducted that concentrated on the elaboration of the contents of the demonstrative modules. The whole of the ARF was conceptualized as a compound of six modules. However, the demonstrative part was reduced to the four modules which were closely associated with recommendation generation. These were: (i) the dialogue-based obstacle identification module; (ii) the reference process protocol construction module; (iii) the reference protocol-based procedural obstacle identifier module; and (iv) the advisory content generating module. As far as architecting of the different modules was concerned, they were specified on three levels: (i) on the sub-modules level, (ii) on the components level, and (iii) on the algorithms level. The

elaborated architectural models arranged both the architectural elements and their structural interrelationships on the respective levels. The constructive phase was completed by the implementation of all algorithms necessary for the demonstrative part of the ARF.

The confirmatory phase commenced with placing the architectural constituents into the context of the target application. The major research activities were: (i) identification of the content and procedural elements of the design process in the given context, (ii) specification of the design actions in the target application context, (iii) actual operationalization of the computational algorithms, and (iv) assessment of the operation against the derived implementation requirements. The confirmatory phase ended with a detailed elaboration on the findings related to the computational implementation and validation. The results of the validation were expected to confirm that the proposed concept and the resources offered by the ARF could be utilized in the defined application context.

4.2 Strategic issues of the demonstrative implementation

The conceptualization of the ARF gave priority to two interoperating mechanisms, which were decomposed into six modules. We regarded the reference process protocol (RPP) as a most novel and most essential constituent in terms of generating recommendations. Two of the modules were directly related to the RPP. One of them was Module 3.0 (for construction of the RPP) and another one was Module 4.0 (for utilization of the RPP in identification of a possible procedural obstacle in the design process). The major challenges of the demonstrative implementation were (i) to capture knowledge about the design process in the RPP), (ii) to utilize it when inferring the actual state of the design activity flow, and (iii) to generate process-based recommendations. However, these two modules do not interact with the designer. Therefore, to complete the entire process of recommendation generation we selected two other modules. One of them was Module 2.0 (for direct communication with the designer through a dialogue), and the other is Module 5.0 (for consolidating the recommendations and providing an advisory content to the designer).

Thus, four demonstrative modules have been selected for implementation in this part of the promotion study. These are shown in Figure 4.1. From a computational point of view, they demonstrate the entire process of context-sensitive recommendation generation. Before entering the implementation phase, we have analyzed what computational components should be implemented for the ARF. As an outcome of this functional and architectural analysis and conceptualization, forty-one algorithms were required and included in the demonstrative modules of the ARF. Dedicated implementation specifications were elaborated and operationalized for each specific module. In addition, a forerunning usability evaluation of the required algorithms was completed based on critical system thinking and the criticality of the algorithms from the point of view of the demonstrative implementation. An early reflection was also made from the perspective of feasibility by considering candidate programming languages and the available developer resources (e.g., methods, functions and libraries) of the existing programming environments.

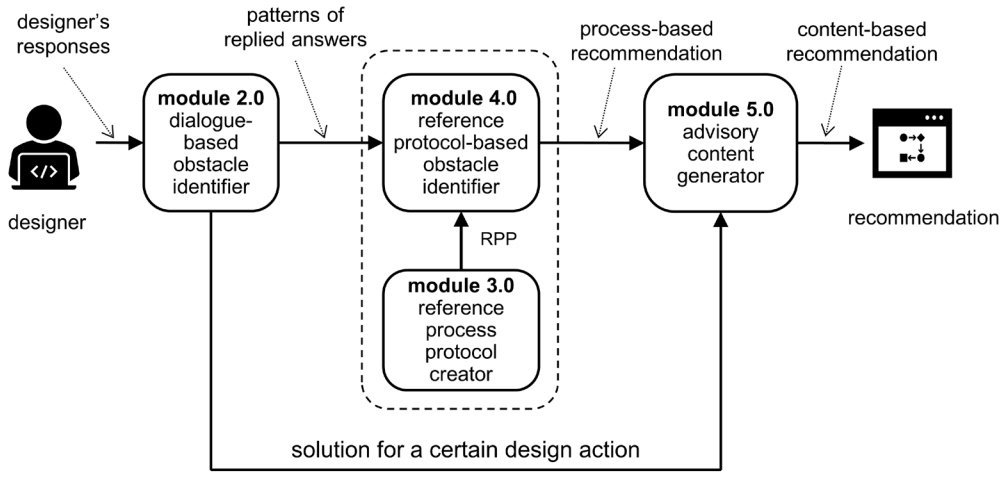


Figure 4.1: The general workflow of the recommendation generation according to the demonstrative implementation

4.2.1 Transformation of the implementation requirements

The goal of the transformation of the implementation requirements was to systematically convert them into technical specification for algorithmic programming, knowledge acquisition, and testing scenario in an application context. The requirements provided guidelines not only for functional and other expectations, but also formulated the criteria to fulfill in order to meet the requirements. All requirements had to be considered in order to functionally harmonize the operations of the implemented modules of the ARF and to produce the expected output. The results of the analysis are shown in Table 4.1. It includes a technical specification of the modules of the demonstrative implementation

4.2.2 Possible approaches to implementation of the demonstrative algorithms

In order to reduce the unnecessary time, costs, and workload that are concomitant with the development of algorithms purely from scratch, we applied a different strategy. We also considered the availability, applicability, and adaptability of proprietary and commercialized algorithms needed for computational implementation. Accordingly, based on the extent of their reusability, we classified the relevant algorithms into three groups, namely: (i) existing algorithms that can be reused without any modification; (ii) existing algorithms that can be adapted for the purpose of application; and (iii) brand new algorithms that should be deigned and coded for the specific purpose. The classification of the required algorithms is shown in Table 4.2. The classification was based on the below considerations.

As briefed above, the first group included existing algorithms, which were tested and used as solutions for particular tasks: for instance, search algorithms, matrix operations, string similarity comparators, and descriptive statistical analyzers. Usually, they are available

Table 4.1: Technical specification of the modules of the demonstrative implementation

modules	algorithmic programming	knowledge resources	requirements
DOI	pattern similarity algorithm	knowledge contents about the computational methods for performing design actions	FR01, FR02, FR03, CR01, CR02
CRP	graph construction, visualization and analysis decision tree modelling matrix-oriented operations	domain knowledge about design actions and process for ML-algorithm development knowledge about the characteristics of dataset for training ML-algorithm	FR04, FR05, FR06, CR03, CR04, CR05, CR06
ROI	probabilistic computation and analysis process flow		FR07, FR08, FR09, FR10, FR11, CR07, CR08, CR09, CR10, CR11
ACG	data mining text-similarity	knowledge contents about the computational methods for performing design actions	FR12, CR12
application context - APAS	ML-based development statistical analysis	simulation of parking problem	FR10

in the form of built-in functions or library items of programming environments. If the computational functions provided by them are needed, they can be reused in software development without modification. The second group included existing algorithms, which partially fulfil the needs for certain computational functions, complying interface specifications, or handling certain data constructs. However, they offer the opportunity of modification for the purpose with reasonable efforts.

Typically, these are also available in common programming libraries, but they often represent proprietary codes or programs that are computational functions shared within online communities of software developers. The extent of adaptation needed for making this type of algorithms applicable in software engineering, depends on technical, economic, knowledge and practical factors and can be largely influenced by the target software and environment.

In our case, these were all considered when selecting and making decisions on the use of adaptable codes in the demonstrative computational components. The last group includes novel algorithms that have not been designed and coded previously and cannot be derived by modifying existing ones. In the process of development of the demonstrative part of the ARF, several brand-new algorithms were needed for the realization of the target

Table 4.2: Classification of the required algorithms according to their reusability for the demonstrative implementation

FN.	type of the required algorithms		
	adoptable	**adaptable**	**to be generated**
F2.0	A2.01, A2.02	A2.03, A2.04, A2.05, A2.07	A2.06
F3.0	A3.02, A3.08	A3.01, A3.03, A3.10, A3.11	A3.04, A3.05, A3.06, A3.07, A3.08, A3.09, A3.12, A3.13
F4.0		A4.02, A4.05, A4.09, A4.10	A4.01, A4.03, A4.04, A4.05, A4.07, A4.08
F5.0	A5.01, A5.02, A5.03, A5.04, A5.06	A5.05, A5.07	A5.08, A5.09

functionalities. Despite these differences, all the three groups of algorithms had to go through various rigorous pre-application and post-application testing.

4.2.3 Determining the critical algorithms for the demonstrative implementation

This Section discusses the criticality of the algorithms from the perspective of implementation of demonstrative modules. Our assumption was that critical algorithms are (i) novel in multiple aspects (new designs and not yet exhaustively tested in practical applications) and play a crucial role (influential in the computational implementation of the ARF). In addition, we assumed that focusing on the critical algorithms is necessary and sufficient for a crude but effective functionality validation of the demonstrative implementation (but not for competing with long lasting tests in practical applications). Typically, a critical algorithm is an algorithm that influences the operation of a large number of other algorithms of a software system. Changes in a critical algorithm may have a large impact on other algorithms regarding data provisioning, procedural timing, and operational reliability.

This is especially important because many algorithms are designed to be reactive and mutable to inputs. They are often unpredictable in the sense that their outcomes are not easy to anticipate and they may produce unexpected effects. They can also be sensitive to data values and to change of computational constrains, but they may also depend on multiple other factors. What these facts meant for us was that one aspect is probably not sufficient and adequate to identify a critical algorithm. Therefore, we determined the criticality of algorithms based on a combination of three criteria: (i) complexity of functionality – which ranks algorithms according to the complexity of the computational function they realize, (ii) data sensitivity – which refers to the impacts of change in the amount of input data on the computational performance of an algorithm, and (iii) intensity of human interaction – which refers to the degree or intensity to which an algorithm demands input from the user. Figure 4.2 shows a Venn-diagram, which classifies the required algorithms according to the implementation criteria.

The criticality analysis of the algorithms was based on those detail descriptions, which were included in Section 3.7 in Chapter 3. Interestingly, as shown in Figure 4.2, algorithm A4.04 – '*Select the best method for the process flow model (PFM) in context*' was found to be the most critical algorithm. The reason for this was that one of the related requirements considered formulated the need to limit the number of interactions with the designer to the lowest possible minimum. This in turn imposed a limitation on the total number of questions that could be posed to the designer in a dialogue.

Regardless, most of required algorithms were determined according to the first two expectations and do not require direct input from the designer. The investigation of the interrelationships among the computational components found that eight of the algorithms generated from scratch fell into the overlapping areas in the Venn diagram. These algorithms will be further discussed in Section 4.5, which focuses on the specification of the computational implementation. Here we talk about the following algorithms: (i) algorithm A2.07 – organizes a dialogue, (ii) algorithm A3.12 – assembles the RPP, (iii) A3.13 – visualizes the graph of the RPP, (iv) algorithm A4.01 – identifies the *n* entities of the PFM in context, (v) algorithm A4.03 – selects the candidate PFMs in context, (vi) algorithm A4.07 – predicts the next design action, (vii) algorithm A4.08 – assembles the extended PFM, and (viii) algorithm A4.09 – generates a proposal.

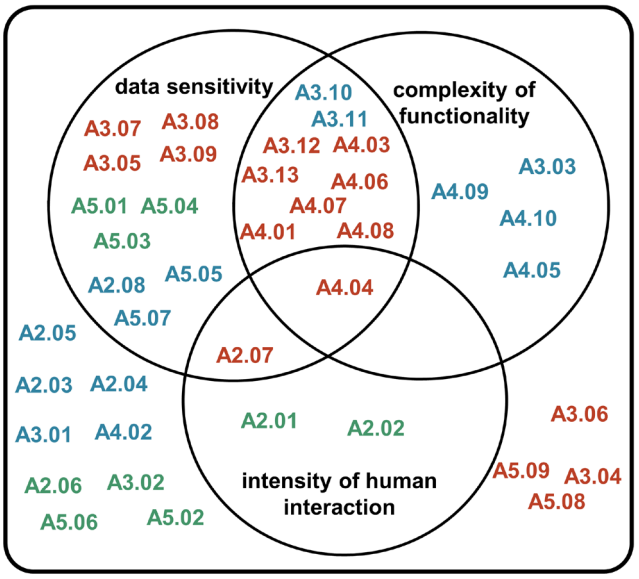


Figure 4.2: The classification of the required algorithms according to criteria

4.2.4 Forerunning considerations

The lowest level of the implementation of the demonstrative part of the ARF raised a number of concerns. In addition to the above productivity concerns, the selection of the working (programming) environment was also high on the list of concerns. It has an influence on the coding (first-time and adaptive) as well as on the logical and data integration of the outcome required algorithms. The primary intention was to use as many standard (commercialized), widely used, and tested computational resources as possible. The below considerations and decisions were made with this in mind.

In general, a programming language is the formal language with a set of instructions which provide the desired output. Every programming language is based on certain syntactic and semantic rules. In the field of information engineering, several programming languages have been widely used, for example, C, Java, Python, and C++. Currently, the computer programming languages have evolved into the fifth generation (5GLs). However, this is the result of a gradual evolution, rather than an overnight action, which was instructive for us

The first- and second-generation languages (1GLs and 2GLs) were closely related to the computational architectures and computational mechanisms [2]. The first-generation software was written in machine codes. The second-generation languages were low-level assembly languages that were specific to a particular computer and processor [3]. The concept of higher-level programming appeared in the case of the third-generation languages (3GLs). These languages are generally translated by compliers into machine language/code of the target computers for execution. The high-level programming languages were developed for general application purposes. Typical representatives are C, C++ and Java. They have been applied in business and scientific programming, as well as in other commercial applications. The fourth-generation languages (4GLs) include statements similar to statements in human languages. These were used mainly in database programming and scripting. Commonly known examples of these languages are Perl, Python, Ruby, Go, R and MATLAB. All 4GLs were designed to reduce programming effort compared with the earlier generations [4]. The fifth-generation programming languages are based on constraint-driven problem solving. Dedicated mechanisms are applied in the development of the program, rather than algorithms written by a programmer. In other words, 5GLs are designed to make the computer able to solve a certain problem for a user [5].

According to the specific programming requirements, the DOI module is dedicated to identify an obstacle related to a particular design action and to offer a solution based on the exact inference. The module monitors what a designer was doing at that time through a dialogue. The principles of human-machine interaction using a dialogue, content-oriented decision table, and a pattern similarity measure were taken into consideration. The PRC module focused on building a reference process protocol and provision of knowledge concerning the constituent elements of RPP including (i) matrix representation for timed action model, (ii) construction of ML-based decision tree classifier, and (iii) construction of a graph and network. The implementation of ROI module required probabilistic inferring by means of

a Bayesian network, and semantic inferring by using the interoperation of a decision tree model and probabilistic reasoning. Finally, the implementation of ACG modules required an information retrieval approach. For instance, using a context-based similarity measure and semantic matching algorithms to select the most informative contents for the proposed recommendation. To this end, it required the integration of the implemented modules and functionality testing in the application context of street parking problems.

4.2.5 Selection of computational resources for the working environment

As posited by the title of the chapter, this research cycle aimed at the implementation of a demonstrative part of the active recommender framework. For the purpose of implementation, 4GLs were considered. There were however many candidate languages and programming environments available. To select the most appropriate one, an IEEE's survey on the top programming languages[1] was used, in addition to the consideration of the implementation requirements. The former included data from IEEE sources that used the above languages and programming environments for academic purposes. The survey of the IEEE ranked them based on a relative weighting and by combining eleven metrics from eight sources (e.g., Google Search, GitHub, and IEEE Library). The abovementioned website allowed us to rank the programming languages based on customized metrics and preferred data sources. In view of the requirements and expectations for programming, (i) the ranking was based on IEEE spectrum, (ii) data sources from IEEE Xplore digital library were selected, and (iii) language types used for enterprise, desktop, scientific applications and referenced in academic publications in the years 2019 and 2020 were preferred.

The list of the top ten programming languages is shown in Figure 4.3. From these, there were four languages belonging to the category of 4GLs, namely Python: (second ranked), R (fourth ranked), Processing (fifth ranked), and MATLAB (sixth ranked). Considered the required algorithms, the Processing® package was deemed not to be aligned with our application context. It is a software tool that is created to facilitate the development of visualization-oriented applications with an emphasis on animation. Thus, the candidate working environments to choose from were Python, R, and MATLAB. Table 4.3 shows the results of a comparison of the working environments from the perspective of the demonstrative implementation.

Our investigation considered not only the main features of the programming languages, but also if there were widely tested resources (for instance, built-in computational functions, component libraries, and toolboxes) included in the working environments. The candidate working environments were compared based on the appropriateness of the useable resources for the target implementation (i.e., natural language processing, process modelling, machine learning algorithms, and context-aware recommendation options as presented in Table 4.4. The brief descriptions of the abovementioned three working environments are as follow.

[1] https://spectrum.ieee.org/top-programming-languages/

Python® is a general-purpose programming language. It is an interpreted language (without compiling a program source code into machine-language instructions. Therefore, it is suited for web programming and data science.

R® is a programming language/environment that has been massively used in various businesses for statistical analytics and machine learning applications. R supports scripting-based application development and programming. The environment includes scripts that can be executed independently within various applications. The scripts can control the applications and can be used for the purpose of automation.

Figure 4.3: Top ten programming languages as used and referenced in the academic publications (source: [2])

MATLAB® is a procedural (imperative) programming environment mainly for matrix manipulations, implementing algorithms, plotting functions and data, and developing user interfaces. As a kernel concept, MATLAB interprets a procedure as a set of instructions that can be referenced through a procedure call. This helps developers in reusing the library functions and codes. This programming environment (language) is widely used for engineering calculations and simulation purposes because it is focused on mathematical procedures and models.

Table 4.3: Comparison of the working environments from the perspective of the demonstrative implementation

	Python	**R**	**MATLAB**
basic language	object-oriented programming language	interpreted language	math and matrix- oriented language
primary objective	general purpose	statistic software and data analysis	engineering and technical computing
functionalities	high-performance linear algebra, graphics, and statistics	statistical computing and graphics support	testing algorithm without the act of compiling software development tools
libraries	extensive support libraries	widely range packages	standard library, toolboxes, user's development libraries

Concerning the required algorithms for the demonstrative modules, there were no significant differences in the resources provided by the three programming languages/environments. In the context of application for the roadside parking problem, we found that the reusable functions/programs supported the simulation of the Ackerman steering. Several functions were available in the robotics system toolbox and the user's development library of the MATLAB for automatic parking. After a qualitative comparison of the three programming languages/environments, the MATLAB package was preferable for the demonstrative implementation and for the testing of the computational operation in the context of the target application.

4.3 Tactical issues of the demonstrative implementation

4.3.1 Fundamentals for the implementation dialogue-based obstacle identifier module

The implementation of the dialogue-based obstacle identifier module was based on three fundamental principles: (i) human-machine interaction through a dialogue; (ii) content-oriented decision table; and (iii) exact inference using pattern similarity measure. To identify an obstacle in the design process, the ARF communicates directly with a designer. It aims at recognizing the current state of design action by capturing the pattern of answers supplied by a designer. Natural language is the basis of the communication. Typically, task-based dialogue systems and Chatbots are used in conversational recommender systems [3]. The proposed task-based approaches were designed for a particular task and set up to have

Table 4.4: Comparison of usable resources for implementing the required algorithms

demo modules	Python	R	MATLAB
DOI module	string matching text processing services	text-process functionality	text similarity measures matrix-oriented operation distance-based similarity measure function
RPC module	network library similarity matrix	matrix operation	matrix-oriented operation graph and networks algorithms
ROI module	machine learning modeling graph visualization and analysis	machine learning modeling	statistical analysis and machine learning toolbox graph visualization and analysis
ACG module	string matching	text-process functionality	text analytic processing toolbox semantic-based algorithms
application context - APAS	N/A	N/A	Ackerman Kinematics-Robotics system toolbox Ackerman auto parking simulation – developer's library

short conversation to get information from a user in order to complete the task. Chatbots are systems designed for extended conversations, with the goal of mimicking the unstructured conversational characteristic of human-human interactions [4]. Therefore, to simplify the implementation of the DOI module, we preferred a task-oriented approach.

The ARF interacts with a designer through a multi-turn dialogue. Form-based user interface and structured texts for the input are commonly used in the application specific context. The very first question is regarding the current design action. The designer replies to the question with the term or phase identifying the design action. The text similarity measure is applied to match the answer with the design action available in the knowledge repository. String-based similarity is used to compare two terms. Each term is modelled as a set of tokens. The similarity between the two strings is assessed by manipulating sets of tokens such as terms, words, or phrases. Once the current design action is recognized, a set of questions will be retrieved. The number of questions is varied according to the actual design action. The designer follows the pre-defined dialogue path and replies to the questions with binary (i.e., positive or negative) answers. The combination of replies is defined as the pattern of answers and will be used as an input for the exact inference.

The design content-orientated decision table contains information to support decision-making concerning a particular design action. The pieces of information included in the table were derived from the condition patterns concerning the decision criteria. To build a decision table, we needed to (i) determine the maximum size of the possible solutions, (ii) eliminate any impossible situations concerning inconsistencies and redundancies of the conditions, and (iii) simplify the table with the potential solutions. A combination of different conditions, which corresponds to a certain solution, defines a pattern of decision conditions. To find the best solution, exact inferring is applied based on the similarity measure of the patterns. The similarity measure is a way of measuring how data samples are related or close to each other. It is usually expressed as a numerical value. The similarity value gets higher when the data samples are more alike. If two patterns were exactly the same, then the similarity value was set to 1, and the solution associated with the pattern of decision criteria was selected. Otherwise, no solution was found. It can be assumed that an obstacle occurred somewhere in the preceding design actions.

This was the basis of exact inference. Formally:

$$solution(j) = \begin{cases} true, & v_{ij} == 1 \\ false, & otherwise \end{cases} \tag{4-1}$$

where: *solution(j)* is the method usable for a particular design action (which corresponds to the pattern of decision variable j in the decision table), and v_{ij} is the similarity value between patterns i and j.

With the intention to calculate the similarity of the patterns, the decision conditions are converted into binary values stored in the decision matrix, $DM_i \leftarrow \{0,1\}^{(m \times n)}$, where (1) represents the considered criterion, and (0) means otherwise. The pattern similarity can be

expressed formally as:

$$v = 1 - sim\left(\vec{P}_d, \vec{P}_k\right) \tag{4-2}$$

where: v is similarity between $\vec{P}_d$ and $\vec{P}_k$, $\vec{P}_d \leftarrow \{0,1\}^n$ is a vector of answers supplied by a designer to a certain sequence of questions, where (1) means a positive answer, and (0) means a negative answer, $\vec{P}_k \leftarrow \{0,1\}^n$ is the vector of the considered pattern of binary value representing the condition of decision criteria in row k of DM_i, n is the total number of questions, and m is the total number of patterns in the decision table.

Calculated using this formula, the value of distance-based similarity is the smallest distance between each pair of points. The currently used metrics are: (i) cosine similarity – measuring the distance based on the cosine of the angle between two vectors projected onto a multi-dimensional space, (ii) the Manhattan distance – calculating the distance between two points based on the sum of the absolute difference of their Cartesian coordinates, (iii) the Euclidean distance – calculating the distance based on the square root of sum square of deviations between two coordinates, and (iv) the Minkowski-distance – expressing a generalization of the Euclidean and Manhattan distances. Considering the reusable resources that were available in the libraries, the principle of cosine similarity was used as the measure of similarity of the patterns. Symbolically:

$$cosin\_sim\left(\vec{P}_d, \vec{P}_k\right) = \frac{\sum_{i=1}^{n} P_{d,i} \times P_{k,i}}{\sqrt{\sum_{i=1}^{n} \left(P_{d,i}\right)^2} \sqrt{\sum_{i=1}^{n} \left(P_{k,i}\right)^2}} \tag{4-3}$$

4.3.2 Fundamentals for the implementation of reference process protocol creator module

The principal definition of the concept of the reference process protocol (RPP) was described in Section 3.4.6 in Chapter 3. There are other constitutional elements included in it: (i) the process flow model (PFM); (ii) the timed action model (TAM); and (iii) the decision tree model (DTM). The PFM is a fundamental element of the RPP to represent a design entity and its contents (including input-output data, computational method). In the implementation of the module, a design entity is a computational representation of a design action. As an element of a formalized design process, a design entity contains specific contents (which are distinct for each of them). A network of design entities is constructed by creating relationships among them in the TAM. If multiple relationships occur in the case of any design entity in the network, it defines a decision point, which may need decision-support. This support concerns the action of selecting the subsequent design entities and the most appropriate method.

The hybrid inference approach was used for the purpose of selecting the subsequent design entities and the most appropriate method. In these actions the RPP is utilized. Eventually, the hybrid inference is based on the combination of probabilistic reasoning by means of

the Bayesian networks and the model-based reasoning that relies on using the decision tree model. To be able to investigate a design process, the RPP was supposed to be known by the ARF. In this context, 'knowing' means having the specification of the relationships among the design entities, rather than knowing the outcome of completing the design entities.

4.3.2.1 Fundamentals to construct a process flow model

A process flow model (PFM) is a state-transition model of the design process, or part thereof. It is represented by a Petri-net like model which is represented by the incidence matrix. Symbolically:

$$PN = (S, T, C_{pre}, C_{post}) \tag{4-4}$$

where: $s_i \in S$ is a finite, non-zero set of n states, $t_j \in T$ is a finite, non-zero set of m transitions, C_{pre} is the input incidence matrix, $c^{pre}_{(i,j)} \leftarrow \{0,1\}$, and C_{post} is the output incidence matrix, $c^{post}_{(i,j)} \leftarrow \{0,1\}$. The incidence matrix representation of PFM is defined by:

$$C_{PFM} = (C_{post} - C_{pre})^{(n \times m)} \leftarrow \{-1,0,1\} \tag{4-5}$$

The elements of the matrix C_{PFM} represent flow relations of the net configuration of the process flow model, PFM, where $c^{pFM}_{(i,j)} \leftarrow \{-1,0,1\}$ is the directed arc $f_{(i,j)} \in F \leftarrow (S \times T) \cup (T \times S)$, which is defined by the following conditions:

$$f_{i,j} = \begin{cases} (s_i, t_j), if\ c^{PFM}_{i,j} = -1; \\ (t_j, s_i), if\ c^{PFM}_{i,j} = 1; \\ \emptyset,\ otherwise. \end{cases} \tag{4-6}$$

where: $i \in n$, and $j \in m$.

The Petri-net models provides various patterns for modelling the net configuration of the design process. The basic configuration of Petri-net is a *1-to-1* state machine, which comprises one input state and one output state connected by a transition. As shown in Figure 4.4, all sample types of net configurations are the extensions of a *1-to-1* state machine.

Using Petri-net model in the implementation of the demonstrative part of the ARF makes the challenge less complicated and allows keeping the resulting extension of the PFM feasible. A configuration of design actions is classified according to three considered patterns, where: $1 \leq m \leq 3$ as follows:

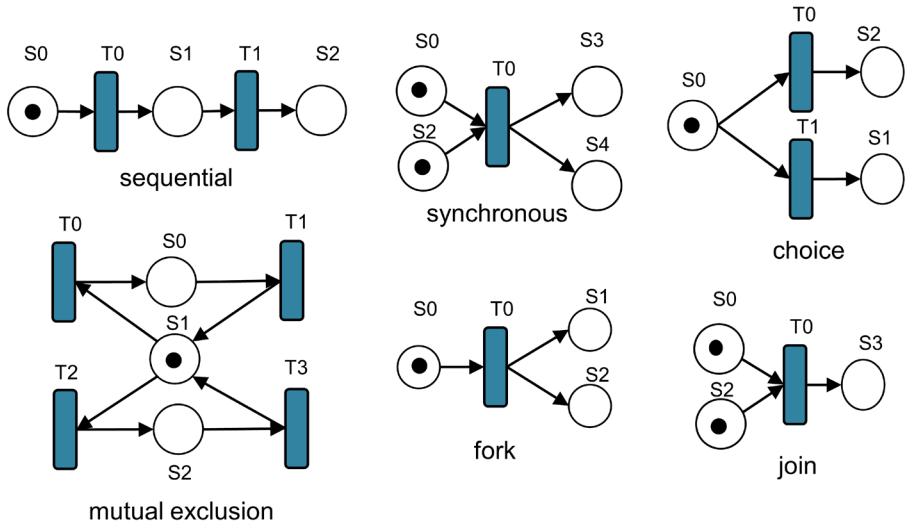


Figure 4.4: Different types of net configuration [5]

Pattern 1: 1-to-1 state machine configuration

$$S := \{s_{in}, s_{out}\},\ T := \{t_1\},\ F := \{(s_{in}, t_1),(t_1,s_{out})\} \tag{4-7}$$

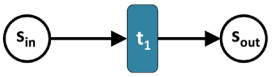


Figure 4.5: Fluent configuration of a state machine

Pattern 2: *n-to-1* synchronization, which consists of n input states and one output state connected by a transition

$$S := \{s^1_{in}, s^2_{in,\ldots,} s^n_{in,} s_{out}\},\ T := \{t_1\},$$
$$F := \{(s^1_{in}, t_1), (s^2_{in}, t_1),\ldots, (s^n_{in}, t_1), (t_1,s_{out})\} \tag{4-8}$$

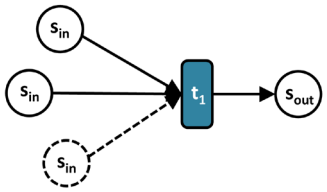


Figure 4.6: Confluent configuration of *n-to-1* synchronization

Pattern 3: *1-to-n* distribution, which consists of one input state and n output states connected by a transition

$$S := \{s_{in}, s^{1}_{out}, s^{2}_{out,\ldots,} s^{n}_{out,}\}, \quad T := \{t_1\},$$
$$F := \{(s_{in}, t_1), (t_1, s^{1}_{out}), (t_1, s^{2}_{out}), \ldots. (t_1, s^{n}_{out})\} \qquad (4\text{-}9)$$

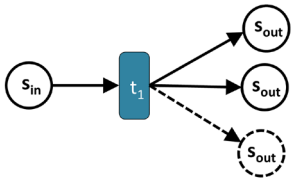


Figure 4.7: Configuration of *1-to-n* distribution

These patterns are the basic elements of a process flow model, which is extended by a connector. As a chosen transition, it is included in those cases where n number of input states of a process flow element are identical to n number of output states of the preceding ones.

$$S = \{s^{pre}_{out}, s^{post}_{in}\}, T = \{t_{con}\},$$
$$F = \{(s^{pre}_{out}, t_{con}), (t_{con}, s^{post}_{in})\}, s^{pre}_{out} \equiv s^{post}_{in} \qquad (4\text{-}10)$$

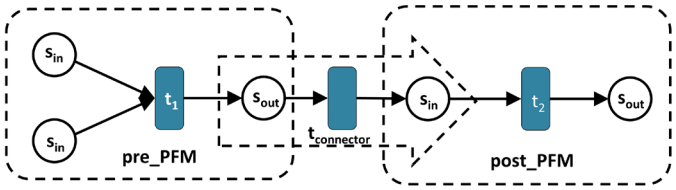


Figure 4.9: Configuration of PFM included an assembly of consecutive process flow elements with a connector

4.3.2.2 Fundamentals to construct a decision tree model

Decision trees (DTs) are widely used learning techniques to build classification models that closely resemble human cognition [6]. They mimic human thinking logic while making a decision. In turn, the designer is able to interpret and understand the logic behind the decision process by extracting the interpretable rules with their constraints [7]. Decision trees are widely recognized as interpretable models, which divide a complex classification task into several simpler ones. As presented in Figure 4.9, a simplified tree classifier can be implemented as a hierarchical tree structure, *G(V,E)*, where: $V \in \{V_1, V_2\}$ is a finite, non-empty set of nodes, V_1 is a set of leaf nodes containing the class value, and V_2 is the

set of intermediate nodes corresponding to one of the attributes. The set of edges E represents distinct attribute values. The user can reveal the decision-making process by following the tree structure. The decision model is created as a black box in other machine learning algorithms (e.g., in SVM and ANN). This means that the user, or even the designers of the decision algorithm, cannot understand how variables are being combined to make predictions [8]. This is a reason why the concept of a decision tree was selected as a decision support means for the RPP.

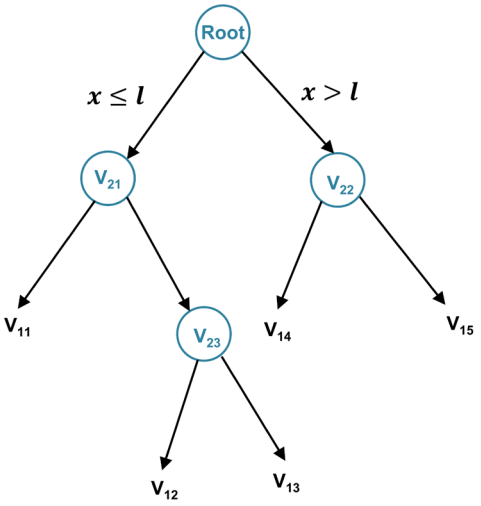


Figure 4.9: Simplified structure of decision tree

Because of their (i) flexibility, (ii) robustness to noise, (iii) the low computational cost for model construction, and (iv) the ability to learn from a small size of a training set, the decision tree induction (DTI) algorithms were preferred over the other learning algorithms [9]. The algorithm was used to build a tree structure through a series of binary splits from a root node via branches passing several decision nodes, until coming to the leaf nodes. Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions. Various DTI algorithms concerning the splitting criteria i.e., ID3, C4.5, CART, CHAID, QUEST have been development over years. The most widely used metrics for determining the splitting point are based upon information gain and the Gini Index [10]. The metrics measure the impurity in the candidate nodes and suggest how heterogeneous or homogeneous a given set of data is. If the impurities turn out to be zero, it meant that the data set was classified.

The information gain is an impurity-based criterion that measures the entropy of a dataset. It is interpreted relative to the attribute in the dataset, which is defined as the difference between the entropy of set S and the entropy of S_v under the given attribute conditions. The concept of the information gain was used in the algorithms ID3 and C4.5. The information gain was calculated using the following equation:

$$g(S,A) = H(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} H(S_v) \qquad (4\text{-}11)$$

where: *g(S,A)* is the information gain of attribute A concerning the dataset *S, V(A)* is the set of all possible values for attribute *A*, and S_v is the subset of *S* for which attribute A has value v. The information gain of learning on the decision tree is equivalent to the mutual information of classes and attributes in the dataset.

The concept of Entropy originates from the information theory, and it measures the amount of disorder and the unpredictability in a system. At the implementation of the decision tree,

it was defined as the expected value of information measuring the uncertain data present in the dataset.

$$H(S) = -\sum_{i=1}^{n} p_i \, log_2(p_i) \quad (4\text{-}12)$$

where: p_i is the probability of S belonging to class i.

The Gini Index (GI) is another impurity-based criterion that measures the divergence between the probability distributions of the target values of the attributes. It is used in the CART algorithm. The formula for a calculation of the Gini impurity is as follows:

$$I_G = 1 - \sum_{i=1}^{n} (p_i)^2 \quad (4\text{-}13)$$

where: I_G is the measure of the Gini impurity, n is a number of classes present in the nodes, and $p_i = n_i/n$ is the probability of class n.

Common evaluation metrics are used to evaluate the decision tree model, for instance, (i) classification-related metrics (e.g., precision, recall, accuracy), and (ii) statistical methods (e.g., root mean square error, mean absolute error). The choice of the evaluation metrics depends on the task given, for instance, (i) classification, (ii) regression, or (iii) clustering.

The rest of this section discusses the decision tree model for the classification task. The key concept of classification-related metrics is derived from the confusion matrix. It is a two-by-two table that contains four outcomes produced by a binary classifier [9]: (i) true positive (TP) means an outcome that the decision model correctly predicts the positive class, (ii) true negative (TN) is an outcome that the model correctly predicts the negative class, (iii) false positive (FP) is an outcome that the model incorrectly predicts the positive class, and (iv) false negative (FN) is an outcome that the model incorrectly predicts the negative class. A descriptive statistical processing of these outcomes provides a meaningful metrics as shown in Table 4.5. The evaluation measure aims to reach the operating point that minimizes misclassification rate [11]. Commonly, these metrics are used for a comparison purpose. To select the optimal model, it can decide based on multiple metrics for instance, speed, robustness, scalability, and rule structure [12].

4.3.2.3 Fundamentals to compose the reference process protocol

A reference process protocol is represented in the causal probabilistic network of design entities. The network consists of the interrelations of multiple sets of a sequence of concurrent design entities. Each sequence can be seen as a design activity flow that transforms and/or adds value to a set of inputs with the common purpose of developing an algorithm. The relationships of entities in the network are determined by the timed action model (TAM), which is organized according to two conditions: (i) compositional relationship – which considers the dependency of the output and input states of two subsequent design entities; and (ii) temporal relationships – which assumes that there are no compositional

Table 4.5: Metrics for evaluation of the classification performances

metrics	equations	description
Accuracy (A)	(TP+TN)/(TP+TN+FP+FN)	the ratio of correctness prediction to the total number of samples in a data set
Precision (P)	TP/(TP+FP)	the ratio of the number of correct positive predictions to the total number of positive predictions
Recall (R)	TP/(TP+FN)	the ratio of the number of correct positive predictions to the total number of correct predictions
F1-score (F)	(2×(P×R))/(P+R)	harmonic mean of precision and recall

relationships between two considered design entities.

In the latter context, the temporal arrangement of the design entities depends on the sequence of design tasks that they belong to. If these two conditions are fulfilled, then it is guaranteed that every single entity is placed in the right order in a TAM. However, it is possible that multiple relationships occur at any design entities in the network. That means that there are multiple alternatives to create a design activity flow. Without acquiring supplementary information from a designer, based on the probabilistic reasoning by means of a Bayesian network, the RPP provides the necessary decision support.

In the context of the RM design process, a graphical representation of the RPP was constructed in the form of a Bayesian network. The nodes of this network represented the design entities, $e_i \in \mathcal{E}$, and the directed arcs between the entities, $l_{(i,j)} \in L$, captured the conditional relationships. The probability distribution of the network was symbolical expressed as follows [13]:

$$p(e_i \in \mathcal{E}) = \prod_{i=1}^{n} p(e_i|e_{pre}) * p(e_0) \tag{4-14}$$

where: $p(e_i) \in P$ is the conditional probability of entity $e_i \in \mathcal{E}$, and $p(e_0)$ is the conditional probability of entity e_{pre}. The entity $e_0 \in \mathcal{E}$ represents a node of origin in the graph representing RPP. The backward probability distribution $p(e_0|e_{pre})$ is determined by the frequency of co-occurrence of any pair of entities in the historical processes.

$$p(e_i|e_{pre}) = \frac{freq(e_i,e_{pre})}{\sum_{i=1}^{n} freq(e_i,e_{pre})} \tag{4-15}$$

where: $freq(e_i, e_{pre})$ is the frequency of the co-occurrence of entities e_i, and $e_{pre} \in \mathcal{E}$.

In order to compose an RPP, the decision tree model had been trained and stored in the knowledge repository. Should there be a need for decision support because of the multiple options occurring at an arbitrary decision point, the decision tree model will be used to select the best support option.

4.3.3 Fundamentals for the implementation of reference process protocol-based procedural obstacle identifier module

4.3.3.1 Probabilistic inference by means of Bayesian network

Based on the RPP, this investigation of the design process is performed at two levels: (i) focusing on the design actions (in which case, the contents of individual design actions are investigated), and (ii) focusing on the design activity flow (in which case, the relationships of the design actions are investigated). To identify a procedural obstacle in the design process, the latter is exposed to backward reasoning in order to trace back the preceding design actions in the RPP. We interpreted this approach as a rationality-based event processing, which observes an obstacle in the design process based on the relationships of the elements in the RPP. According to our interpretation, the event processing is concerned with which design actions are possible to be done in the design process.

The fundamental principle of probabilistic inference is inferring the actual design activity flow based on the probability relationships of n design entities in the RPP. It is calculated by the joint probability distribution (JPD) of the segment of the process flow model, PFM. Fundamentally, a process flow model is a state-transition model of the design process. As such, it represents the actual (current) state of design process. The process flow model includes the possible sets of related design entities. Each set can be constructed as a design activity flow as the procedural basis of the procedural recommendation generation. The actual design activity flow is selected by the candidate segments with the highest value of JPD. Symbolically:

$$\mathbb{P}_{ctx} = max\ (p(\mathbb{P}_1), p(\mathbb{P}_2),\ldots, p(\mathbb{P}_n)) \tag{4-16}$$

where: $\mathbb{P}_{ctx}$ is the actual design activity flow, $p(\mathbb{P}_1)$ is the joint probability distribution of the segment of the process flow model, PFM, $\mathbb{P}_i = \{e_{i-(n-1)}, e_{i-(n-2)}, e_{i-(n-3)},\ldots,e_i\}$, n is the total number of entities in that segment, and $i > n$. Adapting the notation of [14], the joint probability distribution is calculated as follows:

$$p(\mathbb{P}_i) = \prod_{i=1}^{n} p(e_i|\, e_n) \tag{4-17}$$

where: $e_n \in \mathcal{E}$ is a preceding node of entity $e_i \in \mathcal{E}$

Let us suppose that the design entity e_{ctx} is present at Node e_{31} in the RPP. It is possible that the segments of actual design activity flow, $\mathbb{P}_{ctx}$, can be found in multiple pathways. This situation is shown in Figure 4.10. In this case there happen to be three options for candidate segments. They and their elements are described by Equation (4-18):

$$\mathbb{P}_{ctx} = opt \begin{cases} \mathbb{P}_1 := \{e_{11}, e_{22}, e_{31}\} \\ \mathbb{P}_2 := \{e_{12}, e_{21}, e_{31}\} \\ \mathbb{P}_3 := \{e_{13}, e_{22}, e_{31}\} \end{cases} \tag{4-18}$$

The possible preceding entities are present at Node e_{21} and Node e_{22}. With respect to Node e_{22}, there are two candidate entities e_0, which are present at Node e_{11} and Node e_{13}. Considering Node e_{21}, the entity e_0 is present at Node e_{12}. To find the best representative segment of $\mathbb{P}_{ctx}$, we applied probabilistic inference based on the joint probability distribution (JPD) of the segment of the PFM, as defined in Equation (4-16).

4.3.3.2 Hybrid inference for the generation of process-based recommendation

Hybrid inference combines probabilistic reasoning and model-based reasoning to infer the design entities and their proper methods, which should be included in the most informative PFM. Once a segment of the PFM is selected in a given context, the design entities included in the PFM are identified. To rectify the actual design activity flow, computational methods are selected for the current design entity and the design entity that precedes it. We employed the decision tree classifier to select the most appropriate method M for the considered design entities. Machine learning algorithms can generate predictive models on target classes for each test case of the datasets. Classification is the task of learning a target function that maps each attribute in a dataset to one of the predefined class labels.

The classifier acquires data in the form of $x_i \in X \longrightarrow \mathcal{M}$, where x_i is an attribute derived from the patterns of decision criteria corresponding to the class labels $\mathcal{M}$, and $m_i \in \mathcal{M}$ is the method for a given design entity. The classification process for prediction of usable methods can mathematically be described as a function:

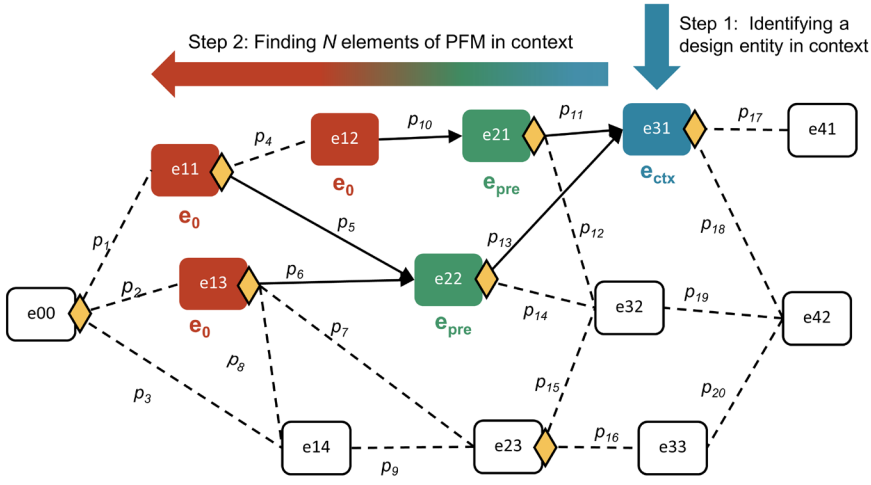


Figure 4.10: Backward reasoning for the investigation of the actual design activity flow ($e_{ctx} = e_{21}$, $n = 3$)

$$\mathcal{M} = cls\_f\left(\vec{P}_d, \Phi\right),\ \mathcal{M} \subset \widehat{\mathcal{M}} \quad (4\text{-}19)$$

where: $\mathcal{M}$ is a set of classes representing the usable method of the new sample, $\vec{P}_d$ is a vector of answers replied by the designer, <cls_f> is the classification function, Φ is the parameter set of the classification function, and $\widehat{\mathcal{M}}$ is the class label of the training set.

Initially, two design entities with the proper methods are composed. In the next step, the forward investigation for the next design entity is performed by using probabilistic reasoning. The candidate entities will be explored based on their relations to the current design entity in the RPP. The selection process includes the calculation of the joint probability distribution of the extended process flow model, which considers three design entities. The formula of the extended PFM, $\mathbb{P}_{ext}$, can be expressed symbolically as follows:

$$\mathbb{P}_{ext} := \{(\hat{e}_{pre}), (\hat{e}_{pre}),(e_{post})\} \quad (4\text{-}20)$$

where: $\mathbb{P}_{ext}$ is an extended PFM, $\hat{e}_{pre}$ is the preceding design entity, $\hat{e}_{ctx}$ is the current design entity, and e_{post} is the next design entity. As reference for the process-based recommendation generation, the extended process flow model is selected that is characterized by the highest value of joint probability distribution (JPD).

4.3.4 Fundamental for the implementation of advisory content generator module

The advisory contents were considered as sets of knowledge to support the execution of the design actions and the proposed method. Knowledge can be stored in various forms, such as (i) descriptive texts in a document, (ii) corpus of domain specific knowledge, (iii) structural representation of knowledge, and (iv) unstructured texts in webpages. In our case, the content-based recommendations are generated based on two sources: (i) the profile contents of the design action included in the data model, and (ii) the contents included in knowledge sources. For the demonstrative implementation of this module, we used webpages as knowledge sources. The fundamental concepts of the computational implementation are related to the information retrieval. In the pre-processing stage, information in the web pages has been processed and transformed into a recommendation item. The computational processes involved (i) extracting the raw texts, (ii) finding the main contents, (iii) creating a document, and (iv) indexing the document with key terms.

Text similarity measurement is widely used in text mining operations such as (i) searching and information retrieval, (ii) text classification, (iii) information extraction, and (iv) document clustering. Recommendations can be computed by several techniques, for instance, by estimating text similarity of two documents, and making queries on keywords. Measuring the similarity of two documents can be based on four different indicators: (i) string-based similarity operates on string sequence and character composition, (ii) corpus-based similarity determines the similarity between two concepts based on the information

extracted from a corpus, (iii) knowledge-based similarity uses information from semantic networks to identify the degree of words similarity, and (iv) hybrid text similarities aim to combine the mentioned methods to reach the better performance by adopt their benefits.

We preferred a hybrid method to generate advisory contents. The most informative contents are retrieved by making queries based on the key terms defining the usable method. The computational process executes the best matching algorithm (called BM25). It is one of the hybrid similarity techniques, which combine string-based and corpus-based similarities. The BM25 measures the frequency of documents in the collection, in which the given term has appeared. This measure is called inverse document frequency (IDF). The best matching algorithm is defined by the function:

$$\text{BM25}(D,Q) = \sum_{t \in Q} (IDF(t) \times W_{t,D}) \tag{4-21}$$

where: D is a document, Q is a query, and $IDF(t)$ is the measure of the inverse document frequency of the *t-th* term in the query terms. $IDF(t)$ can also be looked at as a statistical weight used to measure a kind of 'informativeness' of the term t in a text document collection and expressed as a weight factor $W_{t,D}$. For the BM25, the actual value of $IDF(t)$ is calculated by the following formula:

$$IDF(t) = log\ (N\text{-}df(t)+0.5)/(df(t)+0.5) \tag{4-22}$$

where: N is the number of documents in the input data, and df(t) is the number of documents in the input data containing each term. The weight factor, $W_{t,D}$, of $IDF(t)$ can be computed by the following formula:

$$W_{t,D} = \frac{(k_1+1)\times tf(t,D)}{tf(t,D)+k_1\left(1-b+b\left(\frac{|D|}{\bar{n}}\right)\right)} \tag{4-23}$$

where: $tf(t,D)$ is the frequency of appearance of the term (t) in a document (D), k_1 is the parameter that controls the scaling function between the term frequency of each matching terms and the final relevance score of a document-query pair, $k_1 > 0$, b is the parameter that captures how the length of a document affects the relevance score, $0 < b < 1$ [15].

4.4 Specification of the resources used for the working environment

4.4.1 Fundamental programing language

MATLAB is a proprietary programming environment often used for scientific research, in engineering projects, and numerical computation. It has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming to create the procedural programming. The demonstrative implementation of the ARF

follows the object-oriented and procedural techniques which involve (i) identifying the computation components, (ii) analyzing the shared characteristics of the components, and (iii) classifying the components based on their similarities. A computation component is constructed in an object-oriented programming. An object is specified in a proprietary formalism of the MATLAB. It includes a function with a parameter [output]. The function is a transformation of (inputs) by the logical operations of <function>. Thus, a formal expression of a computational component is as follows:

```
component [output] = function(inputs)          (4-24)
```

The construction of the computational process should take into account the states and the output-input transitions of multiple components to achieve the desired state. As evidenced above, we followed the computational programing style of MATLAB. To implement a module, the related computational components were constructed in the style (form) of procedural programming. Putting together everything, the legacy of MATLAB is proven by the facts that it (i) included effective resources for programming the algorithms, and (ii) facilitated the demonstrative implementation by its built-in functions and toolboxes.

4.4.2 Built-in functions

Matrix operations

The most basic MATLAB data structure is the matrix. A matrix is a two-dimensional, rectangular array of data elements arranged in rows and columns. The elements can be numbers, logical values (true or false), dates and times, and strings. Matrix operations follow the rules of linear algebra. The required size and shape of the inputs in relation to one another depends on the operation. In the demonstrative implementation, data structures of main variables, for example, decision matrix, Times action model, Process flow model, and reference process protocol are constructed in the matrix representations.

Data constructs

Various built-in functions were used for data modelling and knowledge repository building. The data model of the object, i.e., a design entity, and the recommendation item were constructed in a form of structure arrays, which consisted of multiple fields. Each field contains properties of an object which is distinct from each other. These data models are stored in the knowledge repository. The functions used for implementing these components were: <struct>, <cell2struct>, and <table2struct>.

Graph and network algorithms

A graph model *G(V,E)* comprises nodes, $v_i \in V$, and edges, $e_i \in E$. The structure of a graph (its network) is determined by the connections among the nodes. Each node represents an entity and each edge represents a connection between two nodes. The connections can be definitive or probabilistic. The reference process protocol was conceptualized as a causal probabilistic graph. The graph management functions of MATLAB were helpful to support the development of the algorithms related to the CRP and ROI module. Several built-in

functions could be used, for instance,

- <digraph> to construct a directed graph connecting nodes with directional edges
- <predecessors>/<successors> to find the preceding or successive nodes in directed graph
- <inedges>/<outedges> to count the number of incoming/outgoing edges from/to a node in a directed graph
- <shortestpath> to find the shortest path between two nodes.

4.4.3 Applied toolboxes

Text analytic processing toolbox

The text analytic processing (TAP) toolbox provides algorithms and visualizations for pre-processing, analyzing, and modelling text data. The TAP includes tools for processing raw text from several sources such as equipment logs, web services, surveys, and operator reports. The built-in functions of the TAP toolbox can support various tasks, for example: (i) extracting text from files, (ii) extracting individual words, (iii) converting text into numerical representation, (iv) computing textual similarity, (v) conducting sentimental analysis, and (vi) building statistical models. The functions used in the implementation of the ARF were:

- text similarity measure functions (which offer several options for computing textual similarity of contents in a document and contents in queries, viz., *tfidf*, PageRank, and best matching (bm25))
- text mining functions (viz., <tokenizedDocument>, <BagOfWord>, <topkword>
- document generation functions (viz., <joinWords>, <removeStopWords>, <extractSummary>.

Statistical analysis and machine-learning toolbox

The statistical analysis and machine-learning toolbox supports the implementation of components that are needed for the construction of a decision tree model. The relevant functions are as follows:

- to train the binary classification tree, <fitctree>
- to compute the cross-validated classification error, <cvloss>
- to predict the usable method using the classification tree, <predict>.

Robotics system toolbox

The robotics system toolbox includes tools, functions, and algorithms for designing, simulating, and testing mobile robots. We found that some of these resources can be applied to simulate parking scenarios as well as to test the operation of the implemented modules. For instance:

- to create a car-like model that uses Ackerman steering, <AckermannKinematics>
- to find an obstacle-free path between start and goal locations within roadmap path planner, <findpath>.

4.4.4 Library of user development functions

Ackerman auto parking

The program implementing the Ackerman auto parking was developed by Khaled [16]. It is able to simulate roadside parking and to use the Ackerman steering functions. Both the program and its functions can potentially be modified for various parking scenarios. This is an important opportunity from the viewpoint of testing the functionality of the demonstrative implementation of the ARF. The simulation is able to show the motion paths and to ensure that a parking plan selected by the developed algorithm is working properly.

4.5 Specification of the implementation of the demonstrative modules

4.5.1 Architecting and implementation the dialogue-based obstacle identifier module

The dialogue-based obstacle identifier (DOI) module comprises three interrelated sub-modules, namely: (i) sub-module 2.10 – design action identifier, which contains two computational components (2.11-2.12); (ii) sub-module 2.20 – knowledge manager, which comprises three interrelated computational components (2.21-2.23); (iii) sub-module 2.30 – dialogue manager which comprises two computational components (2.31-2.32). Figure 4.11 shows the interrelationships of the computational components within the DOI module. Altogether, eight algorithms were implementation for the DOI module. Here we discuss only two algorithms: Algorithm A2.07 – '*organize a dialogue*' and Algorithm A2.08 – '*execute pattern matching*'. The list of variables used in the computational components included in the DOI module is shown in Table 4.6.

4.5.1.1 Sub-module 2.10: Design action identifier

The design action identifier sub-module includes the computational components 2.11 and 2.12. It aims at activating a dialogue and obtaining information directly from the designer to identify the current design action. The goal is to introduce a purposeful change in a given state of the design activity flow.

Component 2.11 captures and evaluates the designer's response and activates a dialogue. The input data consists of (i) an event which represents a moment in time when the ARF recognizes the pattern of designer's facial expressions, (ii) the response of the designer to inquiry of the ARF. The computational process obtains the response from the designer and returns a Boolean parameter, which is true if the designer accepts the offer, and returns false otherwise. It is an entry point where the ARF interacts with the designer.

Component 2.12 identifies a design action in a given context. When a dialogue is activated, the ARF poses the first question to the designer. It asks about the design action, which the designer was working on. When the designer replies, the response is used to identify the related design entity (which is a knowledge-based representation of a design action).

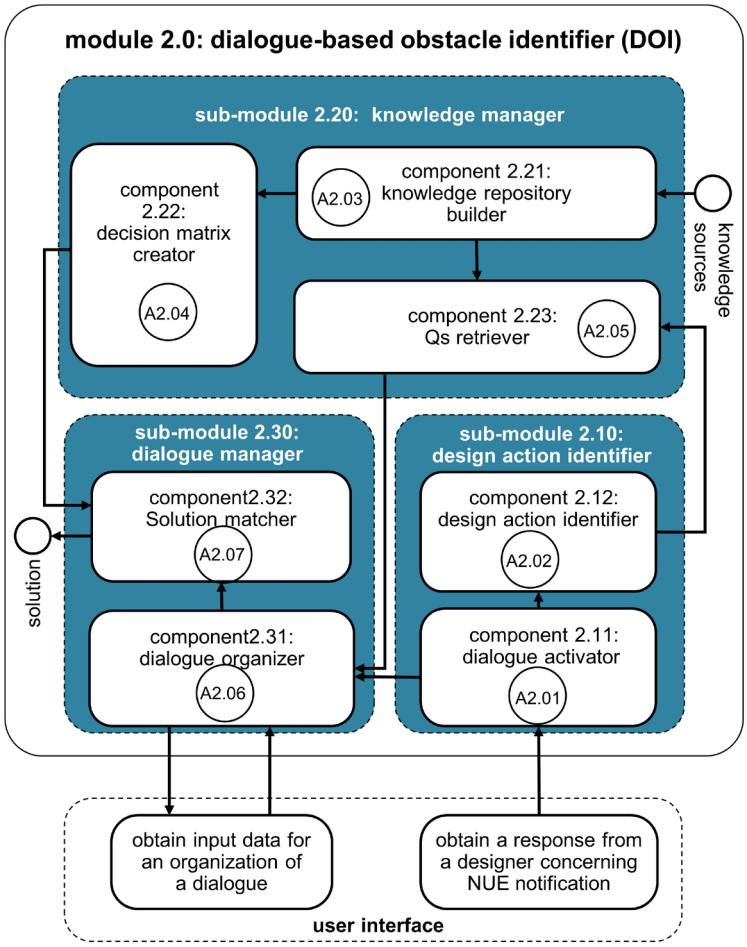


Figure 4.11: The interrelationships of the computational components of the DOI module

The designer's answer is captured as a descriptive text in the form-based user interface. The technique of token-based similarity is applied to find a design entity. It compares the similarity of the terms in order to identify the design action. The identifier of design entity is stored in the database. We used the built-in function <contains> of MATLAB to determine the similarity of the terms. The function returns *1* (true) if the term denoting the design action is found in the recorded name of a design entity and returns *0* (false) otherwise.

4.5.1.2 Sub-module 2.20: Knowledge handler

The knowledge handler sub-module includes three components (viz., components 2.21 – 2.23), which are interrelated. Component 2.21 generates a repository of knowledge elements related to the interpretation of a design action. This component aims at handling the domain-specific knowledge, which is needed to find the equivalent design entity. The input is a data table, which contains three knowledge elements: (i) a finite set of questions, (ii)

a collection of lookup tables containing a set of decision criteria, and (iii) the identified conditions and the potential solutions. The computational process of generating the repository involves the mapping of the relationships of the knowledge elements in the data table into the relational database.

Component 2.22 constructs a decision matrix to capture the binary values representing the conditions of the decisional options. The matrix captures the patterns of conditions to derive potential solutions based on the contents of the lookup table. Exact inference and static knowledge representation are used for finding a solution. The computational process completed by component 2.22 can be expressed symbolically so as:

Table 4.6: List of variables used in the computational components included in the DOI module

variables	description
e_{ctx}	design entity in context
$\mathcal{E}$	finite set of design entities available in the knowledge repository
Q_s	collection of questions available in the knowledge repository
c_n	decision criteria
DM_i	decision matrix
dD_i	vector of answers replied by a designer
$p_{i,j}$	identified conditions of decision criteria
s_m	possible solutions
s_{best}	the best matching solution

```
[DEC_M] = capture_DecisionCond(LUT)
```
(4-25)

where: `[DEC_M]` is a matrix containing possible patterns of conditions corresponding to the decisional options, and `[LUT]` is a collection of lookup tables. The lookup table is manipulated to capture the decision conditions. It can be expressed in a matrix form as follows:

$$\begin{bmatrix} s_1 \\ \vdots \\ s_m \end{bmatrix} = [DM_i] * \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} \tag{4-26}$$

where: $s_m \in S$ is a finite, non-zero set of potential solutions, $c_n \in C$ is a finite, non-zero set of decision criteria, and $[DM_i]$ is the decision matrix for a design entity $e_i \in \mathcal{E}$. Mathematically, it is specified as follows:

$$[DM_i] = \begin{bmatrix} p_{(1,1)} & \cdots & p_{(1,n)} \\ \vdots & \ddots & \vdots \\ p_{(m,1)} & \cdots & p_{(m,n)} \end{bmatrix} \tag{4-27}$$

Each element of this matrix, $p_{(i,j)}$, represents a condition {0,1} for the decision variable C_i, so as $p_{(i,j)} \leftarrow \{0,1\}$, where: (*1*) indicates that the condition is considered at selecting the potential solutions, and (*0*) means that the condition is not considered. Each row represents a pattern of identified conditions for a certain set of decision variable. The size of the matrix is $m \times n$, where: m is the total number of patterns of the decision conditions, and n is the

total number of the decision variables.

Component 2.23 retrieves a set of questions that are related to a design entity in a given context. Whenever a design entity is recognized in the given context, the ARF will search and retrieve sets of relevant questions as contents for making a dialogue.

4.5.1.3 Sub-module 2.30: Dialogue manager

The sub-module 2.30 includes the computational components 2.31 and 2.32. Each component serves a specific purpose, but they also interact. The goal of their interaction is (i) to organize a dialogue, (ii)to capture the pattern of responses given by the designer to the questions, and (iii) to find the best matching solution, which corresponds to the pattern of decision conditions presented in the decision matrix.

Component 2.31 organizes the dialogue between the designer and the ARF. The related context information is derived based on the set of questions and collected as structured information from the lookup table. The objective of this process is to capture the pattern of designer's responses to a set of decision options. Symbolically:

```
[dD] = organizeDial(entQs, designerSays)      (4-28)
```

where: `[dD]` is a vector capturing the pattern of the designer's answers. The size of this vector is equal to the total number of questions. The dialogue is organized by the algorithm A2.06 – '*organize a dialogue*', which manages the sequence of questions and captures the pattern of designer's answers. Computationally, the input data is a finite number of questions (*Qs*) and the responses (answers) by the designer [`DesignerSays`]. The total number of questions is related to the number of decision options in the lookup table for a particular design entity. The output data is a vector, $dD_i \leftarrow \{0,1\}$, where: (*1*) means that the received answer is '*yes*', and (*0*) means that it is '*no*'.

Component 2.32 finds the best match between the designer's responses and the set of patterns of the decision conditions in the decision matrix. Symbolically:

```
[SB] = findBSolution(DEC_M, dD)               (4-29)
```

where: `[SB]` is the best matching solution, which is retrieved from a set of solutions corresponding to the pattern of conditions in the decision matrix. We used algorithm A2.07 to calculate the similarity of the patterns of answers provided by the designer and the pattern of decision conditions. The best matching solution is found if the similarity value is equal to 1. This means that the patterns of answers and the patterns of decision conditions should be exactly the same. Otherwise, the algorithm A2.07 concludes that no potential solution was found. The algorithm finds a set of design entities in the repository that correspond to the concerned design entity. The results are input data for the ROI module and help further investigate the obstacle occurring in the design process.

ARF Algorithm A2.06: organize a dialogue

Input:	e_{ctx}:	design entity in context
	Qs:	collection of questions stored in the knowledge repository
	DesignerSays:	binary input from a designer ('*yes*' or '*no*')
Output:	*dD:*	vector representing designer's responses {0,1}

```
1:  Qe = Qs(e_ctx)        % Retrieving a set of questions related to the design entity in context
                            from the knowledge repository
2:  N_q ← total number of questions (Qe)
3:  for i = 1 to N_q do
4:      repliedQ ← Qe(i)
5:      designerSays ← [yes or no]
6:      if designerSays == [yes] do
7:          d (i) ← (1)        % Capturing a designer's response 'yes' to a
                                 returned question by 1
8:      else designerSays == [no] do
9:          d (i) ← (0)        % Capturing a designer's response 'no' to a
                                 returned question by 0
10:     end if
11:     dD (i) = d
12: end for
13: Return vector (dD)
```

4.5.2 Architecting and implementation of the reference process protocol creator module

The reference process protocol creator (RPC) module has been developed for handling process related knowledge in the reference protocol. Figure 4.12 shows the architecture and the interrelationships of the components of the module. The computational processes were divided into two groups. One group was mainly based on the provision of RPP contents. It focused on the principles of gaining information about the constituting elements of RPP. They were processed in the sub-modules 3.10-3.30.

Another group was for the composition of constituent elements of the reference protocol. It was operationalized in the sub-module 3.40. The implementation of the whole module required thirteen algorithms in total. Four algorithms, including A3.10-A3.13, were the main constituents for this module. The detailed descriptions of these algorithms are given in this section. The variables used for the computational components are listed in Table 4.7.

ARF Algorithm A2.07: execute pattern matching

Input:	*dD:*	designer's responses to the returned questions
	$\mathcal{E}$*:*	design entities available in the repository
	LUT:	lookup table containing corresponding solutions s_i
	DM:	decision matrix
Output:	s_{best}*:*	the best solution (with a maximum similarity value)
	e_id:	Index identifying design entities in context

```
1:  M ← total number of rows in DM
2:  for i = 1 to M do
3:      P(i)) = DM (i,[1:end])          % Capture a pattern of decision criteria from the
                                          decision matrix
4:      v ← cosineSimilarity (dD, P(i))
5:      V(k) ← v
6:  end for
7:  [idx] ← find (V(i) == 1)
8:  if isempty (idx) do
9:      Ne← total number of design entities in the knowledge repository
10:     for j = 1 to Ne do
11:         id ← contains (e_ctx, Ɛ(j))
12:         e_id(j) ← id
13:     end for
14: Return vector (e_id)
15: else do
16:     s_best ← LUT (idx)          % Retrieve the corresponding solution #idx from the
                                      lookup table
17: end if
18: Return s_best
```

4.5.2.1 Sub-module 3.10: e-Data model builder

In the computational process of constructing the reference process protocol, we defined a design entity as a computational representation of a design action. It is a data model, which contains the data representing the properties of a design action. This sub-module is intended to handle sets of knowledge related to a design entity. It has two computational components: the first component is for modelling an individual design entity, the other is for construction of a repository to store design entities.

Component 3.11 creates a data model to handle the contents of a design action. The data model is composed of five elements, which are formally described as follows:

```
[entD] = mdl_EntD(task,Id,inP,outP,mth,dT)   (4-30)
```

where: `[Task]` is a design task, `[ID]` is an identifier of a design entity represented as a textual description of a design action, `[inP]` is a set of input parameters, [outP] is a set

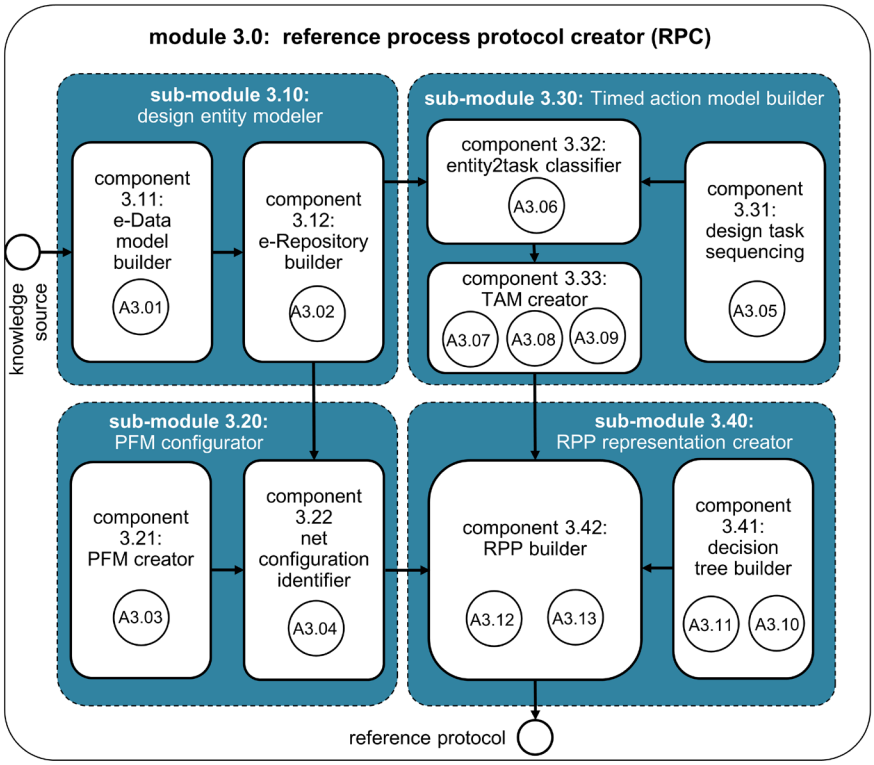


Figure 4.12: Interrelation of computational components of RPC module

of output parameters related to `[Mth]`, `[dt]` is a duration of completing a design action, which is computationally executed by a method `[Mth]`, and `[entD]` is a data model of a design entity.

The purpose of component 3.12 is to construct a repository of the design entities. The computational implementation is symbolically expressed as:

$$[\text{Repo\_entD}] = \text{genRentD}(\text{entD}) \tag{4-31}$$

where: `[Repo_entD]` is the location of the storage of knowledge resources for construction of a reference process protocol. It contains a finite, non-zero set of design entities $e_i \in \mathcal{E}$.

4.5.2.2 Sub-module 3.20: Process flow model configurator

A process flow model (PFM) is a state-transition model of a design process. Its primary element is represented by a design entity. The sub-module 3.20 uses a Petri-net-like structure to represent a PFM and identifies its pattern in order to classify the configuration of design entities. There are two interrelated components in this sub-module, component 3.21 and component 3.22.

Component 3.21 creates a representation of the process flow model, which is, as mentioned above, a Petri net-like structure. The model comprises three basic elements that are: (i) a finite set of states *S*, (ii) a finite set of transitions *T*, and (iii) a set of arcs *F*. The computational function <create_PN> finds a deviation of incidence matrix $C_{PFM} \leftarrow (C_{post} - C_{pre})$, and assigns a set of markings $m_0 \leftarrow \{0,1\}$ to the states indicating the initial behavior of the model. Formally:

```
[PFM] = create_PN(Pre,
          Post, M0)   (4-32)
```

where: [Pre] is an input incidence matrix, and [Post] is an output incidence matrix. The dimensions of the matrices are $m \times n$, where m is the total number of the transitions, and n is the total number of the states, [M0] is the initial markings on each state, and [PFM] represents a process flow model.

In this representation: $s_i \in S$ is a set of states, $t_j \in T$ is a set of transitions, and $f_{(i,j)} \in F$ is a set of arcs connecting a state and a transition, and vice versa. An arc expresses the flow relation, which is captured based on the contents of the incidence matrix: $c_{(i,j)} \leftarrow \{-1,0,1\}$.

$$PFM = \begin{bmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{bmatrix} * \begin{bmatrix} m_0^1 \\ \vdots \\ m_0^n \end{bmatrix} \quad (4\text{-}33)$$

Component 3.22 identifies a structure of design entity based on the patterns of Petri-net configurations. A design entity is an element of the process flow model.

Table 4.7: List of variables used in the computational components included in the RPC module

variables	description
$\mathfrak{D}$	finite set of design tasks
d_i	design task *i*
$\mathbb{D}$	design process
e_i	design entity *i*
$e^p{}_i$	design entity *i* associated with a certain pattern *p* of a PFM
$e^t{}_i$	design entity *i* associated with a certain design task *t*
$\mathcal{E}$	finite set of design entities stored in the knowledge repository
$f_{i,j}$	flow relation of state *i* and transition *j*
s_i	state *i* of a PFM
t_i	transition *i* of a PFM
$\mathbb{M}$	finite set of patterns of PFM
S	finite set of states of a PFM
T	finite set of transition of a PFM
F	finite set of flow relations of a PFM
$\mathcal{M}$	finite set of computational methods
m_i	method for an execution of design entity *i*
dT	decision tree model
$\mathbb{P}$	process flow model
$\mathbb{R}$	reference process protocol
extR(e_i, e_j)	external relation of (e_i, e_j)
intR(e_i)	internal relation of e_i
$seq(e^t{}_i, e^k{}_i)$	temporal relation of (e_i, e_j) concerning a sequence of design tasks (d_t, d_k)
$freq(e_i, e_j)$	frequency of co-occurrences of (e_i, e_j)

This component identifies a certain pattern of design entities, which can be composed into a process flow model. Symbolically:

$$[\text{ePFM}] = \text{classify\_PN}(\text{entD}) \quad (4\text{-}34)$$

where: [ePFM] is a finite non-zero set of design entities, which were identified by certain patterns of the Petri-net configuration $e^{p}_{i} \leftarrow (e_i, \mathrm{m}_p)$, and $\mathrm{m}_p \in \mathbb{M}$ is a set of the defined patterns of the Petri-net configuration. Figure 4.13 shows an example of net configuration of a process flow model created by using Algorithm A3.03.

4.5.2.3 Sub-module 3.30: Timed action model builder

A timed action model (TAM) is a time-stamped arrangement of design entities in a design process. It is an element of the reference process protocol, which aims at representing (and controlling) a sequence of design entities in a design process. The construction of a timed action model consists of five interrelated components, which are 3.31-3.35. They are implemented as follows:

Component 3.31 organizes the sequence of design tasks. This organization process aims at clarifying a set of design tasks for a given design process and finding the temporal relationships among design tasks. Their relationships allow the ARF to find a sequence of these tasks. The computational implementation can symbolically be expressed as:

$$[\text{SeqTask}] = \text{sequence\_Tasks}(\text{D\_Task}, \text{rules}) \quad (4\text{-}35)$$

where: [D_task] is a finite, non-zero set of design tasks, $d_t \in \mathfrak{D}$, and [rules] is a finite, non-zero set of rules. Each rule has a Horn's clause form of $r_i : p_1 \cap p_2 \ldots \cap p_m \rightarrow c_i$ where $p_1 \in P$ is the condition part of the rule, $r_i \in R$, $c_i \in C$ is a conclusion defining a temporal dependence relation of a pairwise design tasks $d_t, d_{t+n} \in \mathfrak{D}$ and $t,n \geq 1$. [SeqTask] contains a set of conclusions concerning the relations of the considered design tasks.

Component 3.32 classifies a design entity according to its relevance to design tasks. This process assigns the design entities to a given task whose completion needs them. Each task has a chronological order in the design process. To complete the design process, the designer may take actions step-by-step, following the sequence of the design tasks, if both the starting task and ending task are defined. Without consideration of the composition relationships between the design entities, the temporal relationship regulates the places

```
  PN_name: '2-to-1 synchronization'
set_of_Ps: {'State01'  'State02'  'State03'}
set_of_Ts: {'entD'}
set_of_As: {'State01'  'entD'  [1]  'State02'  'entD'  [1]  'entD'  'State03'  [1]}
```

Figure 4.13: Information representing a configuration of process flow model – '*2-to-1* synchronization'

of design entities in a design activity flow (i.e. it determines which one should be placed before or after another one). The component implementing this is formally expressed as follows:

$$\texttt{[entD\_T] = catg\_entD(D\_Task, entD)} \tag{4-36}$$

where: `[entD_T]` is a finite, non-zero set of design entities associated with a certain type of design task, $e^t_i \leftarrow (e_i, d_t)$.

Component 3.33 constructs a matrix to represent temporal relations of design entities. The matrix contains all numerical data needed to capture the arrangement of design entities with regard to the possible sequences of the design tasks. It is a computational representation for the construction of a timed action model. Symbolically:

$$\texttt{[eTM] = const\_ETM(SeqTask, entD\_T)} \tag{4-37}$$

where: [eTM] is a square matrix, $n \times n$, identifying an arrangement of a couple of design entities, and N_e is total number of design entities. The relationship of the entities in the matrix $q_{i,j} \leftarrow \{0,1\}$ follows the rule:

$$r_1 : p_1 \cap p_2 \rightarrow c_1 : d_t \prec d_{t+n} \tag{4-38}$$

$$q_{i,j} = seq\left(e_i^t, e_j^{t+n}\right) = \begin{cases} 1, & p_1 \cap p_2 = true; \\ 0, & otherwise. \end{cases} \tag{4-39}$$

where: p_1 and p_2 is a finite, non-zero set of design tasks d_t, $d_{t+n} \in \mathfrak{D}$ for a design process $\mathbb{D}$ and $t, n \in \mathbb{N}$

Component 3.34 constructs a matrix to represent the composition relationships of the design entities. It aims at capturing the (logical) relations of the entities and converting them into a computational representation. Formally:

$$\text{[eRM] = const\_ERM(entD)} \tag{4-40}$$

where: [eRM] is an entity-relation matrix of dimension $n \times n$. This matrix contains all composition relationships between the couples of design entities (e_i, e_j). There are three types of entity relations accounted for, namely: (i) internal relation (which refers to a transition from an input state to an output state established by means of an entity e_i); (ii) external relation (which exists in the case where an output variable of an entity e_i is identical to an input variable of an entity e_j); and (iii) no relation (which means that the entity couple made up by e_i and e_j has no direct relationship). The relationship of the entities $\varrho_{i,j} \leftarrow \{0,1,2\}$ in the matrix is defined by the following conditions:

$$\varrho_{i,j} = R(e_i, e_j) = \begin{cases} 1, if\ intR(e_i, e_j) = true; \\ 2, if\ extR(e_i, e_j) = true\ ; \\ 0\ ,\ otherwise \end{cases} \quad (4\text{-}41)$$

where: $intR(e_i, e_j)$ represents an internal relation of design entities i and j, and $extR(e_i, e_j)$ represents an external relation of design entities i and j, where $i \neq j$, and $i, j \in N_e$.

Component 3.35 constructs a matrix to represent a timed action model. It is a square matrix, which specifies the relationships of design entities according to the sequence of design tasks and the external interactions. Formally:

$$\texttt{[TAM] = construct\_TAM(eRM, eTM)} \quad (4\text{-}42)$$

where: `[TAM]` is a square matrix representing a timed action model, whose rows and columns are design entities, $e_i, e_j \in \mathcal{E}$. The dimension of the matrix is $n \times n$, where n is total number of the considered design entities, and $i,j \in n$. In a TAM, the entity composition relationships $t_{i,j} \leftarrow \{0,1\}$ are defined by the external relations of the entities e_i and e_j. These entities are considered as the intended actions to complete design tasks d_t and d_{t+n}, respectively.

$$\mathrm{t}_{i,j} = \begin{cases} 1, if\ extR(e_i, e_j) = 1\ and\ seq(e_i^t, e_j^{t+n}) = 1,\ t, n \geq 1; \\ 0,\ otherwise \end{cases} \quad (4\text{-}43)$$

where: $extR(e_i, e_j) \leftarrow \{0,1\}$ is the external relation of e_i and e_j. It is equal to *1*, when entities e_i and e_j have an external type of compositional relationship with each other, otherwise it equals to *0*. The formula $seq(e^t_i, e^{t+n}_j) \leftarrow \{0,1\}$ represents a timed transition of e_i and e_j. It equals to *1*, when (i) e_i belongs to a design task t_t and e_j belongs to a design task t_{t+n}, and (ii) these two design tasks have a temporal dependence relation, otherwise it equals to *0*. A timed action model is constructed by using the algorithm A3.09. The matrix representing a TAM is shown in Figure 4.14.

4.5.2.4 Sub-module 3.40: Reference process protocol creator

The sub-module 3.40 comprises three interrelated components (3.41-3.43). The component 3.41 develops a design tree classifier as one of the main components of the reference protocol. The classifier plays a crucial role in the process of design support. It selects the most appropriate method concerning a design action and presents it to the designer as a recommendation. Component 3.42 is the main contributor to the implementation of the proposed ARF. The reason is our fundamental assumption that a mathematical representation of a reference protocol can be generated for the computational implementation of provisioning process-context dependent recommendations. Component 3.43 is designed to visualize a graphical model of the reference protocol for the purpose of communication.

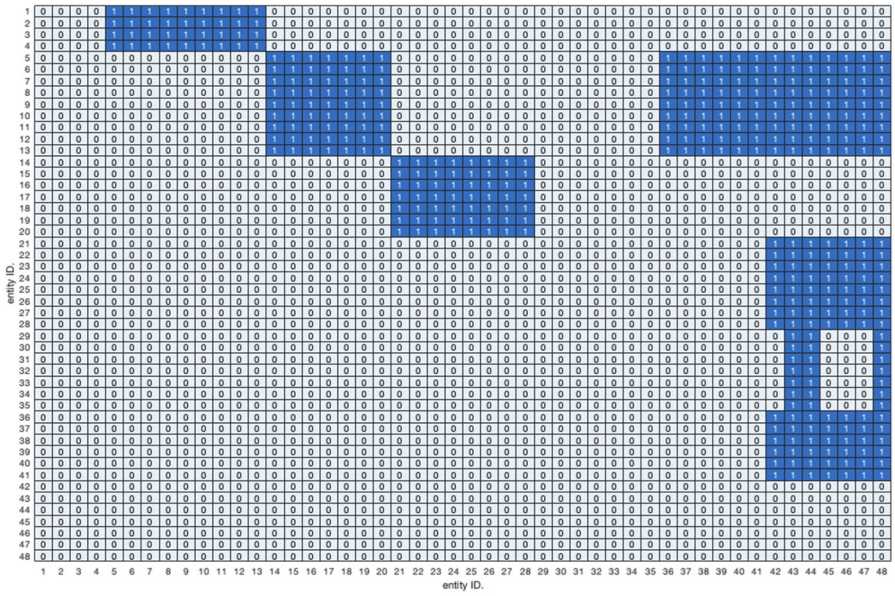


Figure 4.14: Matrix representing a Timed Action Model (n = 48)

The component 3.41 constructs a decision tree model to allow selecting the most appropriate method for execution of a design action. This component uses a tree-based non-parametric supervised learning approach for classification of the considered methods. The decision tree model is trained from the training dataset, which is formed by varying the features of the characteristics of the predictors. The component 3.41 employs a learning algorithm (LA) for the identification of the classification model. This LA is supposed to provide the best fitting relationship between a set of predictors and the associated class labels. The computational component is expressed symbolically as:

$$\texttt{[DTM] = constr\_DTM(Mth, DR, DF, S)} \qquad (4\text{-}44)$$

where: `[DTM]` is a decision tree classifier, and `[Mth]` is a set of methods usable for a computational execution of design action.

Each method is identified as a response to a training set. `[DR]` is a set of decision rules in the form of IF-THEN statements. Each rule contains one condition or multiple, conditions, that are associated with a prediction. `[DF]` is a set of data features. Each feature represents a decision condition. `[S]` is a training set containing the instances of the data features.

The component 3.41 employs the algorithm A3.10 '*training a decision tree classifier*' to train the model. A decision tree was constructed based on an inductive method as presented in [7]. A cross-validation technique is applied to determine the accuracy of the model. The training set is partitioned into K folds. The prediction accuracy is calculated by the

ARF Algorithm A3.10: train a decision tree classifier

Input:	S:	Training set with features $x_i \in X$
Output:	$d\mathcal{T}_{best}$:	Best performing decision tree classifier

```
k ← number of partitions of training set S
S ← {s1, s2, s3, ... sk}                    % Partition training set S into K folds
si ← test set i ∈ k
dT ← {}
for i = 1 to k do
    St ← S − si
    dti ← DTI (St, X, L)                    % Construct a decision tree by using Algorithm 3.11
    Stest ← si
    A ← accuracy (dti, Stest)               % Determine the prediction accuracy of tree(i)
    Accuracy (i) ← A
end for
dT ← { dti, dtj ... ....dtk}
Accuracy (idx) ← max (Accuracy (i), ...., Accuracy (k))
dTbest ← dtidx
Return dTbest
```

following equation.

$$A_i = \frac{c_{i,i} + \left(\sum_{j=1}^{N}\sum_{k=1}^{N} c_{j,k} - \sum_{j=1}^{N}(c_{i,j}+c_{j,i}) - 2c_{i,i}\right)}{\sum_{j=1}^{N}\sum_{k=1}^{N} c_{j,k}} \qquad (4\text{-}45)$$

where: A_i is the prediction accuracy of the model, N is the number of class labels, and $c_{i,j}$ is the total number of the predicted class i that was classified to the actual class j.

The ARF Algorithm A3.11, '*Decision tree induction*', creates a tree-like structure that classifies the instances in one of the given classes by learning some decision rules deduced from the data features, or a conditional probability distribution defined on the features and classes. The algorithm starts with a training set and an empty tree. In the first step, the feature which best splits the training data will be determined as the root node of the tree. Selecting the best split is based on the degree of impurity of the child nodes. The training set s_i is partitioned with the aim of having each of the partitions as pure as possible. In the demonstrative implementation, the Gini impurity measure is applied as the splitting criteria due to the fact that the response to the training data is categorical.

The instances are tested according to the splitting criteria and assigned to the child node, recursively, until it reaches the leaf node, which sorts the instance into the respective class. The algorithm terminates if any one of the following three situations occurs: (i) all training sample are of the same class, $s_i \in L$; (ii) the current feature set is empty, X={∅}; (iii) no training samples exist, S={∅}. A detailed description of the Algorithm A3.11 is as follows:

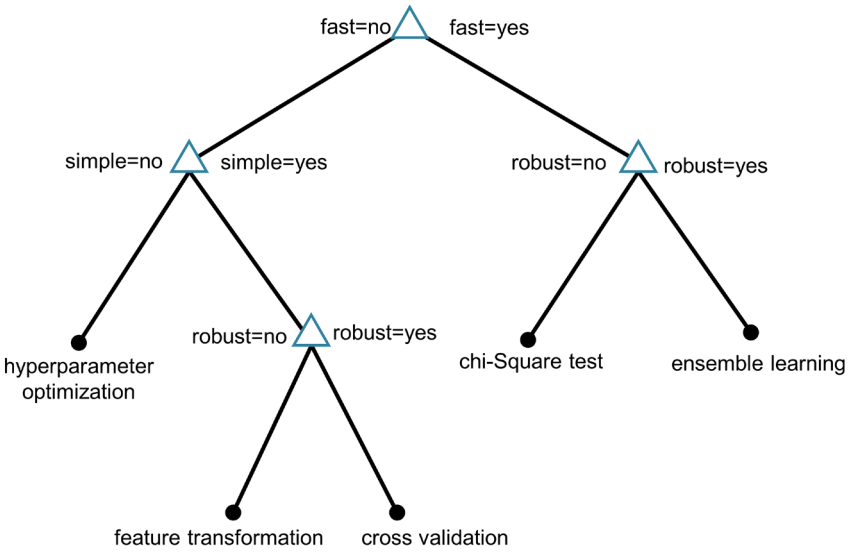
ARF Algorithm A3.11: Decision tree induction (DTI)

Input: D: Training set with sample data $s_i \in S$ on features $x_i \in X$
l_m: class label L
Output: $d\mathcal{T}$: Decision tree classifier

```
dT ← {}
if l_m ← ∀ s_i ∈ S do
    dT ←{Node(leaf) == l_m}
end if
if X = {∅} do
    l_m ← max (N(s_i))
    dT ←{Node(leaf) == l_m}
end if
D̂ ← D
n ← number of features of training set D
for i = 1 to n do
    S ← D̂(i)
    if s_1 ← (S − s_0) ∈ f_z do                    % Sample data S with a binary split
        p_o ← n(s_0)/N(S)
        p_1 ← (1 − p_0)
        I_g(i) ← GNI(p_0, p_1)                     % Calculate Gini impurity of sample data S
        wI_g(i) ← I_g(i) * ∑n(s_0→1)/N(S)          % Calculate Gini impurity of sample data S
    end if
    W← {wI_g(1), wI_g(2),…, wI_g(n)}
    W_idx ← min (W)
    Node (i) ← x_idx                               % Best split feature x_idx at Node (i)
    dT ← dT + {Node (i)}                           % Attach Node(i) to corresponding branch of Dtree
    D̂ ← (D̂ − D(idx))                              % Remove sample data D(idx) from D̂ based on feature x_idx
end for
Return dT
```

The decision tree model (DTM) supports the selection of the computational methods for testing the performances of the predictive model. Figure 4.15 shows an example of the simplified decision tree model and the decision rules, which were extracted from the model. The decision variables include (i) the prediction time, (ii) simplicity, and (iii) robustness. The five candidate methods are: (i) hyper-parameter optimization, (ii) feature transformation, (iii) cross validation, (iv) Chi-square test, and (v) ensemble learning. To avoid any fundamental mistakes, it should be mentioned that the training set was arbitrarily generated for the demonstrative purpose.

Component 3.42 is the RPP builder. The component constructs a matrix to represent a reference process protocol. The matrix contains a set of process flow models and a set of



a: Example of a decision tree model

```
1  if fast=no then node 2 elseif fast=yes then node 3 else Hyperparamter optimization
2  if Simple=no then node 4 elseif Simple=yes then node 5 else Hyperparamter optimization
3  if Robust=no then node 6 elseif Robust=yes then node 7 else Chi-Squared test
4  class = Hyperparamter optimization
5  if Robust=no then node 8 elseif Robust=yes then node 9 else cross validation
6  class = Chi-Squared test
7  class = Ensemble learning
8  class = Feature transformation
9  class = cross validation
```

b: Rule set extracted from the model

Figure 4.15: An example of the simplified decision tree model and the decision rules

computational methods. Each PFM represents a segment of the design process. It is created by a set of design entities and their relationships that are determined in the timed action model. A method for the execution of a design entity is selected by using the design tree model. The scope of the component is formally represented as follows:

$$[RPP] = compose\_RPP\ (DTM,\ TAM,\ ePFM) \tag{4-46}$$

where: [RPP] is a matrix representation of a reference process protocol, which contains a finite, non-zero set of process flow models. The dimensions of the matrix are $m \times n$, where m is the total number of the process flow models, and n is the total number of the design entities.

Each process flow model $\mathbb{P}_i$ is represented as a vector, which contains Boolean parameters

$v_{i,j} \leftarrow \{0,1\}$. Each element of the vector corresponds to one of the design entities in the model. The Boolean parameter is set to value '*1*' if the entity is contained in the design process or a part thereof, and '*0*' otherwise. With these, the design entities are composed into the process flow model $\mathbb{P}_{idx}$. Some of the design entities may require decisional support in order to select the best method. The decision tree classifier couples the entities with multiple potentials methods. The computational notations of a reference protocol and its elements are formally expressed as follows:

$$\mathbb{R} = [\mathbb{P} * \vec{\mathcal{E}}] \odot d\mathcal{T} = \left(\begin{bmatrix} v_{1,1} & \cdots & v_{1,n} \\ \vdots & \ddots & \vdots \\ v_{m,1} & \cdots & v_{m,n} \end{bmatrix} * \begin{bmatrix} e_1^p \\ \vdots \\ e_n^p \end{bmatrix} \right) \odot \begin{bmatrix} dt_1 \\ \vdots \\ dt_n \end{bmatrix} \tag{4-47}$$

$$\vec{\mathbb{P}}_{idx} \coloneqq \left\{ e^p_{(idx,i)}, e^p_{(idx,j)}, e^p_{(idx,k)}, \dots, e^p_{(idx,n)} \right\}, v_{idx,n} \neq 0 \tag{4-48}$$

$$dT \coloneqq \{dt_i, dt_j, dt_k, \dots, dt_n\}^T \tag{4-49}$$

where: $\mathbb{R}$ is a reference protocol, $\mathbb{P}_{idx}$ is a process flow model $idx \in m$, and dT is a set of the decision tree for a selection of the most appropriate method for a certain design entity of $\mathbb{P}_{idx}$. Two algorithms were implemented in this component: (i) algorithm A3.12 – compose an RPP; and (ii) algorithm A3.13 – graph construction to represent an RPP.

The content of the RPP is visualized as a graphical representation of a Bayesian network with decision points. The network is constructed based on the conditional independence relations of the design entities included in the segments of the design process. The set of concerned design entities $e_i \in \mathcal{E}$ is represented as nodes. The relations among the entities are represented in the network as directed arcs $l_i \in L$.

Their relationships are quantified by the frequency of co-occurrences of subsequent entities, $e_i, e_j \in \mathcal{E}$. For every decision point, the decision process is done by using the decision tree model. The ARF Algorithm A3.13 is used to construct the graph representing the contents of the RPP. An example of a graphical visualization of a reference protocol is shown in Figure 4.16.

4.5.1 Architecting and implementation of the reference protocol-based procedural obstacle identifier module

The reference protocol-based procedural obstacle identifier (ROI) module is the key contributor to the demonstrative implementation of the ARF. It comprises three procedural sub-modules (4.10, 4.20 and 4.30) as shown in Figure 4.17. The implementation of this module aims at demonstrating the computational processes of the utilization of the reference process protocol. The output is a proposal, which is a basis of the process-based recommendation.

The sub-module 4.10 investigates the RPP in a backward manner in order to find a possible

ARF Algorithm A3.12: compose an RPP

Input:	*TAM:*	Timed action model (TAM)
	$\mathcal{E}$:	design entities stored in the knowledge repository $e_i \in \mathcal{E}$
	$d\mathcal{T}$:	decision tree models stored in the knowledge repository
Output:	$\mathbb{R}$:	Reference process protocol

```
1:  [r, c] ← total number of rows and columns in TAM
2:  t_{i,j} ← TAM(i,j)        %Denoted t_{i,j} ← {0,1} is an external relation of the entities e_i
                                                                  and e_j in TAM
3:  idx ← 1
4:  ℙ̂_{idx} ← {∅}
5:  dT_{idx} ← {∅}
6:  for i = 1: r do
7:      for j = 1: c do
8:          if t_{i,j} = 1 && i < j
9:              ℙ̂_{idx} ← ℙ̂_{idx} + e_i          % Include a design entity e_i in PFM(idx)
10:             dt_i ← dt(e_i)                    % Find the decision tree model for entity e_i
11:                 if dt_i ≅ ∅ do
12:                     dT_{idx} ← dT_{idx} + dt_i
13:                 else do
14:                     dT_{idx} ← dT_{idx}
15:                 end if
16:             i ← j
17:         else do
18:             ℙ̂_{idx} ← ℙ̂_{idx}
19:         end if
20:     end for
21:     idx ← idx + 1
22: end for
23: ℝ ← [ℙ̂_{idx}] ⊙ [dT_{idx}]
24: Return ℝ
```

obstacle in the design process. The sub-module 4.20 utilizes the hybrid inference approach to select the best method to execute the actual design activity flow and predict the next design action. The sub-module 4.30 consolidates the elements of the proposed design activity flow and their contents to generate a proposal. The detailed descriptions of five algorithms are given below, including algorithms A4.10, A4.03, A.4.04, A.4.06, and A4.08.

4.5.1.1 Sub-module 4.10: Context-sensitive design process identifier

The component 4.11 identifies the candidate PFMs that can be representatives of the design process in context. Once the current design entity was identified by the DOI module, the component 4.11 component finds the preceding design entities in the reference protocol, so as:

ARF Algorithm A3.13: graph construction to represent an RPP

Input:	*TAM:*	Timed action model
	$f_{(i,j)}$:	frequency of co-occurrence of entities $e_i, e_j \in \mathcal{E}$
	$\mathcal{E}$:	design entities stored in the knowledge repository
	$d\mathcal{T}$:	decision tree models stored in the knowledge repository
Output:	$G_{\mathbb{R}}$:	graph representing an RPP

```
[r, c] ← total number of rows and columns in TAM
for all [i, j] = 1: [r, c] do
    if (TAM(i,j) == 1) do
        FQM(i,j) ← f(i,j)                    % Construct a frequency of co-occurrence matrix
    end if
end for
E ← incidence matrix (FQM)                   % Convert FQM into an incidence matrix
preN ← E (1)                                 % The first element of the incidence matrix is a set of preceding nodes
postN ← E (2)                                % The second element of the incidence matrix is a set of successive nodes
w ← E (3)                                    % The third element of the incidence matrix is frequencies of co-occurrence of entities of two subsequent nodes
p(e_i|e_j) ← CondiProb(preN, postN, w)       % Calculate conditional probabilities between two subsequent nodes
g ← digraph (preN, postN, p(e_i|e_j))        % Create a directed graph by connecting two subsequent nodes and labelled their edges with p(e_i|e_j))
n ← total number of nodes of graph g
for k = 1: n do
    l ← numedges (g, node (k))               % Count the number of the out-coming edges at node (k)
        if (l > 1) do
            e_k ← identical (E, node (k))    % Find e_k which represents at node k
            dT ← dt(e_k)                     % Retrieve a decision tree model for e_k
        end if
end for
dT ← { dt(e_k), dt (e_{k+1}),..., dt (e_n)}
G_R ← add(g, dT)                             % Add dt(e_k) to corresponded node k
Return G_R
```

```
[PFM_ctx] = idn_PFM(RPP, entD_ctx)
```
(4-50)

where: `[PFM_ctx]` is the process flow model representing the design process in context, $\mathbb{P}_{ctx} \in \mathbb{R}$. It is assumed that the entity in context e_{ctx}, is the last design action in the segment

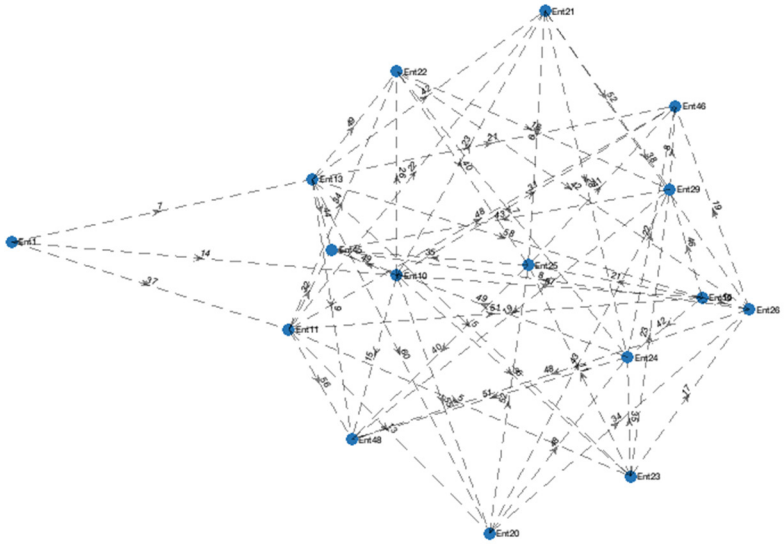


Figure 4.16: Visualization of graph representing an RPP (number of nodes = 16, number of edges = 59)

of the conducted design process. The PFM is segmented with the n number of entities. This can be expressed formally as follows:

$$\mathbb{P}_{ctx} := \{e_{ctx\text{-}(n\text{-}1)}, e_{ctx\text{-}(n\text{-}2)}, e_{ctx\text{-}(n\text{-}3)} \ldots, e_{ctx}\} \tag{4-51}$$

where: $\mathbb{P}_{ctx}$ is the PFM representing the segment of the design process in context. To investigate the states of the design process, it is assumed that an obstacle occurred in the preceding actions. The investigation is performed in multiple iterations. For example, it is supposed that there are three subsequent entities included in the PFM, $\mathbb{P}_{ctx} := \{e_{o,} e_{pre,} e_{post}\}$.

The entity in context, e_{ctx}, is the last element of the model. The preceding entity e_{pre} is in the middle. The entity e_o is the first one. These elements are composed to represent the segment of the design process in context, where $n = 3$. The Algorithm A4.01 was implemented to execute this component.

Component 4.12 selects the best representative process flow models. This component relies on probabilistic inference.The fundamental concept of the implementation was explained in sub-section 4.3.3.1 The computational process includes (i) identification of considered design entities, (ii) composition of segment of the process flow model, (iii) calculation of the joint probability distribution of the segments, and (iv) selection of the best segment, which represents the process flow model in context. The candidate segments with the highest joint probability are selected as the best ones. Symbolically:

```
[PFM_B] = select_BP(PFM_ctx)
```
(4-52)

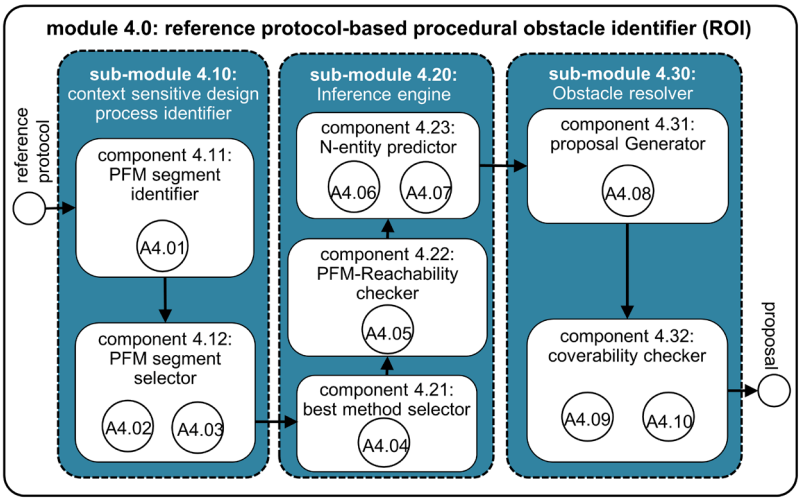


Figure 4.17: Interrelation of computational components of ROI module

where: `[PFM_B]`is the best representative segment of process flow model in context. Included in this component, the Algorithm A4.03 was implemented to select the best representative process flow model in context.

Table 4.8: List of variables used in the computational components of the ROI module

variables	description
dT	decision tree model
$\mathbb{P}_i$	segment of process flow model
$e^p{}_{post}$	successive design entity of the entity in context
$e^p{}_{ctx}$	design entity in context
$e^p{}_{pre}$	preceding design entity of the entity in context
$\widehat{\mathcal{M}}$	set of the best methods
$p(\hat{p}_k)$	joint probability distribution of a segment of design process
$\mathrm{c}\mathbb{P}_{ctx}$	candidtae process flow model in context
$\mathbb{P}_{inf}$	informative process flow model
$\mathbb{P}_{ext}$	extended process flow model
$\mathbb{R}$	proposal
CT_i	total sum of vector elements of the input/output sates of $\mathbb{R}$

ARF Algorithm A4.01: find candidate PFMs in context

Input: e_{ctx}: design entity in context
$\mathfrak{E}$: Repository of design entities
$\mathbb{R}$: reference process protocol
Output: $c\mathbb{P}_{ctx}$: candidate PFMs in context (length = 3)

1: $m \leftarrow$ total number of PFMs in RPP
2: $c\mathbb{P}_{ctx} \leftarrow \{\}$
3: **for** $i = 1$ to m **do**
4: $e_z \leftarrow$ find $(\mathfrak{E}, e_{ctx})$
5: $\mathbb{P}_{idx} \leftarrow$ find $(\mathbb{R}_i, e_z)$ % Find e_{ctx} which is an element of $\mathbb{P}$ on $\mathbb{R}$
6: $c\mathbb{P}_{ctx} \leftarrow c\mathbb{P}_{ctx} + \mathbb{P}_{idx}$
7: **end for**
8: $z_{ctx} \leftarrow$ total number of $c\mathbb{P}_{ctx}$
9: **if** $(z_{ctx} < 1)$ **do**
10: $c\mathbb{P}_{ctx} \leftarrow \{\}$
11: **else do**
12: $e_y \leftarrow$ preceding $(\mathbb{P}_{idx}, e_z)$
13: $z_{pre} \leftarrow$ total number of e_y
14: **if** $(z_{pre} < 1)$ **do**
15: $c\mathbb{P}_{ctx} \leftarrow \{e_z\}$
16: **else do**
17: **for** $j = 1$ to z_{pre} **do**
18: $e_o \leftarrow$ preceding $(\mathbb{P}_{idx}, e_y(j))$
19: $z_o \leftarrow$ total number of e_o
20: **if** $(z_o < 1)$ **do**
21: $c\mathbb{P}_{ctx} \leftarrow \{\forall e_y, e_z\}$
22: **else do**
23: $c\mathbb{P}_{ctx} \leftarrow \{\forall e_o, e_y(j)), e_z\}$
24: **end if**
25: $c\mathbb{P}_{ctx} \leftarrow \{\forall e_o, \forall e_y, e_z\}$
26: **end for**
27: **end if**
28: **end if**
29: **Return** $c\mathbb{P}_{ctx}$

4.5.1.2 Sub-module 4.20 – Inference engine

This sub-module includes three interrelated components (4.21- 4.23). The component 4.21 was designed to explore the possible PFM $\mathbb{P}_{ctx}$ options according to the context and to infer the most informative one $\mathbb{P}_{inf.}$ Using this result, the component 4.22 checks the reachability of $\mathbb{P}_{inf.}$ Furthermore, the component investigates the proposed design entity in context e_{ctx}, to confirm that all required input data are available.

If the reachability is proven, then it confirms that the configuration of $\mathbb{P}_{inf}$ is completed. Otherwise, it will identify the missing entity and add it into the process flow model. The reachability condition of the $\mathbb{P}_{inf}$ is rechecked. The process is done iteratively until the reachability condition of the $\mathbb{P}_{inf}$ is fulfilled. Then, the component 4.23 predicts the next design action, which best matches to $\mathbb{P}_{inf}$.

Component 4.21 selects the best methods for the elements of the most informative PFM by using the decision tree model. The input data consists of (i) the best segment PFM determined by the component 4.12, (ii) the decision tree models, and (iii) the captured patterns of the designer's responses, which relate to the design entities belonging to the considered PFM. The computational operation of the component is symbolically represented as follows:

```
[iPFM] = hybrid_inf(PFM_ctx, DTM, dD)          (4-53)
```

where: `[iPFM]` is the representation of the most informative process flow model which includes the design entities with the best method. The computational process is executed by

ARF Algorithm A4.03: selection of the best representative PFMs in context

Input:	$c\mathbb{P}_{ctx}$:	candidate PFMs in context
	$G_{\mathbb{R}}$:	graph representing the reference process protocol
Output:	$\mathbb{P}_{ctx}$:	The best representative PFM in context (length = 3)

1: $k \leftarrow$ total number of $c\mathbb{P}_{ctx}$
2: **if** $k < 1$ **do**
3: $\mathbb{P}_{ctx} \leftarrow \{\}$
4: **else do**
5: $\{e_o, e_{pre}, e_{ctx}\} \leftarrow c\mathbb{P}_{ctx}$
6: $z \leftarrow$ total number of nodes of $G_{\mathbb{R}}$
7: Node $(e_{ctx}) \leftarrow \{\}$
8: **for** $\forall n_i \in G_{\mathbb{R}}$ **do**
9: $n_i \leftarrow$ find $(G_{\mathbb{R}}, e_{ctx})$ % Find a node representing e_{ctx} on $G_{\mathbb{R}}$
10: **if** $(n_{idx} ==$ true) **do**
11: Node $(e_{ctx}) \leftarrow$ Node $(e_{ctx}) + n_i$
12: **end if**
13: **end for**
14: $\{n_i, n_j, n_k, \ldots, n_n\} \leftarrow$ Node (e_{ctx})
15: $z_{ctx} \leftarrow$ total number of Node (e_{ctx})
16: **if** $(z_{ctx} > 1)$ **do**
17: $T_{idx} \leftarrow \min(T\{n_i, n_j, n_k, \ldots, n_n\})$ % Select a node with the smallest number of processing duration time T
18: Node $(e_{ctx}) \leftarrow n_{idx}$
20: $z_{pre} \leftarrow$ total number of Node (e_{pre})

ARF Algorithm A4.03 is continued

ARF Algorithm A4.03 continuation

```
21:     if (z_pre > 1) do
22:         for j = 1 to z_pre do
23:             n_j ← Node (e_pre)
25:             z_0 ← total number of Node (e_0)
26:                 if (z_0 > 1) do
27:                     p_0(j) ← sum (p({n_0, n_1, n_2, ..., n_z0}))
28:                 else do
29:                     p_0(j) ← p(n_0)
30:                 end if
31:             p_ctx ← p(n_ctx| n_j) * p_0(j)      % Calculate JDP of a node representing
                                                    candidate P_ctx
32:             p_idy ← max (p_1, p_2, ... p_j)
33:         end for
34:         Node (e_pre) ← n_idy
36:         z_0 ← total number of Node (e_0)
37:         if (z_0 > 1) do
38:             p_idz ← max ({p(n_0), p(n_1), p(n_2), ..., p(n_z0)})
39:             Node (e_0) ← n_idz
40:             P_ctx ← {n_idz, n_idy, n_idx}
41:         else do
42:             p_0 ← p(n_0)
43:             n_0 ←Node (e_0)
44:             P_ctx ← {n_0, n_idy, n_idx}
45:         end if
46:     else do
47:         Node (e_pre) ← n_idy
49:         z_0 ← total number of Node (e_0)
50:         if (z_0 > 1) do
51:             P_ctx ← {n_idz, n_pre, n_idx}
52:         else do
53:             P_ctx ← {n_0, n_pre, n_idx}
54:         end if
55:     end if
56: else do
57:     P_ctx ← {n_0, n_pre, n_ctx}
58: end if
59: end if
60: Return P_ctx
```

using Algorithm A4.04 – '*select the best method for the PFM in context*'

The component 4.22 checks the reachability of the process flow model. Here, the term 'reachability' simply refers to the possibility of getting from one node to another node within a graph.

As explained related to the sub-module 3.20, there are three primary types of net configurations of a design entity. Each type consists of varied number of input and output states. When $\mathbb{P}_{inf}$ is identified, this component checks if the reachability of the design entity $\hat{e}_{ctx}$ is achieved when all output-input states of the subsequent elements of $\hat{e}_{ctx}$ are composed perfectly. The output is the vector of initial markings m_i, which contains Boolean parameters.

The length of the vector is the total number of input parameters of the target entity in the instance. A vector element represents the similarity of the output-input parameters of two subsequent entities $m_i \leftarrow\{0,1\}$. It returns (*1*), if they are identical and (*0*), otherwise. The condition defines that the reachability of a process flow model is achieved should all vector elements of m_i be equal to *1*.

$$m_i = \begin{cases} 1, \ if \ \sum_{i=1}^{n} s_{out}^{pre} \equiv s_{in} \ ; \\ 0, \ otherwise \end{cases} \tag{4-54}$$

ARF Algorithm A4.04: select the best method for the PFM in context

Input:	$\mathbb{P}_{ctx}$:	PFM in context	
	$\mathcal{E}$:	Repository of design entities	
	$\mathbb{R}$:	Reference protocol	
	dD:	Captured pattern of designer's responses	
Output:	$\mathbb{P}_{inf}$:	informative PFM (length <= 3)	
1:	$l \leftarrow$ total number of elements of $\mathbb{P}_{ctx}$		
2:	**if** $l < 3$ **do**		
3:	$\mathbb{P}_{inf} \leftarrow \mathbb{P}_{ctx}$		
4:	**else do**		
5:	$i = l - 2$		
6:	$e_0 \leftarrow \mathbb{P}_{ctx}(i)$		
7:	$e_{idz} \leftarrow$ find $(\mathcal{E}, e_0)$		% Find a design entity e_0
8:	$[tf] \leftarrow$ find (dD, e_{idz})		% Find a captured pattern of designer's responses which relates to design entity e_{idz}
9:	**if** ($dD(e_{idz})$== true) **do**		
10:	$s_{idz} \leftarrow$ d$D(e_{idz})$		
11:	$dt_{idz} \leftarrow$ retrieve $(\mathbb{R}, e_{idz})$		% Retrieve the decision tree model which relates to design entity e_{idz}
12:	$\hat{m}_0 \leftarrow$ predict (dt_{idz}, s_{idz})		% Predict the proper method according to d$D(e_{idz})$
13:	**else do**		
14:	$\hat{m}_0 \leftarrow \mathcal{M}(e_{idz})$		
15:	**end if**		

ARF Algorithm A4.04 is continued

ARF Algorithm A4.04 continuation

16: $i == i+1$
17: $e_{pre} \leftarrow \mathbb{P}_{ctx}(i)$
18: $e_{idy} \leftarrow$ find $(\mathcal{E}, e_{pre})$ % Find a design entity e_{pre}
19: $[tf] \leftarrow$ find $(\mathrm{d}D, e_{idy})$
20: **if** $(dD(e_{idy}) ==$ true) **do**
22: $s_{idy} \leftarrow \mathrm{d}D(e_{idy})$
23: **else do**
24: $s_{idy} \leftarrow$ orginzeDial(Qs(e_{idy}), *DesignerSay()*)
25: **end if**
26: $d\mathcal{t}_{idy} \leftarrow$ retrieve $(\mathbb{R}, e_{idy})$
27: $\hat{m}_{pre} \leftarrow$ predict $(d\mathcal{t}_{idy}, s_{idy})$ % Predict the proper method according to d$D(e_{idy})$ by A4.04
28: $e_{ctx} \leftarrow \mathbb{P}_{ctx}(i+1)$
29: $e_{idx} \leftarrow$ find $(\mathcal{E}, e_{ctx})$ % Find a design entity e_{ctx}
30: $s_{idx} \leftarrow$ find $(\mathrm{d}D, e_{idx})$
31: $S \leftarrow$ compose (s_{idy}, s_{idx}) % Combine two sets of patterns of designer's responses which relates to design entity e_{idy}&e_{idx}
32: $d\mathcal{t}_{(idx\&idy)} \leftarrow$ retrieve $(\mathbb{R}, [e_{idy}, e_{idx}])$ % Retrieve the composed decision tree model which relates to design entity e_{idy}&e_{idx}
33: $\hat{m}_{ctx} \leftarrow$ predict $(d\mathcal{t}_{(idx\&idy)}, S)$
34: $\hat{e}_0 \leftarrow (e_{idz}, \hat{m}_0)$
35: $\hat{e}_{pre} \leftarrow (e_{idy}, \hat{m}_{pre})$
36: $\hat{e}_{ctx} \leftarrow (e_{idx}, \hat{m}_{ctx})$
37: $\mathbb{P}_{inf} \leftarrow \{\hat{e}_0, \hat{e}_{pre}, \hat{e}_{ctx}\}$
38: **end if**
39: **Return** $\mathbb{P}_{inf}$

where: m_i is a vector element of `[M0]`, $s^{pre}{}_{out}$ is an output state of a preceding design entity, and s_{in} is an input state of the target entity.

If the reachable condition fails, it means that at least one input parameter of the target entity is missing. The procedure for checking the reachability is iterated in the next run by searching for another preceding design entity having the highest number of co-occurrence frequencies with the target entity, to fulfil the missing marking. The investigation of state-transition of the design activity flow is done at the lower level of the PFM. In the demonstrative implementation, we assumed that the reachable conditions of the PFMs are satisfied for all trials.

The component 4.23 creates a recommendation by predicting the next design entity concerning $\mathbb{P}_{inf}$. The component calculates the joint probability distribution of the extended PFM $\mathbb{P}_{ext}$, which includes two elements of $\hat{e}_{pre}$, $\hat{e}_{ctx} \in \mathbb{P}_{inf}$ and the successive design entity

ARF Algorithm A4.06: predict the next design action

Input:	$\mathbb{P}_{inf}$:	informative PFM
	$G_{\mathbb{R}}$:	Graph representing the reference process protocol
Output:	e_{post}:	Next design entity
Procedure:		
1:	$\{\hat{e}_0, \hat{e}_{pre}, \hat{e}_{ctx}\} \leftarrow \mathbb{P}_{inf}$	
2:	Node $(\hat{e}_{ctx}) \leftarrow$ find $(G_{\mathbb{R}}, \hat{e}_{ctx})$	%Find a node representing $\hat{e}_{ctx}$ on $G_{\mathbb{R}}$
3:	$n_{ctx} \leftarrow$ Node $(\hat{e}_{ctx})$	
4:	Node $(\hat{e}_{ctx} \leftarrow$ find $(G_{\mathbb{R}}, \hat{e}_{pre})$	%Find a node representing $\hat{e}_{ctx}$ on $G_{\mathbb{R}}$
5:	$n_{pre} \leftarrow$ Node $(\hat{e}_{pre})$	
6:	$n_{post} \leftarrow$ successive $(G_{\mathbb{R}}, n_{ctx})$	%Find successive node n_{post} on $G_{\mathbb{R}}$
7:	$k \leftarrow$ number of n_{post}	
8:	**if** $k == 0$ **do**	
9:	$n_t \leftarrow n_{ctx}$	% n_{ctx} is a terminate node
10:	$e_{post} \leftarrow \{\}$	
11:	**elseif** $k == 1$ **do**	
12:	$e_{post} \leftarrow$ identical $(\mathcal{E}, n_{post})$	
13:	**else do**	
14:	**for** $i = 1$ to k **do**	
15:	$p_i \leftarrow p(n_i \mid n_{ctx}) * p(n_{ctx} \mid n_{pre}) * p_0(n_{pre})$	% Calculate JDP of a node representing candidate $\mathbb{P}_{ext}$
16:	**end for**	
17:	$p_{idx} \leftarrow \max(p_1, p_2, ..., p_k)$	
18:	$n_{post} \leftarrow n_{idx}$	
19:	$e_{post} \leftarrow$ identical $(\mathcal{E}, n_{post})$	
20:	**end if**	
21:	**Return** e_{post}	

e_{post} ,. It selects the e_{post} that makesthe $\mathbb{P}_{ext}$, having the highest value of JPD. Symbolically:

```
[PFM_ext, iEntD] = infer_ST(iPFM, RPP)        (4-55)
```

where: [PFM_ext] is the extended process flow model, [iEntD] is the identified design entity that makes $\mathbb{P}_{ext}$ having highest value of JPD. The inference process is executed by using the AFR Algorithm A4.06 – '*predict the next design action*'. The detailed description of the AFR algorithm A4.06 is given below.

4.5.1.3 Sub-module 4.30 – Obstacle resolver

The sub-module comprises two interrelated components (4.31-4.32). The component 4.31 – *proposal generation*, implements algorithm A4.09 to wrap up the contents for a proposal as a process-based recommendation to resolve the procedural obstacle. The component 4.32 – *PFM coverability checker*, implements two algorithms (A4.10-A4.11) for building a

coverability tree of the proposal and using the tree to check the fulfillment of input/output states throughout the proposal.

Component 4.31 generates the proposal concerning the extended process flow model. At this point, all elements of the extended process flow model are identified. They comprise three subsequent design entities representing the proposed PFM. This component aims at consolidating the information related to (i) the entities included in t $\widehat{\mathbb{R}}$ net configuration of the extended process flow model, (ii) the specification of input-output parameters, and (iii) the most appropriate method for each entity. To create recommendation content, this component retrieves the key terms related to the contents of recommendation items, which are stored in its data model. The symbolic representation of the component is as follows:

```
[PS, eT] = Gen_PS(ePFM, iEntD)                    (4-56)
```

where: `[eT]` is a collection of key terms related to the design entities, and `[PS]` is a proposal to resolve the cause of the identified obstacle. The proposal R is formulated by using the reference protocol. Symbolically:

$$\widehat{\mathbb{R}} = \mathbb{P}_{ext} * \widehat{\mathcal{M}} := \{(\hat{e}_{pre}), (\hat{e}_{ctx}), (e_{post})\} * \begin{bmatrix} \hat{m}_i \\ \hat{m}_j \\ m_k \end{bmatrix} \qquad (4\text{-}57)$$

where: $\widehat{\mathbb{R}}$ is a proposal, $\mathbb{P}_{ext}$ is the extended process flow model, $\hat{m}_i$, $\hat{m}_j$ is a set of the best approach corresponding to the given entities e_{pre} and e_{ctx}, and m_k is a method, which is the most frequently used for execution of entities e_{post}. The proposal is generated by using ARF Algorithm A4.08 – '*proposal generation*.

The component 4.32 checks the fulfilment of the proposal. This component builds the net configuration representing the state-transition models of $\mathbb{R}$ and analyzes the coverability of the net. It checks if (i) all required data are available, (ii) all connections in the state-transition models of $\mathbb{R}$ are fulfilled, and (iii) the expected output was correctly produced. The output is the coverability of the extended process flow model. The algorithm A4.10 – '*coverability checker*' returns a logical value [*true*] when the coverability condition is satisfied, otherwise it returns [*false*]. Should the latter be the case, it indicates what required data is missing or where the output-input connection between the subsequent design entities is broken. This information is a supplement to the proposal. The coverability of the PFM is formally expressed as follows:

ARF Algorithm A4.08: Proposal generation

Input:	$\mathbb{P}_{ext}$:	Extended segment PFM
Output:	$\widehat{\mathbb{R}}$:	proposal
1:	$\{\hat{e}^p_{pre}, \hat{e}^p_{ctx}, e^p_{post}\} \leftarrow \mathbb{P}_{ext}$	
2:	$\hat{m}_{pre} \leftarrow \mathcal{M}(\hat{e}^p_{pre})$	
3:	$\hat{m}_{ctx} \leftarrow \mathcal{M}(\hat{e}^p_{ctx})$	
4:	$m_{post} \leftarrow \mathcal{M}(e^p_{post})$	
5:	$\widehat{\mathbb{R}} \leftarrow [(\hat{e}^p_{pre}, \hat{m}_{post}), (\hat{e}^p_{ctx}, \hat{m}_{ctx}), (e^p_{post}, m_{post})]$	
6:	**Return** $\widehat{\mathbb{R}}$	

$$CT_i = \begin{cases} \sum_{i=1}^{t} m_i\,, & if\ \sum_{i=1}^{t}\sum_{i=1}^{n} s_{out} \equiv \sum_{i=1}^{t}\sum_{i=1}^{m} s_{in}\ ; \\ 0, & otherwise \end{cases} \tag{4-58}$$

where: CT_i is total sum of vector elements of the input/output states of $\mathbb{R}$, s_{in} is an input state, and s_{out} is an output state of the entities of $\mathbb{R}$, n is the total number of data elements for the output states, m is the total number of data elements for the input states, and t is the total number of transitions $t = N\text{-}1$. The variable N stands for the total number of design entities of $\mathbb{R}$.

As mentioned in Sub-section 4.3.3.1, the investigation of the design process was performed on two levels. One level is the design action level, and another is the state and transition of design activity flow level). The coverability of the net configurations of the proposed PFM is checked. It is performed iteratively until the convertibility of the PFM is achieved. The operational goal of the component 4.32 is confirming that all required data are available for all elements of the PFM. With regard to the demonstrative implementation, we assumed

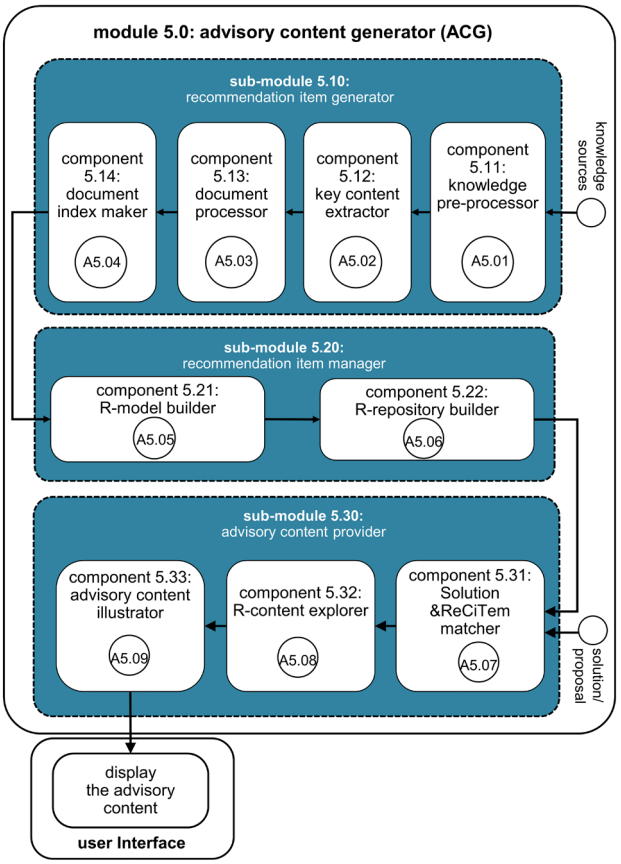


Figure 4.18: Interrelation of the computational components of the ACG module

that the coverability condition of PFM is fulfilled for all proposals.

4.5.2 Architecting and implementation the advisory content generator module of the ARF

The advisory content generation module consists of three architectural sub-modules with nine interrelated computational components as shown in Figure 4.18. The sub-module 5.30 is considered to be crucial contribution to the implementation of this module. It finds the best recommendation item by computing the similarity of terms by identifying the elements in the proposal and the terms indexing the recommendation item. The item is a knowledge model that contains (i) the knowledge source, (ii) the main contents, and (iii) its key terms. Should the best recommendation item be found, it navigates the designer to the knowledge source. The advisory content generation aims at providing descriptive information related to the proposed process-based recommendation. The contents support the designer to operate according to the recommendation. The other two modules (5.10 and 5.20) are responsible for the processes of modelling the recommendation item and building the knowledge repository.

4.5.2.1 Sub-module 5.10: Recommendation item generator

The recommendation item generator sub-module is designed (i) to extract the main contents from the knowledge source, (ii) to compile them into a recommendation item, and to (iii) index with the key terms. The implementation of the sub-module consists of four interrelated computational components (5.11 - 5.14). The implemented algorithms were modified based on the built-in functions included in the text analysis process toolbox of MATLAB.

Component 5.11 converts the knowledge sources into a text description. It aims at decoding contents included in the knowledge sources into a human-readable format. Each source contains descriptive contents related to a given design action. Knowledge sources can be in several formats, for instance, webpage, electronic corpus, and technical documents. An input data is a finite set of strings of raw texts $\tau_i \in T_\alpha$ decoded from knowledge sources. This component is executed by using the reusable functions including web-access (e.g., Webread) and content processing (e.g., findElement, extractHTMLText) provided by MATLAB. The output is a collection of raw texts extracted from the knowledge sources, $s_i \in S_\alpha$.

Table 4.9: List of variables used in the computational components of the ACG module

variables	description
S_α	knowledge source of recommendation contents
T_α	terms included in a recommendation content
D	document containing a recommendation content
$simv$	similarity value
$\mathcal{R}$	repository of recommendation items

The component 5.12 finds the main contents in raw texts. They most probably contain irrelevant information concerning the

description of a solution how to progress in the design process. The computational process selects the relevant contents by finding a query that contains each line of the texts. The input data are (i) raw texts, and (ii) a query (which is a set of search strings used to find the most relevant content). The output data is a finite set of strings of the main content $c_i \in C_\alpha$ with regard to a query. The component is represented by algorithm A5.02, which matches a query to raw texts. This algorithm has been realized by modifying existing code according to the built-in functions (e.g., split function, tokenizedDocuments function, and token-based similarity function).

The component 5.13 generates a document and extracts frequently used words. This component organizes the contents into a well-structured document and analyzes the frequency of the words used in the document. The input data is a set of the strings related the main content, so as $c_i \in C_\alpha$. The output data are (i) a document, which stores contents for understanding the word, and (ii) a collection of words frequently used in the document called a bag of words. Algorithm A5.03 is implemented based on the modification of the built-in functions (e.g., extractSummary, BagOfwords).

Component 5.14 extracts key terms indexing the document. It counts the frequencies of words appeared in the documents and selects the most frequently used terms to index the document. The output is a finite set of key terms indexing the document. Algorithm A5.04 – *extract key terms*, was modified based on the built-in function <topkwords>.

4.5.2.2 Sub-module 5.20: Recommendation items repository builder

The sub-module 5.20 comprises two interrelated components 5.21-5.22. They aimed at constructing the repository of recommendation items. First, component 5.21 builds a data model representing a recommendation item. The model is constructed with three fields of its profile contents included: (i) a knowledge source of recommendation item; (ii) a document containing the descriptive contents of a solution; (iii) key terms that represent the main content in the document. The output is a data model of a recommendation item. We used a built-in function <struct> to create the model.

Second, the component 5.22 constructs a repository of the recommendation items. The input data is a collection of knowledge sources. They are evaluated to ensure that each knowledge source corresponds to a particular solution. It is stored with a finite set of pairs of the knowledge source and the corresponding key terms. The output is a knowledge repository stored as a finite set of recommendation items.

4.5.2.3 Sub-module 5.30: Advisory content provider

The advisory content provider sub-module consists of three interrelated components (5.31-5.33). The ultimate output is a comprehensive process-based recommendation and its advisory contents.

The component 5.31 determines the similarity of the terms defining a solution and

ARF Algorithm A5.07: Calculate text similarity of *k*-terms and ReciTems

Input:	*SL:*	terms describing a solution
	doc:	a document containing descriptive solution
Output:	*simv:*	similarity measures of a document and a solution

```
1:  SL ← "strSolution"
2:  SLt ← Split the sentence into tokenized terms
3:  SLt-w ← Remove general terms from the sentence
4:  k ← size (SLt-w)              %Denoted k is number of terms in SLt-w
5:  l ← size (doc)                %Denoted l is number of lines in doc
6:  for i = 1 to l do
7:      for j = 1 to k do
8:          simv(i,j) ← textsim (line (i), St-w (j))
9:      end for
10: end for
11: Return simv
```

a recommendation item. The computational process calculates the similarity of the recommendation items and the string arrays defined a solution.

```
simValue_M = simKterms(query, RECiTem)        (4-59)
```

where: `[simValue_M]` is a matrix containing the similarity values of a set of recommendation items and the textual description of a solution, and `[Repo_RiTems]` is a repository storing a finite set of recommendation items. This component includes the algorithm A5.07 to measure the similarity of the terms defining the solution and the recommendation items.

The component 5.32 selects the most relevant recommendation content. According to the similarity of recommendation items and the term defining the identified solution, this component generates a short-list of the recommendation items ranked according to the similarity scores. The best solution is at the first rank which offers URL name to navigate the designer to webpage, which contains the most informative advisory content. Symbolically, it is specified as:

```
[BRiTem] = select_BiTem(simValue_M, ReciTems)      (4-60)
```

where: `[BRiTem]`represents the best recommendation item.

This component employs the algorithm A5.08 to find the top N ranked recommendation items and then creates a ranking of candidate items. It calculates the average value of the similarity between the term indexing recommendation items available in the knowledge repository and the textual description of the solution/proposal.

ARF Algorithm A5.08: Top N rank of recommendation items

Input:	*simv:*	similarity measures of a document & a solution/proposal
	$\mathcal{R}$:	Repository of recommendation items
Output:	$\mathcal{S}'_{\alpha}$:	knowledge source of the best recommendation item

```
1:  m ← size (R)          % m is number of recommendation items (RI) stored in the repository
2:  for i = 1 to m do
3:      [r, c] ← size (simv)
4:      for j = 1 to r do
5:          SimV(j) ← sum(simv(j), [1:c])
7:      end for
8:      SimScore (i) ← mean (SimV)
9:      RI ← retrieve RI(i) from R
10: end for
11: ShortList ← sort (SimScore>0, RI, 'descend')
12: RI_best ← RI (ShortList ==1)
13: S'_α ← retrieve (S_α, RI_best) from R
14: Return S'_α
```

The component 5.33 displays the recommendation. The process concludes which overall contents related to the best recommendation item, and the intended design entity. By using a form-based user interface, the component provides the necessary information structure, which, as mentioned earlier, includes the process-based recommendation and its related advisory contents. Should the elements of a process-based recommendation be generated, the description of the advisory contents is filled in the form.

```
[CR] = genCR(BRiTem,PS)                    (4-61)
```

where: `[CR]` represents the final form of the recommendation, and `[PS]` is the proposal.

4.6 Putting the demonstrative implementation into application context

4.6.1 On the necessity of testing the demonstrative implementation in application context

This section discussed the testing of the realized functionality of the demonstrative implementation in the application context of designing a reasoning mechanism for an automated parking assist system (APAS). The testing would show how the implemented

modules and components of the concerned mechanisms of the ARF work in the WPE session with the design task $\mathfrak{D}_{I.0}$. According to the reasoning process of APAS in section 3.9.2 in Chapter 3, the task was defined as a development of a machine learning-type algorithm A01 to predict the most appropriate parking case for an actual parking scenario. The ML-type algorithm is not only built into the reasoning mechanisms of the APAS, but it is also known for the ARF at the end of the design process.

To evaluate if the selection of the most applicable motion path for the car to be parked is correct, the testing should show that the adapted machine learning-type algorithm builds a reasoning model to properly select the motion path. The subject of immediate testing is the appropriateness of the machine-learnt model in the given application context. We applied the reasoning-with-consequences principle to reason about and to test the functionality of the implemented demonstrative part of the ARF. As the ultimate output, the chosen motion path fits (appropriate for) the actual parking scenario, that the ML-type algorithm works correctly. However, not only the proper working of the machine learning algorithm, but also its proper training/learning should be assumed. If these are shown to be correct, then the concerned algorithm fulfils the requirements related to the functionality of the demonstrative implementation.

Considering these, the testing process was decomposed to the following stages: (i) showing that the selected motion paths are correct (or which one is better); (ii) claiming that then the operation of the machine learning algorithm had to be correct; (iii) claiming that then the data and training of the machine learning algorithm had to be correct; (iv) claiming that then the support provided by the ARF for the design task had to be correct. It must be mentioned that this did not allow us to conclude about what might be and what might not be proven with regard to a full-scale support of the entire reasoning mechanism design process and the service packages of the whole of the ARF.

4.6.2 Introducing the concrete application context

The guiding assumption concerning the testing of the demonstrative implementation was that a self-driving car (level 4) car sought and found a free parking space on the roadside and the APAS was activated by the on-board computer for parking the car. The parking scenario is shown in Figure 4.19. The APAS collected the necessary information from a set of sensors, including (i) a 2D representation of the parking lot (its width and length), and (ii) the distances between the car and the other objects (black cars) at a given moment in time in the parking scene. The front-side ultrasonic sensor is used for the measurement of the size of the longitudinal parking spaces. Basically, it scans the potential parking space while passing it. The length of a parking space is computed using the following equation [17].

$$D = \frac{t_2 - t_0}{t_1 - t_0} D_s \tag{4-62}$$

where: D is the length of the empty parking space, D_s is the distance between the front sensor and the rear sensors, t_0, t_1 is the time that the front sensor detects the first and the second

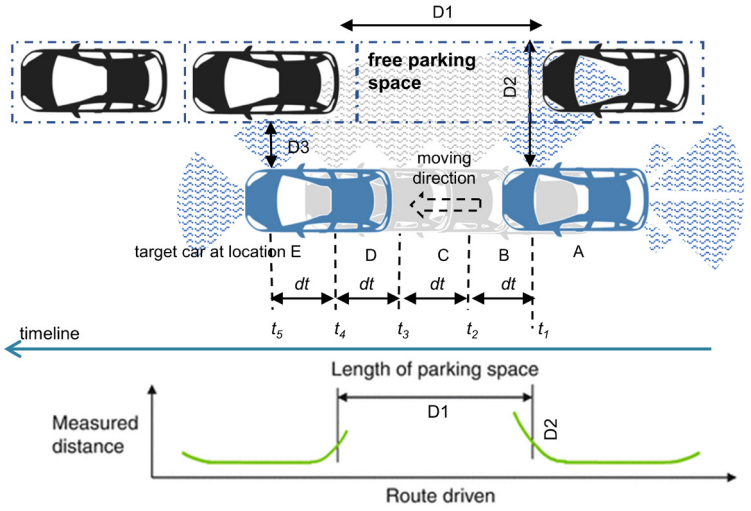


Figure 4.19: Spatial-temporal representation of a parking scenario (adapted from [17])

significant distance changes, respectively, and t_1 is the time that the rear sensor detects the first significant distance change. Whilst the car is moving toward the empty parking space, the situation reasoning mechanisms of the APAS evaluate the state of the parking scene and selects the proper space. Then, the working principle session commences and uses the developed ML-type algorithm to find the best match parking case and the actual parking situation (ASRM algorithm A_{01}).

The process of evaluation of the best parking cases includes multiple stages of reasoning, which in turn requires multiple interrelated algorithms, for instance, (i) an algorithm for extracting the morphological information structures from the candidate parking cases (ASRM algorithm A02), (ii) an algorithm for comparing the topological sub-structures of the motion paths of the past parking cases (ASRM algorithm A03), and (iii) an algorithm for adapting the optimally-matching motion path (ASRM algorithm A08). Should the best motion plan be selected, it will be converted into the action plan in the decision logical generation session (which is not addressed in this dissertation).

4.6.3 Overview of the testing of the demonstrative implementation in the concrete application context

In the simulation of parking scenarios, the evaluation of a parking situation differs from the real-life situation. As shown in Figure 4.20, the spatial information of a car is determined by a bounding box. The location of the car is identified in the Euclidean space by the x-y axis coordinates of the vertices (corners) of the bounding box. It is symbolically expressed as follows:

$$car = [x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4] \tag{4-63}$$

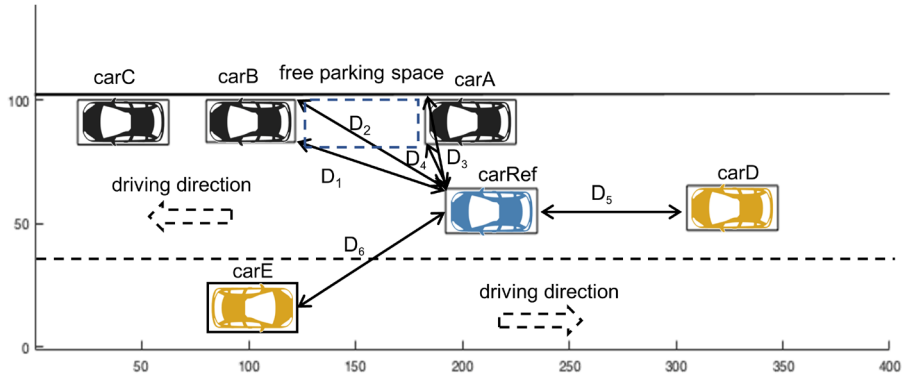


Figure 4.20: Simulation of a parking scenario

where: x_i, y_i is the x-y coordinates of the vertices (corners) of the bounding box.

Assumed is that the 2D model of the spatial situation was developed in the preceding design session, which centered on 'situation modelling'. The model was stored in the knowledge repository. The situation model contains the spatial-temporal information of the parking situation. A descriptor of a situation is used to characterize the situation, which captures a set of parameters identifying the locations of cars included in the parking situation at time t. Figure 4.21 shows the spatial information of the parking scenario presented in Figure 4.20. To categorize the similar situations, the similarity measures are calculated based on these parameters.

The parking situation is evaluated by the dimensions of parking space and the distance between car to be parked and the car following in the driving lane. Three possible parking scenarios were determined: (i) parallel parking, (ii) perpendicular parking, and (iii) not applicable for parking. After the evaluation of the parking scenarios, a dataset is generated from the instances of the seven groups of attributive information, including the distances, D_1-D_6, which correspond to the evaluated parking scenarios.

```
SampleCase =
  struct with fields:

      carA: [182.5450 225.1550 225.1550 182.5450 100 100 82.1100 82.1100]
      carB: [80.1271 121.8271 121.8271 80.1271 100 100 82.9000 82.9000]
      carC: [20 62.6100 62.6100 20 100 100 81.9100 81.9100]
      carD: [305.0813 347.6913 347.6913 305.0813 65.2954 65.2954 47.2054 47.2054]
      carE: [102.2287 145.0687 145.0687 102.2287 34.0802 34.0802 15.9902 15.9902]
    carRef: [192.2287 235.0687 235.0687 192.2287 64.0802 64.0802 45.9902 45.9902]
```

Figure 4.21: Spatial information representing a parking scenario at initial time t_0

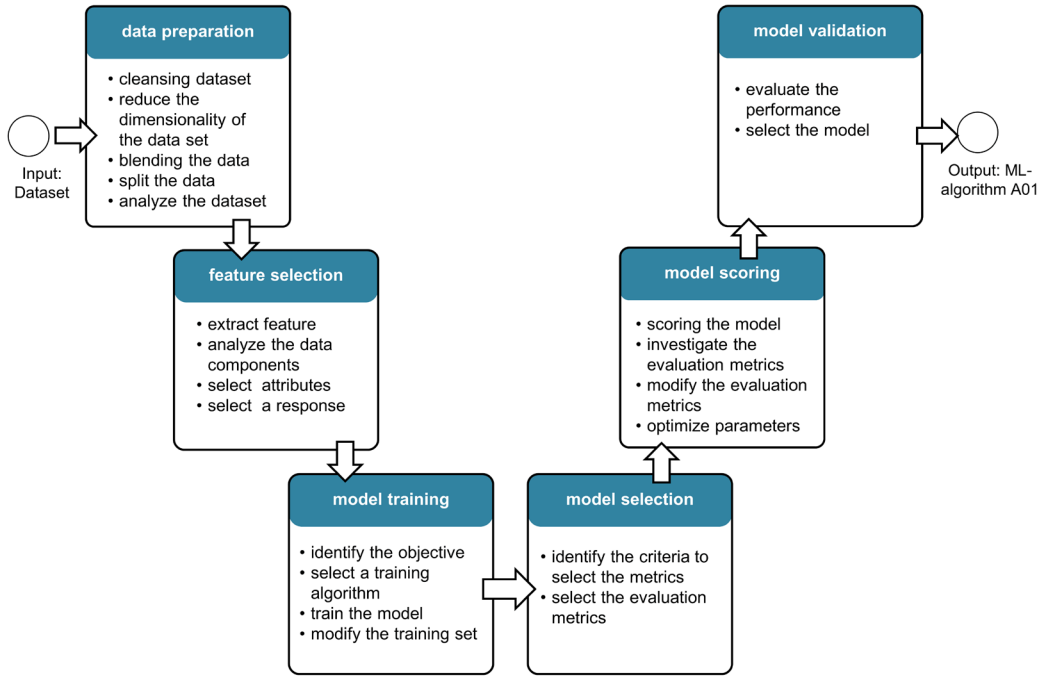


Figure 4.22: Design sub-tasks for a development of machine learning-type algorithm A_{01}

4.6.4 Development of machine learning algorithm for predicting the most appropriate parking case

In practice, a complicated design task is decomposed into multiple design sub-tasks. A design sub-task provides a set of design actions, which should be done in order to accomplish the concerned element of design process. For the demonstration case, Figure 4.22 shows an example of the design actions concerning the design task $\mathfrak{D}_{1.0}$. The goal of the design task is what parking position and direction the APAS will select to perform a proper parking with the minimum number of maneuvers.

According to the design sub-tasks and the example of design entities as shown in Figure 4.22, the design activity flow can be created in multiple ways through the network of design entities. The interrelationships of design actions associated with the computational methods allows the designer to develop the ML-type algorithm A01. However, various constrains concerning i.e., logical, theoretical, and practical aspects limit the design actions. These conditions influence the construction of the reference process protocol.

To train the program developed for selection of the most applicable motion path for the car to be parked, the dataset consists of the records of 887 parking cases characterized by six features, including the distances, D_1-D_6. But, in order to be able to test the recommendation related to the feature selection task, we added one more feature which is not relevant to the parking situation. Hence, the training data set contains seven features (F01-F07). The instances of the features correspond to the response variable, which is identified by three types of parking positions, where: (*0*) is not applicable for parking, (*1*) is parallel parking, and (2) is perpendicular parking as shown in Figure 4.23.

4.6.5 Supporting the development of ML-type algorithm A01 by the demonstrative implementation of the active recommender framework

4.6.5.1 Identification of knowledge content for the decision support

In the process of the recommendation using the RPP, the investigation of obstacle and the exploration of the proper design activity flow are done based on the probabilistic reasoning. To select the proper usable method, the decision tree model is applied. Referring to the work of Rathore and Kumar (2017), they identified ten characteristics of a faulty dataset, which have the largest influence on the performance of the learning algorithm [18]. These aspects can be applied to analyze the characteristic of the data set for training the decision tree model in particular for the first design sub-tasks (i.e., the data preparation, the feature selection, and the model training). Hence, we applied these aspects as the prediction variables for the selection of the most appropriate method concerning an intended design entity related to the first three design sub-tasks

They are: (i) noise (that reflects the lack of information or unreliable information), (ii) high dimensionality of input data (that means having too many features in the training dataset), (iii) heterogeneity of the data (that means having different natures of the features e.g., discrete, discrete ordered, or continues values in the training dataset), (iv) redundancy in

SampleSet =
887×8 tall table

F01	F02	F03	F04	F05	F06	F07	P
0.19474	4.0744	1.7513	4.5859	2.835	173.68	1.5601	2
0.066945	3.5029	0.68191	5.7026	8.6811	122.16	-86.126	1
95	92.73	57.824	54.063	142.78	137.18	-26.49	2
0.22625	2.3116	1.2423	5.8483	3.2115	133.93	-24.76	1
28.791	37.006	17.038	27.953	223.01	194.12	67.747	1
0.16491	2.5401	1.0618	3.567	3.037	165.53	-13.805	1
148.39	94.549	45.1	241.47	469.87	157.94	-39.623	0
0.017571	3.463	1.3346	3.9044	4.2699	201.64	-9.6275	1
:	:	:	:	:	:	:	:

Figure 4.23: Dataset for training the ML-type algorithm A_{01}

the data (that refers to the similar instances of features describe multiple types of the class label), (v) outliers (that refers to the anomalies of data points that are out of the general behavior of the training dataset), (vi) missing value (that means values that are left blank in the dataset), (vii) amount of training data (that is the number of instances available to train the learning algorithm), (viii) class imbalance (that is related to the number of class labels which are not properly distributed), (ix) learning function (that indicates the types of learning function that should be performed (i.e., linear or non-linear)), and (x) type of dependent variable (that means the types of output value or response of the learning algorithm (i.e., categorical or numerical value).

Considering the rest of the design-sub tasks (i.e., model selection, model scoring, and model validation), they focused on the performances of the learning model. Should the learning model be trained, the prediction variables concerning the performance metrics are considered, including (i) the prediction speed, (ii) robustness of the model, (iii) the flexibility of a computational method to perform the evaluation, (iv) the simplicity of the computational method to perform the evaluation, and (v) the ease of interpretation of the results. Table 4.10 shows the prediction variables, which are related to the design sub-tasks. These variables are analyzed and specified with regard to the considered design entity. As an example, Table 4.11 presents a sample ruleset that was extracted from the decision tree model that was used as selecting the methods related to the design entity 'fitting a model'. It can be remarked that the dataset for training the decision tree was intuitively generated for the demonstrative propose. The correctness of the extracted rules was not theoretically tested.

4.6.5.2 Generation of recommendation according to reference protocol-based procedural obstacle identification

For the demonstrative case, the graph representing the RPP includes 51 design entities and 442 connections. We assumed that all relationships of the design entities are logically valid for the recommendation generation. The expected output is the process-based recommendation, which comprises three design entities and their contents. To develop an efficient ML-type algorithm, the most crucial design sub-task was the model training. In this context we had to assume that a non-usual event could be detected when the designer was fitting the model without the consideration of dataset characteristics.

By the above-described process, in the concrete practical application case, the design entity '*fitting a model* (e_{25})' was identified as the current design action. The contents of this design entity are shown in Figure 4.24. The set of questions was related to some prediction variables of the rule set including: (i) amount of training required, (ii) learning function, (iii)

```
currentEntD =
  struct with fields:

          task: {'Train_Model'}
    identifier: {'fitting a model'}
      inputVar: {'predictors'  'response'}
     outputVar: {'trained_model'}
       methods: {'KNN'}
     durationT: 5.0904
```

Figure 4.24: the content of the identified current design entity

dependent variable, (iv) prediction speed, and (vi) robustness. It might be the case that - due to lack of background knowledge or relevant information - the designer cannot answer these questions. In this case, the ARF will pose sub-questions to guide the designer. The answers given by the designer are used to identify the conditions of the prediction variables (for example, {large, linearity, categorical value, yes, yes}).

Table 4.10: Prediction variables influencing the selection of usable method

	prediction variables	design sub-tasks: data preparation	feature selection	model training	metric selection	model scoring	model validation
1	noise	•					
2	missing value	•					
3	dimensionality	•	•				
4	heterogeneity		•	•			
5	redundancy		•	•			
6	outlies	•					
7	amount of training required			•			
8	class imbalance		•	•			
9	learning function			•	•		
10	dependent variable		•	•			
11	prediction speed			•	•		
12	robustness			•	•		
13	flexibility				•	•	•
14	simplicity				•	•	•
15	ease to interpretation of the results				•	•	•

Based on the combination of the answers, it assumed that the exact inference cannot find the best match pattern for finding the solution. Then, the investigation of the obstacle using the RPP is executed. The joint distribution probability was calculated for selecting the preceding design entity. As result, the design entity, e_{14} – '*select the attributes*', and the method is 'information gain'. It shows two design entities connecting with the red line in the RPP as presented in Figure 4.25. The result can be interpreted that two design entities e_{14} and e_{25} executed by the method '*information gain*' and '*KNN*' are the most frequently used in the historical cases, but they are probably not suited for the identified characteristics of dataset.

It might be the case that the designer dealt with this issue in the one of the historical cases. The prediction process is continued. In another case, if the required data for executing the prediction process are not available, then a dialogue is activated automatically to request (collect) the missing information from the designer. The set of questions related to the design entities e_{14} will be retrieved to organize the dialogue. The designer provides the conditions of prediction variables and the ARF uses the decision tree to predict the best method for the entity e_{14}. In this case, it was found that '*Chi-square test*' was selected and was already defined as the method for the entity e_{15}. It means that this method was used in the historical cases, but it was not the most frequently used with the '*KNN*' (see Figure 4.25, where their relationship is shown by the yellow line in the RPP.)

Table 4.11: A rule set extracted from the decision tree concerning the selection of the usable method for '*fitting the learning algorithm*'

	extracted rule set	recommendation
1	if imbalanced data = no && learning function = linearity && prediction speed = no, then	perceptron based neural network
2	if imbalanced data = no && learning function = linearity && prediction speed = yes, then	decision tree classifier
3	if imbalanced data = no && learning function = non-linearity && amount of training data required = small, then	support vector machine
4	if imbalanced data = no && learning function = non-linearity && amount of training data required = large, then	ensemble classifier
5	if imbalanced data = yes && redundancy = yes && learning function = linearity && prediction speed = no, then	perceptron based neural network
6	if imbalanced data = yes && redundancy = yes && learning function = linearity && prediction speed = yes, then	Bayesian network
7	if imbalanced data = yes && redundancy = yes && learning function = non-linearity && amount of training data required = small, then	probabilistic based neural network
8	if imbalanced data = yes && redundancy = yes && learning function = non-linearity && amount of training data required = large, then	Support vector machine
9	if imbalanced data = yes && redundancy = no && robustness = no, then	K-nearest neighborhood
10	if imbalanced data = yes && redundancy = no && robustness = yes && learning function = linearity, then	probabilistic based neural network
11	if imbalanced data == yes && redundancy == no && robustness == yes && learning function == non-linearity && prediction speed == no, then	K-nearest neighborhood
12	if imbalanced data == yes && redundancy == no && robustness == yes && learning function == non-linearity && prediction speed == yes, then	Gaussian Naïve Bayes

Here, the prediction variables related to the preceding design entity will be combined with the prediction variables of the current design entities. The decision tree will select the proper method for the current one. Based on the combination of these prediction variables, the '*decision tree classifier*' was selected. It was identified as the design entity e_{24} in the RPP. Next step, the joint distribution probability will be determined to select the next design entity. The popularity-based approach is considered as the basis to the case-related recommendation generation. Three design entities are involved in the process.

The result showed that the design entity e_{48} – '*evaluate the performance*' was selected and the method for performing the entity was 'analysis of the classification metrics'. The proposed design activity flow is shown with the green line in the RPP represented in Figure

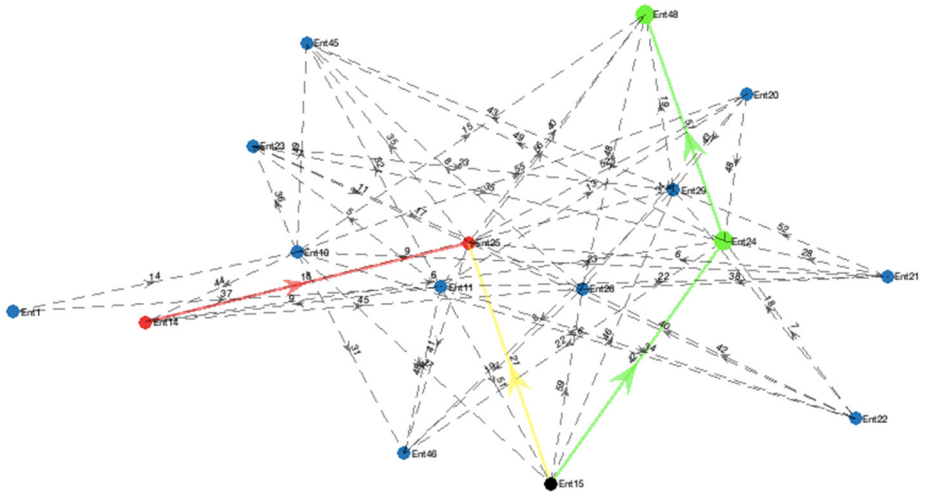


Figure 4.25: The proposed design activity flow (presented as sub-graph representing the RPP)

4.25. The advisory contents for these three elements were generated in the module 5.0 – ACG. The complete recommendation is shown in Figure 4.26, including both the process-based recommendation and the content-based recommendation. To proceed with the proposed design activity flow, the designer can refer to web pages to get more information about the recommended methods.

The ARF makes the recommendation for the designer concerning how to develop the machine learning-type algorithm A_{01}. The comprehensive recommendation presented to the designer is shown in Figure 4.26. We used MATLAB to generate the process-based recommendation provided by the ARF to the designer. The following design actions were performed:

- Design entity e_{15} – selecting the attributes by using the Chi-square test
- Design entity e_{24} – fitting the model by using the decision tree classifier
- Design entity e_{48} – evaluation of the model by the analysis of the classification metrics

	pre_DAaction	DAction_ctx	post_DAction
Task	Feature selection	Train_Model	Validate_model
ID	EnD_15	EnD_24	EnD_48
Design action	Select Attribute	fitting a model	evaluate performances
Input parameter	TrainingSet	predictors && response	trained_model && Optimal_data && exampleSet
Output parameter	predictors && response	trained_model	predictive_model && cost matrix
Method	Chi-Squared test	decision tree classifier	analysis of classification metrics
Description	https://www.mathworks.com/help/stats/fscchi…	https://christophm.github.io/interpretable-ml-bo…	https://www.analyticsvidhya.com/blog/2019/08/11-im…

Figure 4.26: The comprehensive recommendation presented to the designer

For selecting the attributes, the dataset was analyzed by Chi-square test. The advisory content explains the description of Chi-square test[1]. As shown in Figure 4.27, the Chi-square scores of two features (F06 and F07) are relatively very low. Hence, they will be removed for fitting the decision tree classifier. Five features are used for predicting a parking case. To evaluate the performance of the model, the confusion matrix was analyzed in terms of its: (i) accuracy, (ii) precision, (iii) recall, and (iv) F1-measure. The description and the formulas of these metrics were given in Table 4.5 (in the sub-section 4.3.2.3).

The testing sample includes 900 instances for each feature. The trial was run for 500 samples. We also developed the KNN model without the removal of the irrelevant features. The evaluation results are shown in Figure 4.28. The accuracy shows that how many of the correct parking cases that the algorithm predicts. The precision determines how precise the model predicts the predicted positives. It means how many of missing the opportunities for parking. The recall measures how many of the actual positives are predicted as true positive. It means how many of the predicted parking cases cannot be parked, but the model predicts them differently. If the recall value is low, it is high possibility that the predicted positive is not suitable for parking. The F1-measures balances the precision and recall. The performances of the decision tree classifier are slightly better than the KNN model for all metrics. For the demonstrative case, it shows that the recommendation is helpful not only eliminating the obstacle in the design process, but also improving the performance of the learning model.

4.6.5.3 Testing the functionality of the ML-type algorithm A_{01}

To confirm that the ML-type algorithm A_{01} works properly, the selected motion path for parking should be tested in the concrete parking case. The criterion is that the selected motion path should fit the actual situation. It can be further analyzed in the sub-process of generating the parking plan. It should show the reference motion path for parking which can be further analyzed for generating the parking plan. Path planning generation determines a suitable collision-free path from a given start to a required goal position within the parking space.

To capture the motion path, the state of a vehicle is defined by four elements:

$$carMdl = (x,y,\theta,\varphi) \tag{4-64}$$

where: x,y is the coordinates of the location of the vehicle, θ is the orientation angle, and φ is the steering angle.

Since the algorithm is used to predict the parking position, it determines the most similar parking situations and the parking cases based on their spatial information. The algorithms

[1] "Chi-square test examines whether each predictor variable is independent of a response variable by using individual Chi-square tests. A small *p*-value of the test statistic indicates that the corresponding predictor variable is dependent on the response variable, and therefore is an important feature. A large score value indicates that the corresponding predictor is important."

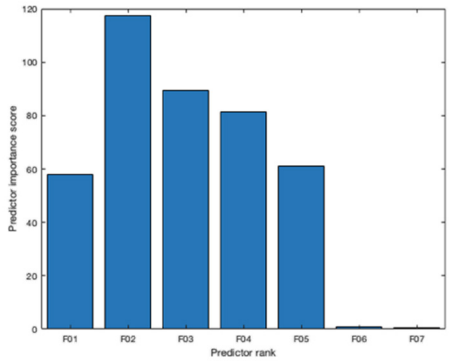


Figure 4.27: Results of the Chi-square test for the feature selection

	Model_KNN	Model_DTree
Accuracy	75.5	90
Precision	97.419	99.448
Recall	77.041	90.452
F-measure	86.04	94.737

Figure 4.28: Performance evaluation of the trained models

have been tried out in two parking scenarios, which are discussed below.

Parallel parking scenario

The right-side Sub-figure of Figure 4.29 shows the most proper parking case for the actual parking situation. The APAS was supposed to have been installed in the blue car. The virtual motion path comprises four steps: (i) moving to the pre-parking location, (ii) moving forward to reach a ready-to-park position, (iii) moving reverse following the motion path to the parking space, and (iv) moving forward to the parking destination. The topological sub-structure of the motion path can be modified to fit other parallel parking situations.

Perpendicular parking scenario

The actual parking situation is illustrated in the left-side Sub-figure of Figure 4.30. The most similar parking case is selected as shown in the right-side Sub-figure. Typically, perpendicular parking is not suited for the roadside parking problem, but the learning algorithm has the potential to predict different types of parking positions. The motion path is for reverse parking. Its topological structure comprises three steps: (i) moving forward to reach a ready-to-park position, (ii) moving reverse following the motion path to the parking space, and (iii) turning to the perpendicular position and moving backward to the destination. Like in the case of parallel parking, the topological sub-structures of the motion path can be modified to fit other perpendicular parking situations.

These parking scenarios showed that the algorithm A_{10} was able to select the proper parking cases for different parking situations. The motion paths presented in these cases might not be the optimal ones for the considered actual situations. Other potential parking cases would be selected as the candidates. In the next step of the reasoning sub-process in the WPE session, the morphological information of the candidate parking cases needs to be extracted and analyzed to find the optimal motion path.

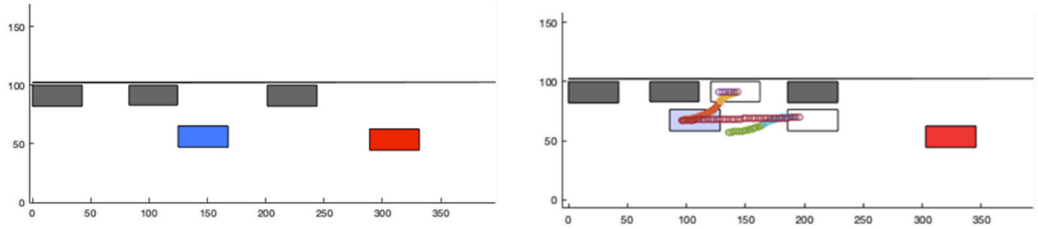


Figure 4.29: The actual parking situation (left) and the retrieved parking case (right) for the parallel parking

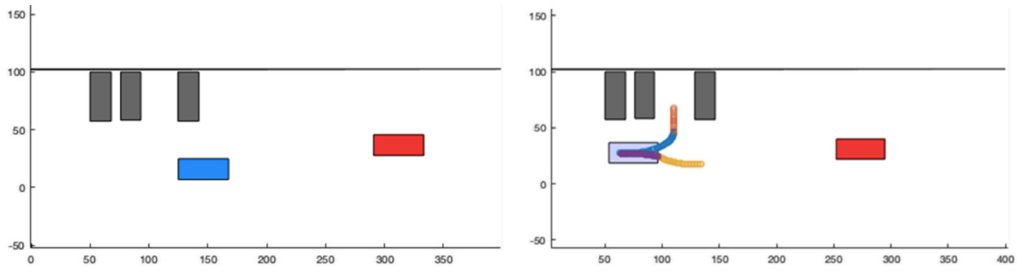


Figure 4.30: The actual parking situation (left) and the retrieved parking case (right) for the perpendicular parking

4.7 Observed limitations and other concluding remarks

4.7.1 Observed limitations of the demonstrative implementation

- Due to the lack of the knowledge contents for all design entities, the demonstrative implementation cannot provide support for the entire process of completing design task $\mathfrak{D}_{1.0}$. In addition, the recommendation generation was also limited because of the scope of the design scenario set up for the to-be-considered design actions.
- We used string-based similarity to create relationships among the design entities for the construction of the RPP. Some terms denoting the input and output variables were mismatched because they were depicted by the same terms but, in fact, their parameters and data constructs were used in different contexts.
- The RPP was constructed based on the concept of a Bayesian network. It is represented as an acyclic graph. Hence, using the RPP in the recommendation generation cannot support iterative loops occurring in the design process. Further research is necessary to resolve this logical issue.
- In the course of functionality testing, the recommendations were generated based on various assumptions concerning the theoretical concepts and the practical design processes. However, the knowledge contents of the RPP and the decision variables of

the decision tree model were not theoretically tested. Hence, biases might influence the findings concerning the recommendations, and the comparative performance of the ML algorithms.

4.7.2 Improvement opportunities of the demonstrative implementation

- The algorithms implemented for a demonstrative purpose. We mainly concentrated on the realization of the novel and critical functionalities. Although, the aspect of efficiency of the algorithms was not considered yet, the organization of the computational workflow provided opportunities to further investigate the potential optimization of the developed algorithms.
- To avoid fundamental mistakes concerning the contents of the generated recommendations, the theoretical correctness of the reference protocol and its knowledge contents should be validated.
- It is possible to develop semantic inference for the recommendation generation based on the relationships of the knowledge contents included in the RPP. For example, if the designer can provide the preference conditions of the decision variables (which are related to the multiple usable methods), the ARF will infer the flow of the design actions based on the relationships of selected methods.

4.7.3 Concluding remarks

- The implementation and functionality testing of the modules and algorithms of the demonstrative part of the ARF confirmed their correctness from logical and computational points of view.
- Based on the doctrine of hybrid inference, the application of the RPP supports a (semi) automated recommendation generation.
- Using a traceable logical reasoning model, such as the decision tree model, is useful for helping the designer to understand the logic of the decision-making concerning the selection of usable methods.
- Applying probabilistic reasoning makes it possible to generate case-related recommendations based on the popularity of the methods in previous (historical) cases.
- The testing of the demonstrative modules in the context of APAS confirmed the usefulness of the proposed recommendation generation in terms of elimination of an obstacle in the design process and of the possibility of continuation of the design process.

References

[1] Horváth, I. (2008). Differences between ‘research in design context’ and ‘design inclusive research’ in the domain of industrial design engineering. *Journal of Design. Research, 7*(1), 61.

[2] Cass, S., "Top Programming Languages 2020," *IEEE Spertrum*, 2020. Retrieve from https://spectrum.ieee.org/top-programming-language-2020 (accessed Jul. 20, 2021).
[3] Jurafsky, D., & Martin, J. (2017). Dialog systems and chatbots. Speech Language Process (pp. 418–440).
[4] Madotto, A., Lin, Z., Wu, C. S., Shin, J., & Fung, P. (2020). Attention over parameters for dialogue systems. arXiv preprint arXiv:2001.01871.
[5] Gehlot, V., & Nigro, C. (2010). An introduction to systems modeling and simulation with colored petri nets. In *Proceedings of the Winter Simulation Conference* (pp. 104-118). IEEE.
[6] Linda, S., & Bharadwaj, K. K. (2018). A decision tree dased context-aware recommender system. In *International Conference on Intelligent Human Computer Interaction* (pp. 293-305). Springer, Cham.
[7] Jena, M., & Dehuri, S. (2020). Decision tree for classification and regression: A state-of-the art review. *Informatics, 44*(4), 405–420.
[8] Rudin, C., & Radin, J. (2019). Why are we using black box models in AI when we don't need to? a lesson from an explainable AI competition. Harvard Data Science. *Review, 1*(2), 1–9.
[9] Hehn, T. M., Kooij, J. F. P., & Hamprecht, F. A., (2020). End-to-End learning of decision trees and forests. *International Journal of Computer Vision, 128*(4), 997–1011.
[10] Liu, S., Yang, Z., Li, Y., & Wang, S. (2020). Decision tree-based sensitive information identification and encrypted transmission system. *Entropy, 22*(2), 192.
[11] Viaene, S., Derrig, R. A., Baesens, B., & Dedene, G. (2002). A comparison of state-of-the art classification techniques for expert automobile insurance claim fraud detection. *Journal of Risk and Insurance, 69*(3), 373-421.
[12] Çiğşar, B., & Ünal, D. (2019). Comparison of Data Mining Classification Algorithms Determining the Default Risk. *Scientific Programming 2019*.
[13] Cózar, J., Puerta, J. M., & Gámez, J.A. (2017). An application of dynamic Bayesian networks to condition monitoring and fault prediction in a sensored system: A case study. *International Journal of Computational Intelligence Systems, 10*(1), 176–195.
[14] Dejaeger, K., Verbraken, T., & Baesens, B. (2013). Toward comprehensible software fault prediction models using bayesian network classifiers. *IEEE Transactions of Software Engineering, 39*(2), 237–257.
[15] Jimenez, S., Cucerzan, S. P., Gonzalez, F. A., Gelbukh, A., & Dueñas, G. (2018). BM25-CTF: Improving TF and IDF factors in BM25 by using collection term frequencies. *Journal of Intelligent & Fuzzy Systems, 34*(5), 2887-2899.
[16] Mohamed, K. (2021). "Ackermann steering(Car Auto parking)," *MATLAB Central File Exchange*. Retrieve from https://www.mathworks.com/matlabcentral/fileexchange /27299-ackermann -steering-car-auto-parking (accessed Oct. 05, 2021).
[17] Ma, Y., Liu, Y., Zhang, L., Cao, Y., Guo, S., & Li, H. (2021). Research review on parking space detection method. *Symmetry, 13*(1), 128.
[18] Rathore, S. S., & Kumar, S. (2017). A decision tree logic based recommendation system to select software fault prediction techniques. *Computing, 99*(3), 255–285.

Chapter 5

Research cycle 4:
Validation of the usefulness of the recommendation provided by the implemented demostrative module

5.1 Objectives and methodological framing of the fourth research cycle

5.1.1 Research objectives

The computational implementation of the selected demonstrative modules was presented in the previous chapter together with a forerunning analysis of the fulfillment of the functions implied by the technical requirements. As shown by the completed functional validation in the application context, the ML-type classifier was able to select the proper parking case from the available parking scenarios. The functional validation also showed that the implemented demonstrative modules of the ARF could support the designer with conducting the design process in the application context. This chapter focuses on the quality of the recommendations provided by the ARF. Usefulness was chosen as the measure of the quality because it could be captured by indicators (and not only by quantitative and/or qualitative variables).

Usefulness was viewed from the perspective of the designer, and is captured by the observable procedural effects of providing situation and obstacle dependent information in the semantic body of the recommendation generated by the concerned modules of the ARF. In other words, it was examined how useful the provided recommendations are to overcome possible procedural obstacles in the design process by the designer. Usefulness of the recommendation was assessed in a retrospective way (i.e., after the emergence of some procedural blockage), but not in a predictive (preventive) way. Due to the complexity and sophistication of this matter, it was left for future research. Thus, the goal of the validation

study presented in the rest of the Chapter was to confirm that the recommendations generated by the ARF were useful in various situations in the studied application context according to the proposed measurable metrics.

The main issue for the validation of usefulness of the recommendations was the active and insightful involvement of designers in the design process. It was seen as the most influential factor for finding the appropriate method of validation. It was considered with emphasis that the current developments in smart agent technologies make it possible to replace human participants with smart validation entities in repetitive assessment processes. Such programmable surrogates can mimic (i) creative activities; (ii) decision-making actions; and (iii) the social behavior of human designers. Current research knowledge allows for achieving high-fidelity multi-feature surrogates. Having considered all these influential factors and the trends and affordances of technological development, we preferred to conduct the validation study by using a synthetic validation agent (that is, without participation of human designers). Therefore, an agent-oriented validation method was devised and applied. In order to use this approach to usefulness validation, a synthetic validation agent (SVA) had to be designed, implemented, and pre-tested carefully.

Concerning the focus of the abovementioned of usefulness validation, six concrete goals were set: (i) to specify the target design process and actions in the context of the preferred application; (ii) to determine a proper approach to usefulness validation in the context of the conducted design process; (iii) to develop a synthetic validation agent that mimics the decisions and the behavior of designers; (iv) to investigate the contribution of the parts of the demonstrative modules to validation; (v) to assess the usefulness of the provided recommendations in terms of an unbroken progression in the design process; and (vi) to evaluate the findings and making enhancement proposals towards a better provisioning and contents of recommendations.

5.1.2 Methodological framing of the fourth research cycle

As introduced above, the object of validation was the usefulness of the recommendations received by the designer with respect to knowing how to progress in the design process (i.e., which design action to prefer as the next one.). From a procedural viewpoint, usefulness depends on the contents of the recommendations, which should be sensitive (i) to the progress (state) of the design process, (ii) to the number of design actions may be completed, and (iii) to the accomplishment pursued by the designer (contrary to the possible procedural inconsistences).

The activities in the fourth research cycle were driven by the need for both (i) data exploration and (ii) validation of the findings about usefulness as shown in Figure 5.1. Data were supposed to express the effects of recommendations on progression of the designer in the design process. They also had to help with hypothesizing an explanatory theory about the conditions and measure of usefulness. These determined the major activities in the exploratory part of the research cycle.

Structurally, the exploratory part of the research cycle included (i) a preparation stage (discussed in Sub-chapter 5.3) and (ii) an execution of validation stage (discussed in Sub-chapter 5.4). The implemented SVA was used in the exploratory part of the research cycle, which allowed for hypothesizing an explanatory theory about the usefulness of the situation-related recommendations at the end of this sub-chapter. The exploratory part and the confirmatory part of the research cycle were connected through this explanatory theory.

The confirmatory part of the research cycle focused on the consolidation of this explanation considering a representative set of design actions and progression situations. Statistical analysis was used as a method of testing in this part of the research cycle. Beginning with the meaning and implications of the theory, the confirmatory part of the research cycle included (i) an evaluation of the findings stage (discussed in Sub-chapter 5.5) and (ii) a discussion and interpretation of the findings stage (discussed in Sub-chapter 5.6). Reflecting the ideas of practice-based research (PBR), the goal of the validation was to generate information not only about the usefulness of the recommendation, but also about opportunities for improvement.

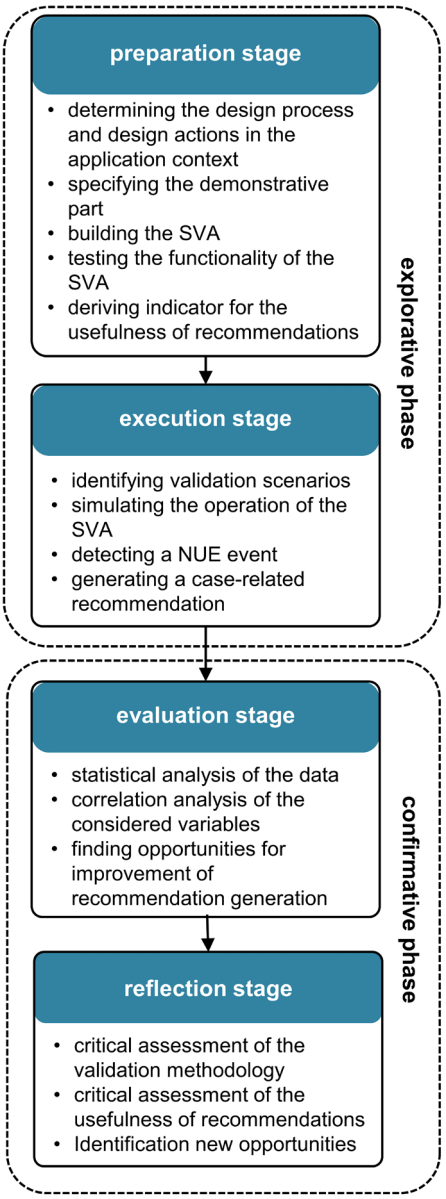


Figure 5.1: Approach of RC4

5.2 Main issues of validation of the demonstrative implementation part

In this sub-chapter, we discuss the issues that were associated with, and influenced the validation of, the usefulness of the provided recommendations. The two main issues were: (i) the criteria and measures for evaluation, and (ii) the method of evaluation.

5.2.1 Criteria and measures of usefulness validation of procedural recommendations

We investigated the metrics that had been used to evaluate the performance of recommendation systems. One of our observations was that measures of algorithm performance were typically used as evaluation metrics in the case of recommender systems. Among others, time-dependency, user behaviors, and computational costs were considered as influencing factors and aspects of evaluation metrics [1]. In the numerical evaluation either various statistical methods (e.g., root mean square error method, mean reciprocal rank method, etc.), or machine learning metrics (e.g., accuracy-based metrics, F-measure, etc.) were used.

Considering the objective of usefulness analysis, we had to conclude that the usual performance evaluation metrics could not capture the very essence of usefulness. The simple reason was that evaluation of the effects of the provided recommendations was not a crucial issue in the case of the studied recommendation systems. This raised the need for more relevant metrics for usefulness validation. Towards this end, new aspects had to be considered such as (i) the perceived cognitive support (e.g., originality – which means recommending something new the users did not know or had thought of before); (ii) the perceived practical influence (e.g., time saving – which means that the recommendation helps reduce search time); and (iii) the perceived quality of the presentation (e.g., sufficiency of description – which means providing recommendation content adjusted to the designer) [2].

In the literature, usefulness is usually interpreted as utilitarian goodness and not as instrumental (usability), pragmatic (problem solving), technical (performance), hedonic (pleasure), and/or transcendental (mystic) goodness [3]. Thus, utilitarian goodness has been defined as the possible target and the strategic objective of validation. Our hypothesis was that a useful recommendation delivers information content that (i) is relevant for the continuation of the design process, (ii) the designer can recognize if it corrects or not, and (iii) the designer can operationalize the best next design action based on it.

5.2.2 Consideration of a simplified decisional behavior of the designers

The issue of having a proper evaluation method orientated our attention to the decisional behavior of designers. The major question was how the designers arrive at decisions and what they consider when making a decision about the design actions for the execution of the design process. Related to the latter, a decisional option S_i, is created for the designer when - after finishing some related preceding design actions and having their output data – she/he selects and operationalizes the next design action. A challenge may be caused by the multiplicity of the possible design actions. It was considered that decisional situations were determined by the interplay of three decision variables. They are shown and interpreted in Figure 5.2.

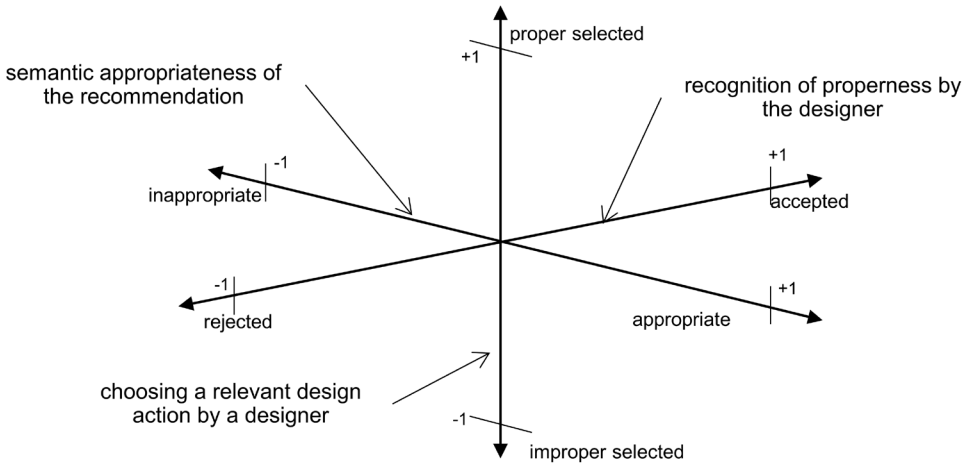


Figure 5.2: Three aspects (or decision variables) of decisional behaviors

The first decision variable was the semantic appropriateness of a recommendation, $\mathcal{R}_i$. Based on the actual inference of the concerned module of the ARF, a recommendation may be appropriate or inappropriate from the aspect of supporting a decision related to choosing the next correct design action to continue the design process with. The second decision variable was the recognition of the properness of the recommendation by the designer in the given procedural context $\mathcal{D}_i$. In spite of its trueness, a proper recommendation can be rejected. On the other hand, the designer may consider an improper recommendation as true and can accept it. In addition, a third decision variable $\mathcal{A}_i$, was considered to express the possible decision alternatives of the designer on the right design action. The underpinning argumentation of this as an influential factor is that, based on the recommendation, the designer may select a proper design action, but there is a chance of failure at selecting the proper design action due to the abovementioned multiplicity or due to a personal mistake. On the other hand, incidental proper selection may also occur.

Using these decision variables, the decisional options could be interpreted as a triplet of relations. Symbolically:

$$S_i = (\mathcal{R}_i, \mathcal{D}_i, \mathcal{A}_i) \qquad (5\text{-}1)$$

These were supposed to occur in handling appropriate and inappropriate recommendations by the designer. An exhaustive combination of the decisional options leads to a model of the decisional behavior of the designer. Eventually, the model captures the eight decisional options concerning the acceptance of a recommendation about taking the next design actions. The possible combinations of the eight options were sorted into four classes, (i) justified objective decision; (ii) unjustified objective decision; (iii) incorrect objective decision; and (iv) negatively justified objective decision. The combinations of these options are shown in

Table 5.1. The descriptions of the decisional options are given below:

- A ***type I justified objective decision*** means that an appropriate recommendation is received by the designer, who accepts it and selects a proper design action based on the content of the recommendation.
- A ***type I unjustified subjective decision*** means that an appropriate recommendation is received by the designer, who rejects it but - contrary to this - finds a proper design action independent of the content of the recommendation.
- A ***type II unjustified subjective decision*** means that an inappropriate recommendation is received by the designer, who accepts it but - for a certain reason - finds a proper design action independent of the content of the recommendation.
- A ***type III unjustified subjective decision*** means that an inappropriate recommendation is received by the designer, who rejects it and - contrary to this - finds a proper design action independent of the content of the recommendation.
- A ***type I incorrect subjective decision*** means that an appropriate recommendation is received by the designer, who accepts it but – for a certain reason - selects an improper design action.
- A ***type II incorrect subjective decision*** means that an appropriate recommendation is received by the designer, who rejects it but – for a certain reason - selects an improper design action.
- A ***type III incorrect subjective decision*** means that an inappropriate recommendation is received by the designer, who accepts it, but - contrary to this - selects an improper design action.
- A ***type II justified objective decision*** means that an inappropriate recommendation is received by the designer, who rejects it, but cannot find a proper design action without asking for other recommendation.

Table 5.1: Options of decision making by the designer

aspects of evaluation		**recommendation provided by the ARF**			
		appropriate recommendation		inappropriate recommendation	
		acceptance by the designer		**acceptance by the designer**	
		accepted	rejected	accepted	rejected
decision on design action	proper design action selected	justified objective decision I (+, +, +)	unjustified subjective decision I (+, -, +)	unjustified subjective decision II (-, +, +)	unjustified subjective decision III (-, -, +)
	improper design action selected	incorrect subjective decision I (+, +, -)	incorrect subjective decision II (+, -, -)	incorrect subjective decision III (-, +, -)	justified objective decision II (-, -, -)

This set of options of decision-making lends itself to a formal logic that can be applied in computation for the operation of the SVA, but it also needs to fulfil some other operational requirements that are discussed below.

5.2.3 Reason and requirements for a synthetic validation agent

We defined the term 'usefulness' with a specific meaning: a recommendation is useful if the designer can unblock a procedural obstacle and continue the design process. We intended to assess the usefulness of the recommendations by conducting real life experimentation. Towards this goal, initially, we planned to conduct the usefulness validation study by the involvement of a probabilistic sample of software algorithm designers and programmers. According to the plans, the designers would have used the demonstrative part of the ARF and could assess the usefulness of the provided recommendations. However, we faced difficulty with the involvement of practicing designers. The difficulty was caused by the coronavirus pandemic, which did not make it possible for us to invite designers to conduct the planned on-site studies. This orientated our attention to other alternative methodologies and computational solutions.

The search for a surrogate method for the validation of the usefulness of the recommendations led us the concept of a synthetic validation agent (SVA). By creating an SVA we intend to mimic the decisional behavior of the designer and to generate a quasi-experimental dataset for the purpose of validation. The above-discussed three decision variables were considered as the key elements of the formal reasoning procedure of the SVA. The SVA has been conceptualized as a generic means of modeling multiple designers with different behavioral patterns in terms of (i) recognizing the appropriateness of recommendations, and (ii) executing the implied design action as a correct continuation of the design process.

From a computational point of view, the SVA generates datasets that mimic the (human) designers' decisional behavior as a follow-up on the proposed recommendations. The principle of prognostic reasoning was used to predict the possibilities of decision options based on the mimicked decisional behavior. As a result, a socially-based indicator could be realized to express the usefulness of the obtained recommendations based on the justified objective decisions of the designer. In simple terms, implemented in the form of agent-based behavior simulator, the SVA was used to simulate if the designer could select the next correct design action or not.

The SVA was expected to fulfill two types of requirements concerning: (i) simulation of decisional behaviors, and (ii) generation of decision data and distributions. The concrete requirements are listed in Table 5.2. The requirements concerning decisional behaviors formulated the expectation for how a designer responds to the offered recommendations (which include four options by the combination of acceptance and rejection of the appropriate/impropriate recommendations). The requirements concerning decisional data are about the criteria for generating representative probabilistic distribution of the

decisional results (i.e., the pattern of choosing proper design actions).

From a computational point of view, the SVA generates datasets that mimic the (human) designers' decisional behavior as a follow-up on the proposed recommendations. The principle of prognostic reasoning was used to predict the possibilities of decision options based on the mimicked decisional behavior. As a result, a socially-based indicator could be realized to express the usefulness of the obtained recommendations based on the justified objective decisions of the designer. In simple terms, implemented in the form of agent-based behavior simulator, the SVA was used to simulate if the designer could select the next correct design action or not.

The SVA was expected to fulfill two types of requirements concerning: (i) simulation of decisional behaviors, and (ii) generation of decision data and distributions. The concrete requirements are listed in Table 5.2. The requirements concerning decisional behaviors formulated the expectation for how a designer responds to the offered recommendations (which include four options by the combination of acceptance and rejection of the appropriate/impropriate recommendations). The requirements concerning decisional data are about the criteria for generating representative probabilistic distribution of the decisional results (i.e., the pattern of choosing proper design actions).

5.2.4 The process of validation of the usefulness of procedural recommendations

The validation process was decomposed into three stages, which were: (i) preparation, (ii) execution, and (iii) evaluation. Each stage was then decomposed into multiple research activities completed by using specific methods and instruments as shown in Table 5.3.

- The ***preparation stage*** included four research activities: (i) specification of design process and actions; (ii) identification of the demonstrative part involved in recommendation generation; (iii) identification of the objective of evaluation; and (iv) development of the synthetic validation agent.
- The ***execution stage*** operationalized the computational mechanism of recommendation generation as well as the validation agent in the application context. The research activities consisted of: (i) identification of validation scenarios; (ii) activation of the validation agent; (iii) detection of an unexpected event; and (iv) generation of a case-based recommendation.
- The ***evaluation stage*** focused on a statistical analysis of the simulation results. There were three research activities included in this stage: (i) statistical analysis of data; (ii) finding correlations between the considered variables and the indicator of usefulness; and (iii) analysis of the findings concerning improvement opportunities of the recommendation generation.

Table 5.2: List of functional requirements for the synthetic validation agent

aspects of requirements	requirements
simulation of elements of decisional behaviors	acceptance of an appropriate recommendation by the designer
	acceptance of an inappropriate recommendation by the designer
	rejection of an appropriate recommendation by the designer
	rejection of an inappropriate recommendation by the designer
	aggregated probability of the acceptance/rejection (appropriate/ inappropriate)
	the decisional tendency/probability of selecting a proper/improper design action by the designer
	combine the tendency statistics with the aggregated probability
	offer a tested behavioral pattern
generation of decision data and distributions	simulate the operation of recommendation process
	generate appropriate recommendations in a statistically normally distributed manner
	generate inappropriate recommendations in a statistically normally distributed manner
	predict the probability of decisional options of the recommendation

5.3 Preparation stage of the validation process

5.3.1 Specification of design activities in the application context

In Chapter 4, we discussed how the functionality of the implemented demonstrative modules was tested in the case of an automatic parking system (APAS), as application context. In the center of this functionality testing was the ML-type algorithm (A_{01}), which had been constructed for searching for past parking cases and retrieving the best matching ones from the repository of the APAS according to the actual parking situation.

The validation of the recommendations was also associated with this part of design process. The following design activities were completed in the application context. The considered part of the design process included six design sub-tasks ranging from the pre-processing of the dataset to the evaluation of the trained machine learning model (see the detailed descriptions of the design-sub tasks in Sub-section 4.6.4 in chapter 4). The input data was a set of sensor data representing the parking situations.

The application designer had to execute one or more design actions to complete each individual task. The potential design actions (see Table 5.6 in Chapter 4), which could be selected by the designer, were all elements of the reference process protocol. The creative aspect of the design process was designing information constructs for training, and its

Table 5.3: Process of validation of the usefulness of procedural recommendations

	stage	methods	Instruments
1.0	**preparation stage**		
1.1	specification of the design process and actions in application context	content analysis and synthesis	
1.2	using the demonstrative part for recommendation generation		
1.3	development of the synthetic validation agent	analytic requirements specification	
		creative system thinking	programming in a software tool
		computer coding	
1.4	testing the synthetic validation agent	pilot functional testing	
1.5	deriving indicator of usefulness	prognostic reasoning	
2.0	**execution stage**		
2.1	identification of validation scenarios	scenario analysis	programming in a software tool
2.2	operationalization of the synthesis validation agent	operationalization of the implemented mechanisms of the ARF and the synthetic validation agent (SVA)	
2.3	detection of an unexpected event		
2.4	generation of a case-related recommendation		
3.0	**evaluation stage**		
3.1	statistical analysis of the data	statistical analysis	statistical analysis toolbox provided by a software tool
3.2	correlation analysis of the considered variables and the decisional options	correlation analysis and synthesis	
3.3	improvement opportunities for the recommendation generation mechanism	system thinking and synthesis	

expected output was the classification model. This model was used for selection of the parking case which matches the actual parking situation best.

5.3.2 Identification of the implemented modules taking part in recommendation generation

Only certain components of the implemented demonstrative modules of the ARF contribute to generation of recommendations. Their contributions vary. Some of them derive information about the procedural situation, others consider reasons about the implications of the situations, and yet others construct and communicate the recommendations. These components of the ARF are associated with two mechanisms, namely: (i) the process monitoring mechanism, which detects a non-usual event and a related procedural obstacle in the design process; and (ii) the decision support mechanism, which provides the

recommendation to remove the detected obstacle and to continue the design process. The functionality of the mechanisms is added up by the lowest level sub-functions, which are implemented by the computational algorithms of the ARF. The implemented algorithms were purposively integrated into the computational components.

Two alternative approaches were used to generate the recommendations - as shown in Figure 5.3. The essence of the first approach was that the pattern of the designer's responses was matched to the decision table within the DOI module. In other words, an interrogative interaction with the designer was organized and used as source of information. The essence of the second approach was that the reference process protocol was operationalized within the ROI module for the same purpose. From the viewpoint of recommendation generation, this second approach has a higher significance. For this reason, only the components concerned with this approach were involved in the validation study.

5.3.3 Development of the synthetic validation agent as surrogate of the designer

5.3.3.1 Fundamental concepts for deriving the decisional model for the agent

First things first …

A synthetic validation agent (SVA) can be viewed as a system that is situated in some environment and is capable of autonomous action in that environment in order to meet its design objectives [4]. In our context, the SVA mimics the flow of procedural decisions of the human designer as they are made after obtaining the recommendations. The expected output of the behavioral simulation made by using the agent is a data set that includes the patterns of the decisional behavior of the designer. We aimed at using this synthesized dataset to validate the usefulness of the individual recommendations. Striving for this, the

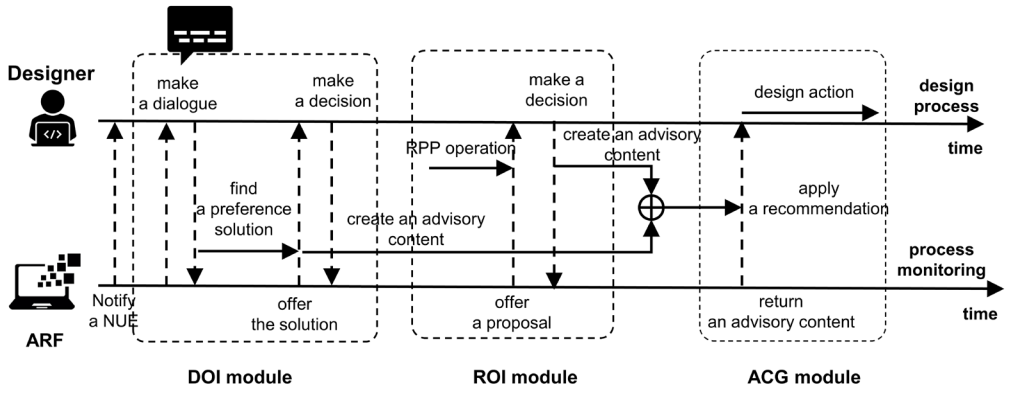


Figure 5.3: Alternative approaches of recommendation generation

challenge was how to capture the features of the decisional behavior of the designer with a view to the obtained recommendations.

It was clarified in the preceding chapters that the RPP is a knowledge-based representation of the design process of ASRMs from the perspective of knowledge engineering. The ARF uses the RPP to generate context sensitive recommendations, which carry specific information for the designer in a particular (necessitating) situation. On the other hand, the designer is supposed to have her/his own knowledge about the conducted design process. Putting together these, the assumption can be posited that the designer shares some common knowledge elements with the RPP. When it comes to accepting or rejecting the recommendation in the case of hindrance, the designer decides on the next best probable design action based on her/his familiarity with the solution generation process and the informativeness (usefulness) of the obtained recommendation.

Assumptions concerning the decisional modes for the agent

The designer's 'decision mode' is interpreted as the knowledge possession-dependent decisional potential of one individual designer without any further knowledge search, sharing, or communicating with others. It was assumed that an actual decision mode of the designer depends on the proportion of the common knowledge (c_k) characterizing the relationship of the designer's knowledge to the knowledge formally captured in the RPP. Thus, c_k can be formally defined as the proportion of the knowledge the designer has related to the design process, which she/he is actually working on, as compared to the knowledge formally captured in the RPP. Formally:

$$E_c = E_r \cap E_d \tag{5-2}$$

$$c_k = n(E_c)/n\,(E_r) \tag{5-3}$$

where: c_k is the proportion common knowledge, $0 < c_k < 1$, E_r is a finite, non-empty set of design entities included in the RPP, $n(E_c) \in \mathbb{Z}$ is the total number of design entities included in the RPP, $n(E_r) > 0$, E_d is a finite, non-empty set of design actions, which are supposed to be known by the designer, and E_c is a finite set of common design entities cognitively shared by the designer and the RPP.

It is very unlikely that there is a 100 percent coincidence with regards to the knowledge of the designer and the knowledge contained in the RPP. It is also very unlikely that there is a 0 percent coincidence between them in a given project because the designer does not have competence enough to deal with the problem in that case. Therefore, we took into consideration the statistical significance of coincidence and operationalized the statistical significance parameter p, with regard to the coincidence. We hypothesized and defined three (most likely) domains of the shared knowledge as follows: (i) (0.05 - 0.25) (assumed in the case of a less competent designer), (ii) (0.25 - 0.75) (assumed in the case of an intermediately competent designer), and (iii) (0.75 - 0.95) (assumed in the case of a highly competent designer).

Based on this interpretation, we introduced the concept of decisional modes (Δ_i) as the representatives of the proportion of the common knowledge indicated by c_k. Formally:

$$\Delta_i = \begin{cases} \Delta_1, & 0.05 < c_k \leq 0.25 \\ \Delta_2, & 0.25 < c_k \leq 0.75 \\ \Delta_3, & 0.75 < c_k \leq 0.95 \end{cases} \quad (5\text{-}4)$$

With regard to the practical design actions, the characteristic of three decisional modes can be described as follows:

Δ_1: The proportion of the common knowledge c_k, is between 0.05-0.25. In this case, the designer possesses insufficient knowledge to judge the appropriateness of the received recommendation, and she/he has no other options to continue the design process. In this decision mode, there is a high probability that the recommendation will be accepted. The probability will be slightly decreasing when the ratio of the shared knowledge is increasing.

Δ_2: The proportion of the common knowledge c_k, is between 0.25-0.75. In this case, the designer may recognize that the recommendation is appropriate to eliminate the obstacle in the design process, but she/he may also have other options originating in her/his experiences with the design process to be completed. In this decision mode, the designer may hesitate to accept a recommendation that she/he is not familiar with. For the designer using the familiar design action and method may seem to be a better choice. Therefore, the probability of accepting the received recommendation is getting lower.

Δ_3: The proportion of the common knowledge c_k, is between 0.75-0.95. In this case, the designer possesses the necessary and sufficient knowledge of the design process to assess the appropriateness of the received recommendation. In this decision mode, the designer also recognizes how to operationalize the recommendation. Consequently, there is a high probability of accepting the recommendation.

The three decisional modes together form a so-called decisional model. This will be clarified and further elaborated on below.

Formal representation of the fundamental concept

A decisional model is assumed to be a pattern of decisional behaviors concerning the acceptance or rejection of a recommendation. We introduced three formulas to capture the pattern, that is for: (i) the probability of the acceptance of a recommendation $p(aR)$, (ii) the probability of shared knowledge $p(s_k)$, and (iii) the probability of knowing the knowledge elements included in the recommendation by the designer $p(f_k)$. These formulas are as follows:

$$p(aR) = f(s_k, f_k) \quad (5\text{-}5)$$

$$p(s_k) = n(E_p \cap E_c)/n(E_p \cup E_c) \quad (5\text{-}6)$$

$$p(f_k) = n(E_p \cap E_c)/n(E_c) \quad (5\text{-}7)$$

where: $p(aR_i)$ is the probability of the acceptance of a recommendation, E_p is a finite, non-empty set of design entities included in the considered segment of the design process, and E_c is as specified by Equation (5-2).

Graphical representation of the connectivity of the knowledge elements

The RPP can be visualized by a circular graph as shown in Figure 5.4.a. It captures both the knowledge elements and their connectivity that serve as the basis of decision-making by the synthetic validation agent (SVA).

The dots in Figure 5.4.a represent the design entities and the dashed lines represent the relationships among the entities. The red dots represent the common knowledge elements shared by the designer and the RPP. The yellow dots represent the concerned segment of the process flow models. The green dots are the knowledge elements shared in the recommendation by the designer.

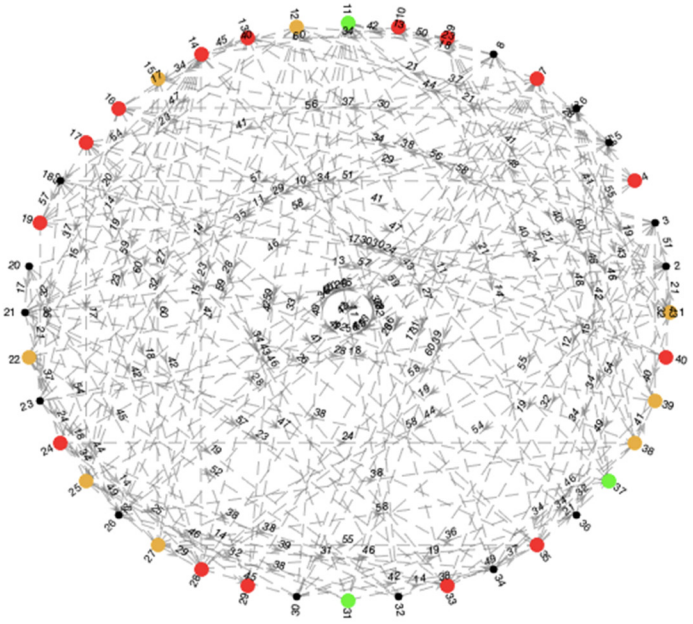
As shown in Figure 5.4.b, the above-described sharing of knowledge can also be represented by a Venn diagram. This is, however, only an abstract and simplified model of the connectivity of the design entities. As an aggregate of knowledge elements, the set E_c includes those common design entities which are shared by the designer and the RPP. PFMs is the set of process flow models, and E_p is as specified above. The intersection (common parts) of these sets represents the shared knowledge elements, a part of which is actually carried by the recommendation, (REC).

Constructing the decision model for the agent

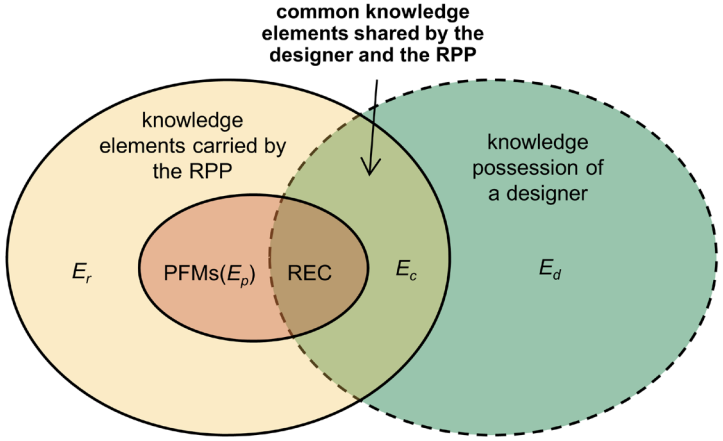
Based on the consideration of these possible decisional situations, a decision model was constructed in the form of a function specified by Equation (5-8). This function captures the deviation of the probability function of the shared knowledge $p(s_k)$, from the probability function of the knowledge elements known by the designer $p(f_k)$.

$$p(aR) = 1 - \alpha * (exp(\beta * p(s_k)) - exp(\gamma * p(f_k)) \quad (5\text{-}8)$$

where: α is the expected probability of accepting a recommendation and $0 \leq \alpha \leq 1$, β is a factor (coefficient) determining the acceptance level, and γ is the expected acceptance level, and $\gamma < \beta < 0$. A broad scrutiny (logical examination) was applied to test the rationality of this decision model. Furthermore, to determine the values of the above defined coefficients (factors) of the reference decision model, we set up design scenarios and numerically analyzed the outcomes. Figure 5.5 shows the graphical representation of the relationships of the decisional modes in the decision model, as implemented in the SVA.



(a) Graphical representation of the connectivity of the knowledge elements included in the RPP



(b) The relations of the knowledge possession of the designer and the knowledge elements related to the RPP

Figure 5.4: Visual representation of the fundamental concept for deriving the decision model of the agent

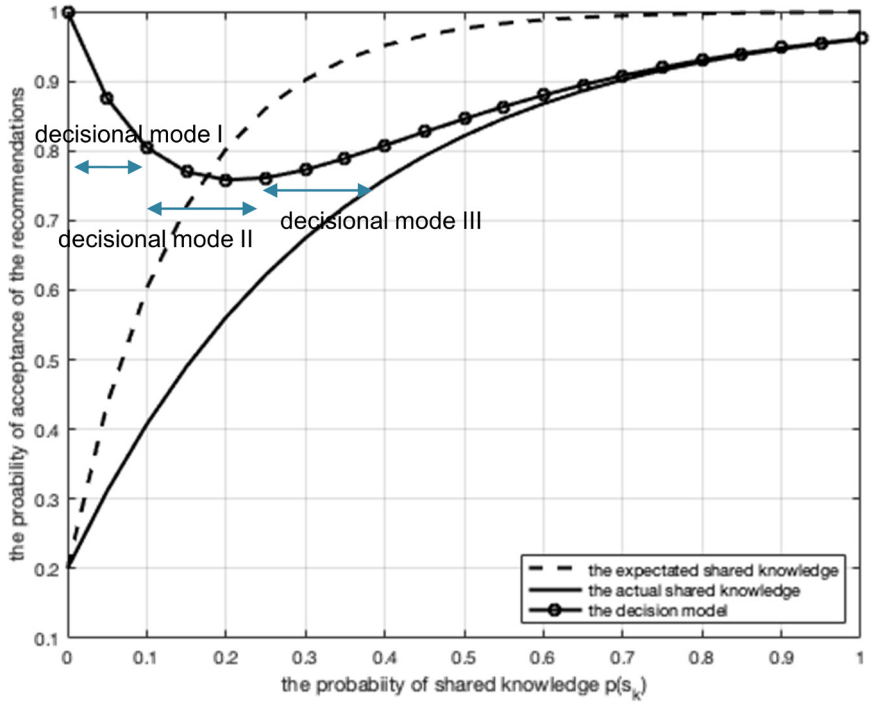


Figure 5.5: The relations of the decisional modes as elements of the decisional model constructed for the SVA

5.3.3.2 Specification of the functionality of the synthetic validation agent

As a generic (system-level) function, the SVA generates a validation dataset that mimics the decision patterns of human designers. The requirements specified in Section 5.2.2 were used to guide the process of decomposition. In order to facilitate the fulfilment of the posed requirements, the mentioned generic function was decomposed into two main functions which are: (i) to simulate decisional behaviors ($F_{1.0}$); and (ii) to generate validation dataset ($F_{2.0}$). The computational sub-functions of the validation agent were derived by a second-level decomposition of these main functions. The result of these decompositions is shown in Figure 5.6. Below, we explain the sub-functions of the SVA in more detail.

The sub-function $F_{1.1}$ simulates how much the designer intends to accept the offered recommendation. This determines the acceptance probability of the recommendations based on the reference decision model. Using this acceptance probability, the sub-function $F_{1.2}$ predicts the probability of the behavioral features considering the decisional options. The outcome mimics the behavior of the designer in terms of recognizing the appropriateness of the recommendations. Considering the possible interplay of the pairs of positive (approving) and negative (disproving) decision options, four behavioral options (features)

were identified: (i) accepting an appropriate recommendation; (ii) rejecting an appropriate recommendation, (iii) accepting an inappropriate recommendation, and (iv) rejecting an inappropriate recommendation. The sub-function $F_{1.3}$ simulates the tendency of this decisional behavior of the designer. It checks the properness of the recommendation by taking into account the follow-up design actions. Based on the computational observation if the designer executes, or not, a particular design action the sub-function $F_{1.3}$ identifies the related design entity. This is possible owing to the one-to-one (computational) mapping between the design actions and the design entities, Based on the result of this identification action, the sub-function $F_{1.3}$ can conclude if a proper recommendation was given. This conclusion is shared with the SVA.

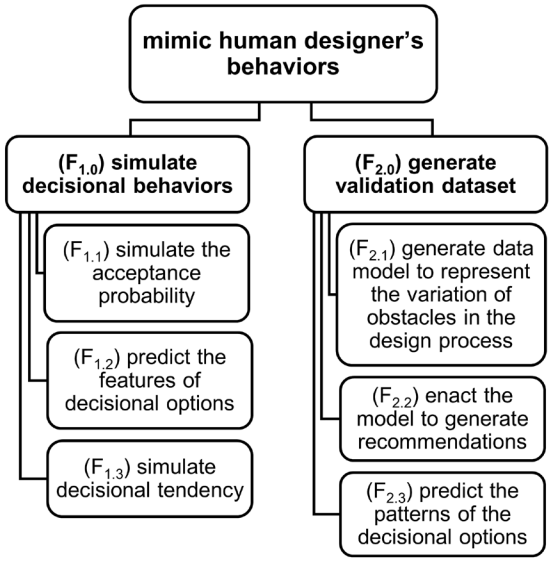


Figure 5.6: Decomposition of the main functions of the SVA to sub-functions

The sub-function $F_{2.1}$ generates the data model to represent a range of obstacles, which may occur at performing certain design actions. In the practical design process, the possibility of the occurrence of an obstacle varies according to several factors, for instance, (i) the experiences of the designer, (ii) the characteristics of the dataset, and (iii) the complexity of the algorithms. Due to the complicatedness of the task of analyzing all of these factors, we left this issue for prospective future research. To by pass this issue, we assumed that the designer has the chance to face an obstacle at any design actions equally well. This assumption was taken into account at the simulation of the data model by the SVA.

The sub-function $F_{2.2}$ enacts the decisional model to generate a recommendation. This sub-function is related to the process-based recommendation mechanism of the ARF. It utilizes the information included in the RPP to: (i) generate candidate process flow models, (ii) to select of design action flow, and (iii) predict the next design actions. The sub-function $F_{2.3}$ predicts the patterns of the decisional options. The logic of the procedure underlying the operation of the $F_{2.3}$ starts with a classification of the recommendations in terms of their appropriateness. The classification combines the appropriateness and/or inappropriateness of each recommendation with the decisional options. The sequence of the decisional options indicates a noticeable decision tendency. The decisional tendency shows a probabilistic nature (i.e., the probability of what the designer actually selects). The expected output of the sub-function $F_{2.3}$ is the probability of the decisional options, as discussed in Section 5.2.2.

5.3.3.3 Specification of the algorithms needed for the smart validation agent

For the realization of the discussed functionality of the SVA, twelve algorithms were needed. They are presented in Table 5.4. In the case of sub-function $F_{1.1}$, in terms of recognizing the meaning of a recommendation, we considered two decisional options. Four algorithms were needed to perform this sub-function. The algorithm A1.01 generates the data model that represents the variations of the common knowledge of the designer is assumed to have. Each instance contains a set of knowledge elements, which characterize an individual designer in the SVA. The input data is the contents of the RPP and the proportion of the common knowledge. The algorithms A1.02 and A1.03 are to calculate the components of the decisional model, including the probability of shared knowledge and the probability of known knowledge elements of the mimicked designer, respectively. The algorithm A1.04 uses the decision model to calculate the probability of the acceptance of the recommendations. The algorithms A1.01 - A1.04 are executed *n* times in a loop to generate the collective behaviors of all mimicked designers.

Belonging to the sub-function $F_{1.2}$, the algorithm A1.05 determines the probability of the four decisional options (features) according to the level of acceptance of the recommendations. For the realization of the sub-function $F_{1.3}$, two algorithms were specified. The algorithm A1.06 identifies the follow-up design entities included in the extended design activity flow graph. The algorithm A1.07 analyses the decisional tendency of the mimicked designer with regard to the properness of the recommendations. The decision tendency is determined based on the following conditions:

Table 5.4: Specification of the algorithms needed for the computational realization

functions	needed algorithms	
F1.0: simulate decisional behaviors of the designer		
F1.1	A1.01:	generate data model to represent the common knowledge of the designer
	A1.02:	calculate the probability of shared knowledge elements
	A1.03:	calculate the probability of known knowledge elements
	A1.04:	calculate the probability of acceptance of the recommendations
F1.2	A1.05:	predict the features of decision behaviors
F1.3	A1.06:	identify the follow up design entities
	A1.07:	simulate decisional tendency concerning the design process
F2.0: generate a validation dataset		
F2.1	A2.01:	generate data model to represent variation of obstacles in the design process
	A2.02:	generate the recommendations
F2.2	A2.03:	generate data model to represent the variation of the appropriateness of recommendations
	A2.04:	simulate the patterns of decisional behaviors
	A2.05:	consolidate the data features

$$c_i = \begin{cases} 1, & n(E_x \cap E_c) \geq 1 \\ 0, & otherwise \end{cases} \quad (5\text{-}9)$$

where: c_i is the decision tendency of the designer, $c_i \leftarrow \{0,1\}$, where (1) means a proper recommendation, and (0) means an improper recommendation E_x is a finite, non-zero set of design entities, which are included in the extended design activity flow, and E_c is a finite, non-zero set of design actions related to the design process, which are supposed to be known by the designer.

The algorithms related to the sub-function $F_{2.1}$ are employed to enact the computational action of recommendation generation. The algorithm A2.01 simulates the process of identifying the obstacle in the design process. What simulation means here is that, based on the data model, the algorithm selects an entity related to which an obstacle probably occurred in the process, and generates a case-related recommendation. The constructed data model represents the variation of the obstacles that probably occur in the design process. The expected output is a set of possible design entities related to which obstacles occur. The algorithm A2.02 is a combination of two ARF algorithms, namely, Algorithm A4.03 (for selecting the candidate PFMs), and Algorithm A4.06 (for predicting the next design action). The combination of these algorithms is dedicated to the identification of the candidate PFMs and to the construction of the design activity flow.

To realize the sub-function $F_{2.1}$, three interrelated algorithms were required. The algorithm A2.03 constructs the data model to represent the variation of the appropriateness of the recommendations. The probability of the decisional options is determined by the Algorithm A2.04. The algorithm A2.05 consolidates all necessary data features. The expected output is the validation dataset that includes the values of (i) all independent variables, (ii) the variables related to the knowledge elements of the designer, (iii) the acceptance probability of the recommendations, (iv) the probability of the decision tendency of the designer, and (v) the probability of the decisional options.

5.3.4 Testing the synthetic validation agent

The objectives of the functionality testing of the SVA were to check if (i) the related algorithms produce the outputs according to the expectations, and (ii) the specified functional requirements are fulfilled. Four sub-functions were tested, as discussed individually below.

5.3.4.1 Testing of the sub-function $F_{1.1}$

In line with the assumption concerning the construction of the decisional model, three situations related to the knowledge possession of the mimicked designer were considered. For the sub-function $F_{1.1}$, the expectation was that the non-linear relationships of the acceptance probability of the recommendations and the proportion of the common knowledge should be reflected by the decision model. To check the fulfilment of this, three scenarios were tested, in which different number of design entities, 50, 100, and 200, were included in the RPP.

From c_k = 0.05 to c_k = 0.25, the probability of the acceptance decreased dramatically when the proportion of common knowledge increased. This trend was noticeable in the decision mode I. From c_k = 0.25 to c_k = 0.75, the probability of acceptance was slightly decreased and reached the lowest value at c_k = 0.5. After that, the trend of the probability of acceptance changed to show a positive relationship with the proportion of common knowledge. This pattern was noticeable in the decision mode II. From c_k = 0.75 to c_k = 0.95, the acceptance probability steadily increased as the proportion of common knowledge of the designer was supposed to increase. This pattern was noticeable in the decision mode III. As shown in Figure 5.7, the results of this computational experiment indicated that the computed patterns of the acceptance levels were consistent with the expectations in the case of each scenario,

5.3.4.2 Testing of the sub-function $F_{1.2}$

The basic requirement for the sub-function $F_{1.2}$ was an aggregated probability of the acceptance of the appropriate or the rejection of the inappropriate recommendations. Three independent variables were used in the testing scenarios to determine the probability of the features of the decision options: (i) the total number of entities included in the RPP, (ii) the accuracy rate of the recommendation generation, and (iii) the proportion of common knowledge. In the test, the first two variables were regarded as the constant parameters and,

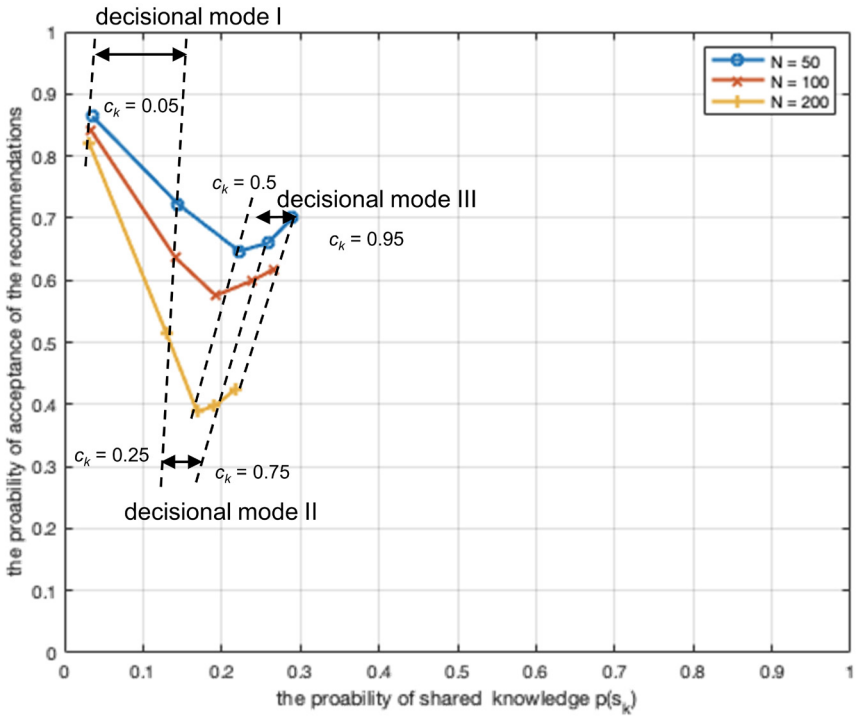


Figure 5.7: The patterns of the acceptance probability of the recommendations according to the probability of shared knowledge

as such, were set to 100 entities and 75 % of the accuracy rate, respectively. We assumed that the accuracy rate represented the probability of the appropriate recommendations. The third variable was for the set of discrete values of 0.05, 0.25, 0.5, 0.75 and 0.95, respectively, representing the five levels of common knowledge.

The results shown in Figure 5.8 indicate that the different patterns (features) of the decision options occurred according to their aggregated probabilities. In line with the decision model I, the SVA (mimicking a particular type of designers) intended to accept all recommendations. In the case of 75 % accuracy rate, the proportion of accepted-appropriate and accepted-inappropriate decisions was 0.75: 0.25, respectively. In the domain from c_k = 0.25 to c_k = 0.75, the acceptance probabilities of the recommendations were found to be according to the decision mode II. At c_k = 0.95, the acceptance probabilities were found to be according to the decision mode III.

The numerical results showed the aggregated probability of the four features (accepted-appropriateness, rejected-appropriateness, accepted-inappropriateness, and rejected-inappropriateness) of the decision options. On the approval side of the recommendation acceptance, the proportions of accepted-appropriate and accepted-inappropriate decisions

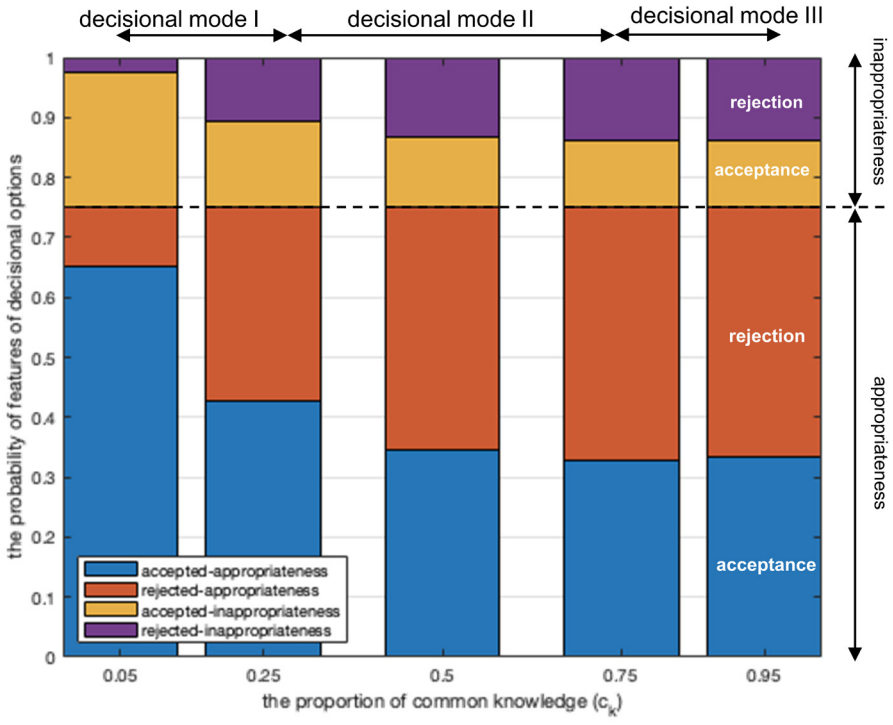


Figure 5.8: The patterns of the features of decisional options according to the levels of the common knowledge

were 0.75: 0.25, respectively, for all scenarios. This was similar to the disproval side of the recommendation acceptance, where the proportion of the aggregated probabilities of the rejected-appropriate and the rejected-inappropriate decisions were 0.75 : 0.25, respectively, for all scenarios. This allowed us to conclude that the requirement posed for $F_{1.2}$ was fulfilled.

5.3.4.3 Testing of the sub-function $F_{1.3}$

The expected output of the sub-function $F_{1.3}$ was the aggregated probability of selecting the proper/improper recommendations by the SVA. To determine the probability of the properness of the recommendations, three independent variables were taken into a consideration. They were (i) the total number of entities included in the RPP, (ii) the proportion of common knowledge, and (iii) the extended design activity flow. In the test, the extended design activity flow was chosen based on the highest values of the joint probability distribution of the process flow models, which included three entities of the case-related recommendation and n number of succeeding design entities.

As in the previous test, the proportion of the assumed common knowledge was set to the five levels (i.e., 0.05, 0.25, 0.5, 0.75, and 0.95) at the investigation of the correlations with the aggregated probability of the proper recommendations: the knowledge elements of the agents were generated randomly by varying the proportion of common knowledge. Three scenarios were tested, in which different number of design entities were included in the RPP, namely: 50, 100, and 200. The simulation was run for generating 100 recommendations for each scenario.

The obtained results showed the aggregated probability of the combined decisional tendency of the SVA for the different proportion levels of the common knowledge. They are presented in Figure 5.9 for each of the three scenarios. We noticed that the proportion of common knowledge and the probability of properness of the recommendations had a strong positive relationship. This was 'signposted' by the linear trends for all scenarios. We confirmed that the algorithms for the realization of $F_{1.3}$ produced the output as per expectation.

5.3.4.4 Testing of the sub-function $F_{2.2}$

The sub-function $F_{2.2}$ was about predicting the probability of the four decisional options concerning the recommendations. To realize the sub-function $F_{2.2}$, that is to determine the pattern of decision options, the algorithms needed to combine all data that were required for the characterization of the decisional behaviors. This included the data concerning (i) the recognition of the appropriateness of recommendations, (ii) the features of the decision options, and (iii) the properness of the recommendations). Like before, five scenarios were defined according to the assumed proportions of the common knowledge (0.05, 0.25, 0.5, 0.75 and 0.95, respectively) in the computational testing.

In the case of c_k = 0.05, the outcomes were according to the decision model I. All recommendations were accepted, but the designer mimicked by the SVA has insufficient

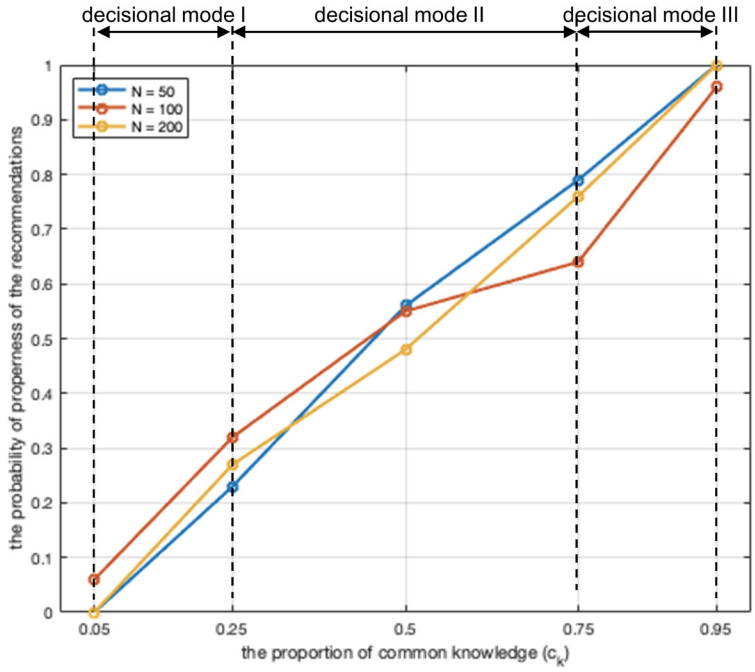


Figure 5.9: The correlations of the aggregated probabilities of the combined tendency of the SVA and the proportion of common knowledge

knowledge to evaluate the appropriateness of the recommendations. Thus, the pattern of the decisional options showed a high proportion of unjustified subjective decisions. In the case of c_k = 0.25 to c_k = 0.75, the patterns of decisional options were generated according to the decision mode II. In the case of c_k = 0.95, the pattern followed the decision mode III. Figure 5.10 shows the results obtained for the probabilities of the decision options for each of the five proportion levels of the common knowledge. These results confirmed that the requirement for the sub-function $F_{2.2}$ was fulfilled.

5.3.5 Deriving indicator for usefulness

As explained at the beginning of this chapter, the objective of the validation study was to provide (preferable quantifiable) metrics to evaluate the usefulness of the recommendations. However, it could not be captured by an exact quantitative variable and function. Therefore, we needed to compose a quantitative indicator. The decisional options were shown in Table 5.1. Their combinations define various decisional situations, which may vary in the design process in terms of judging the appropriateness of the recommendations and choosing the proper follow up design actions. This latter was interpreted as decisional patterns. The goal of the computational simulation was to produce various decisional patterns according to the assumed decisional behavior of the designer. The decisional pattern is the basis of deriving indicator of usefulness of the recommendations.

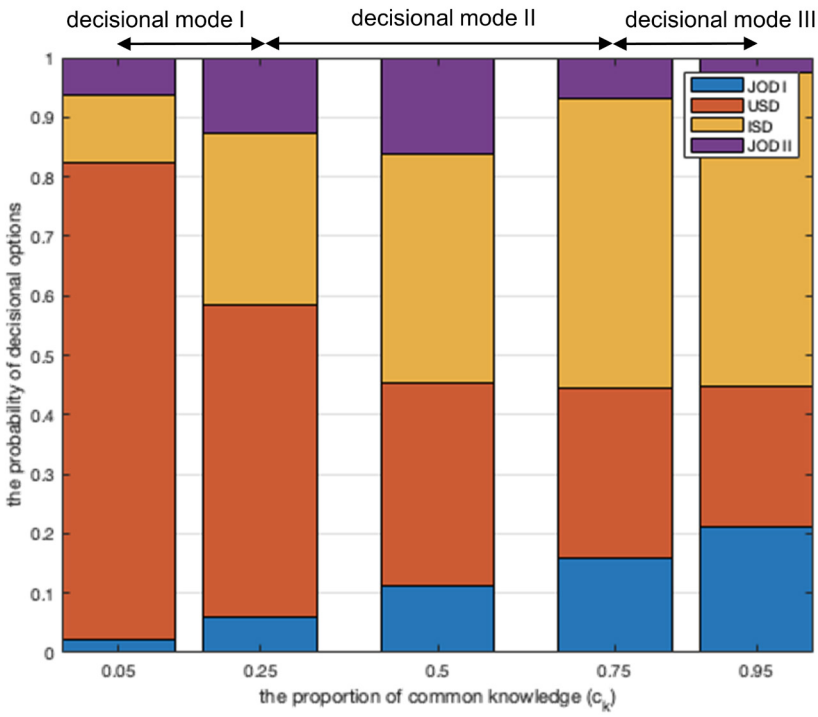


Figure 5.10: The patterns of the decisional options according to the proportion of the common knowledge.

The method of prognostic reasoning was applied to construct a usefulness indicator concerning the recommendation based on the decisional patterns. The term 'prognosis' means prediction of a final outcome based on preliminary data and calculations [5]. In general, prognostic reasoning is a forward-looking process to predict a most likely situation based on the current symptoms or indicators [6]. As a first attempt, we defined the usefulness indicator as a function of the decisional mode:

$$U_i = f(\Delta_i) \tag{5-10}$$

where: U_i is the evaluated indicator of usefulness, which involves two decisional options: (i) type I justified objective decision (JOD_I), and (ii) unjustified subjective decision (USD). These two decisional options logically imply that the designer can continue the design process. According to the discussed four decisional options, the value of the totaled probabilities is equal to 1. Hence, the range of variation of the probability value of the indicator of usefulness is between 0 and 1. Symbolically:

$$U_i = p(u_1) + p(u_2) \tag{5-11}$$

where: u_1 is type I justified objective decision, and u_2 is unjustified subjective decision.

5.4 Execution of the usefulness validation of procedural recommendations

5.4.1 Identification of the validation scenarios

In order to evaluate the usefulness of the recommendations, we investigated the correlations of the possessed knowledge of the SVA and its decisional behavior with a view to the recommendations. To realize prognostic reasoning, the proportion of the common knowledge possessed by the SVA is used to predict the usefulness of the recommendation. The four parameters that established varying validation scenarios were: (i) the total number of design entities (N), (ii) the relationships of the design entities (L), (iii) the performance of the recommendation generation mechanism (a_r), and (iv) the proportion of common knowledge (c_k). The values of the first two parameters were derived based on the RPP. To represent the performance of the recommendation generation, the accuracy rate was used for the third parameter. The appropriate recommendation was identified based on the assumed normal distribution and the statistic measure related to the accuracy rate. The fourth parameter represents the knowledge possession of the designer.

Two validation scenarios were set up with a constant number of 50 and 100 knowledge elements, respectively, for carrying out validation data generation by the SVA. All of the three decisional modes were tested based on these two scenarios. The total number of entities included in the RPP varied in the particular decisional modes. For all scenarios, the accuracy rate was regarded as the controlled variable. The expected accuracy rate was set at 75% in the test. The parameter values for the validation scenarios are presented in Table 5.5.

5.4.2 Operationalization of the synthetic validation agent

Figure 5.11 shows the computational workflow of operationalization of the SVA. Within the workflow, three procedural parts were specified. The procedure I was related to the execution of the algorithms for the sub-functions $F_{1.1}$ and $F_{1.2}$. This procedure included

Table 5.5: Validation scenarios

parameters	**scenario I**			**scenario II**		
maximum number of knowledge elements known by the designer (n_{max})	50			100		
the decisional mode	Δ_1	Δ_2	Δ_3	Δ_1	Δ_2	Δ_3
the proportion of common knowledge (c_k)	0.15	0.5	0.85	0.15	0.5	0.85
the total number of design entities as the elements of the RPP (N)	333	100	59	667	200	118
the total number of connections of the entities in the RPP (L)	13,489	1,246	413	55,209	4,938	1,822

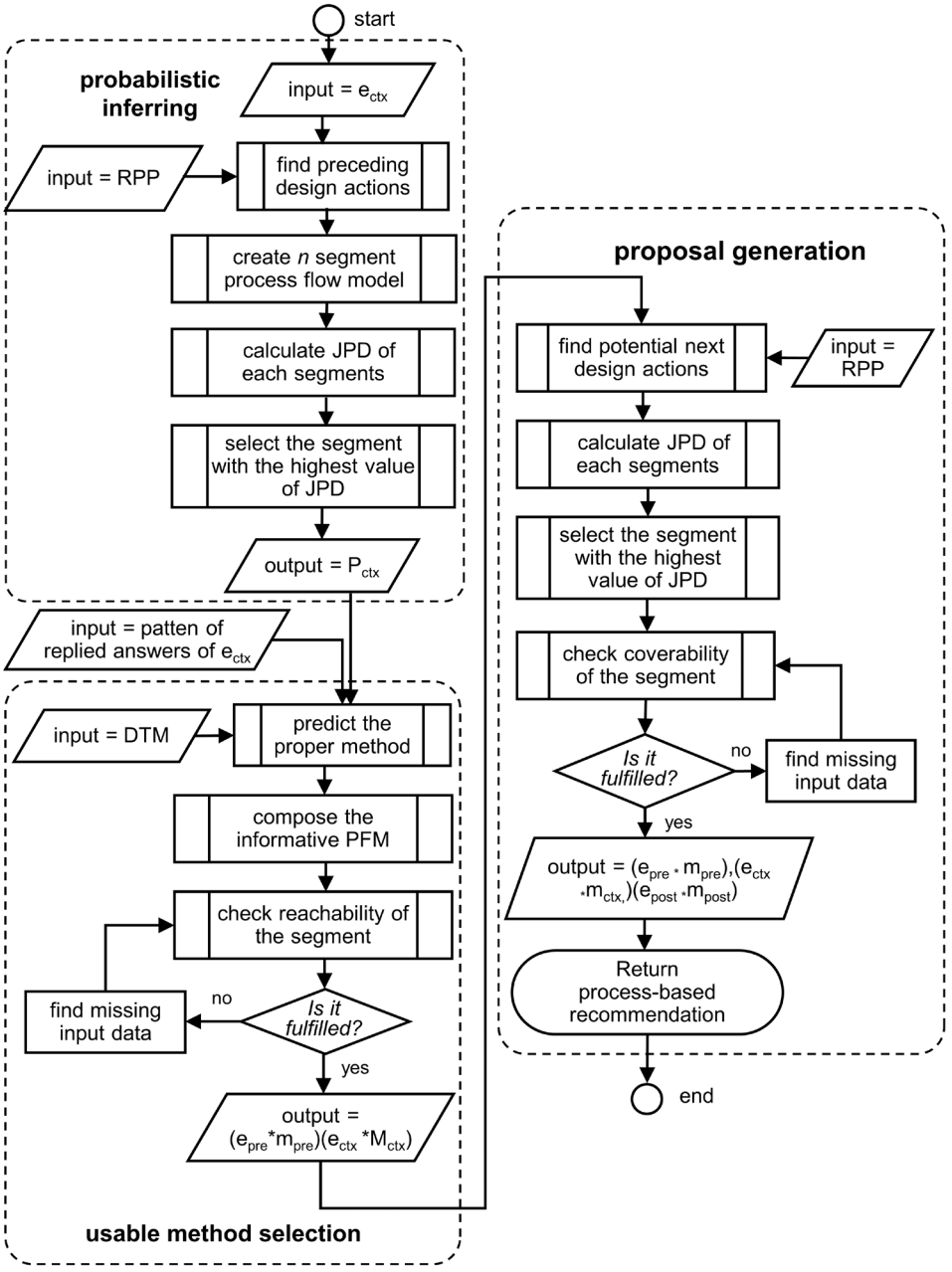


Figure 5.11: The operationalization of the SVA

eleven steps to determine the acceptance probability of the recommendations. The procedure II was related to the execution of the algorithms for the sub-function $F_{1.3}$. The expected output of this procedure was the set of decision tendencies concerning the properness of the recommendations. The procedure III was related to the aggregation of the two sets of data related to the decisional behaviors of the SVA and to determining the patterns of decision options. The expected output of the overall computational workflow was the (so called) validation dataset.

Workflow procedure I:
Prediction of the probability of the features of the decisional options of the SVA

Input data: (i) the reference protocol (RPP), and (ii) the given proportion of common knowledge

Output data: (i) data features containing the independent variables, and (ii) the probability of the acceptance of the recommendations

Step 1: simulate using the data model to identify the obstacles in the design process

Step 2: identify the particular design entity related to which the obstacle probably occurred

Step 3: construct the candidate process flow models that include the concerned design entity

Step 4: construct a design activity flow as the representative of the most informative PFM

Step 5: simulate the knowledge elements of the designer mimicked by the SVA

Step 6: identify the design entities which represent the knowledge elements of the designer

Step 7: calculate the probability of the shared knowledge

Step 8: calculate the probability of the known knowledge elements of the designer mimicked by the SVA

Step 9: calculate the probability of acceptance of the recommendations

Step 10: consolidate the data features that includes n number of instances

Step 11: calculate the probability of the features of decision options

Workflow procedure II:
Simulation of the decision tendency of the SVA

Input data: (i) the process-based recommendations, and (ii) the elements of the common knowledge possessed by the designer who is mimicked by the SVA

Output data: (i) data features containing the independent variables, and (ii) the set of decision tendencies of the designer mimicked by the SVA

Step 1: construct n number of the extended design activity flow

Step 2: identify the follow-up design entities

Step 3: simulate the decision tendency of the designer according to the knowledge elements of the designer and the follow-up design entities

Step 4: consolidate the data features that includes n number of decisions

Workflow procedure III:
Prediction of the patterns of the decisional options of the SVA

Input data: (i) the process-based recommendations, (ii) a given accuracy rate of the process of recommendation generation, (iii) the data features from the workflow procedure I, and (iv) the data features from the workflow procedure II

Output data: a validation dataset

Step 1: simulate the data model to identify the appropriateness of the recommendations

Step 2: evaluate the appropriateness of the recommendation

Step 3: simulate the patterns of the decisional options

Step 4: consolidate the data features for constructing the validation dataset

5.4.3 Identification of the obstacle in the design process

In the practical design process, a non-usual event (NUE) occurs, when the ARF detects the irregular pattern of designer's behavior following his/her procedural decision (concerning the next design action and the related design method). This situation is assumed as the first step of the simulation of the recommendation generation. It starts with the identification of the obstacle which possibly appeared in the design process. This section describes the behavior of the designer when he/she experiences an obstacle. Table 5.6 shows the list of possible reactions that a designer will behave to response the obstacle at any design actions.

The lack of information (information deficiency or incompleteness) or a wrong assumption may lead to wrong decision of the designer, which may in turn create an NUE. This latter appears as an obstacle in the design process. Two accompanying questions arose: (i) what may cause information deficiency or incompleteness with regard to a design action, and (ii) what may lead to the formulation of a wrong assumption concerning a design action? These are fundamental questions since useful recommendations should eliminate the lack of information and should prevent a wrong assumption.

5.4.4 Generation of case-related recommendations

Having identified an obstacle, the concerned module of the ARF generated a procedural recommendation and suggested a proper method by considering (processing) the relevant segment of the process flow model. The generated recommendation was intended to guide the selection of the next design action that would probably lead to a sufficing result. Computationally, the formulation of the recommendation relied on the related segment of the process flow model, which was captured in the reference protocol $\mathbb{P}_{ctx} := \{e_{pre}, e_{ctx}, e_{post}\}$

Table 5.6: Reactions of a designer leading to an expected event

sub-tasks	potential design actions	information deficiency or incompleteness	formulation of wrong assumption
1.0	cleansing dimensionality reduction blending split data	Unable to find incompleteness in the dataset Unable to recognize irrelevant data instances Unable to understand the characteristics of the required dataset	Removing important features from the dataset Selecting mismatching method to pre-process the dataset
2.0	extract features analyze the dataset components select attributes select a response	Unable to recognize irrelevant features in a dataset Unable to determine the correlation of the features	Selecting incorrect features for training a model Identifying incorrect type of predictions
3.0	identify an objective of a model selecting a training algorithm fitting a model modify the training set select the model export the model	Unable to recognize deficiency or incompleteness of the training set Unable to recognize types of predictive features and response Unable to define the objective of the trained model	Defining irrelevant objective to train the model Selecting mismatching algorithm to train the model Over fitting the model Under-fitting the model
4.0	identify the criteria to select the evaluation metrics select the metrics scoring a model modify the metrics optimize the parameters	Misunderstanding the goal of the trained model Lack of knowledge about the metrics Unable to define the goodness of the model	Selecting mismatching metrics for the considered training algorithm Selecting irrelevant metrics to evaluate the model Applying improper metrics to evaluate the model
5.0	analyze the metric modify the metrics optimize the parameters	Unable to recognize the deficiency and incompleteness of the sample set Unable to analyze the results Unable to semantically interpret the results	Select multiple methods for scoring the predictive model Apply the improper method for scoring the model
6.0	evaluate performances	Unable to identify the numerical value to justify the goodness of the predictive model Unable to decide the goodness of the model	Misinterpreting the evaluation metrics Selecting inefficient model due to misinterpretation of evaluation metrics

and the set of best methods that could be used to execute the concerned design action, $\hat{m}_{pre}$, $\hat{m}_{ctx}, \hat{m}_{post} \in \hat{\mathcal{M}}$.

In the validation test, the generation of recommendation was simulated by means of the Bayesian network embedded in the RPP. The highest value of the joint probability distribution (JPD) of the candidate process flow models was selected as a case-related recommendation. What the term 'case-related' means in this context is that the recommendation was generated based on historically-used cases of the concerned design process. The selection of the best method was assumed to be done without getting any further information from the SVA. Figure 5.12 shows the simulation results of the case-related recommendation generation. According to the fundamental concept of deriving the decisional model for the SVA, the circular graph representing the RPP. In the figure, the dots represent the design entities and the dashed lines represent the relationships among the entities.

In the simulation, the total number of the entities was set to 50 elements. Altogether, a total number of 308 links were included in the RPP. In the simulation the design entity e_{13} was identified as the action where the obstacle was appeared. It is shown by the blue dot in Figure 5.12. The simulated recommendation involved three entities, e_{31}, e_{33}, and e_{38}. They

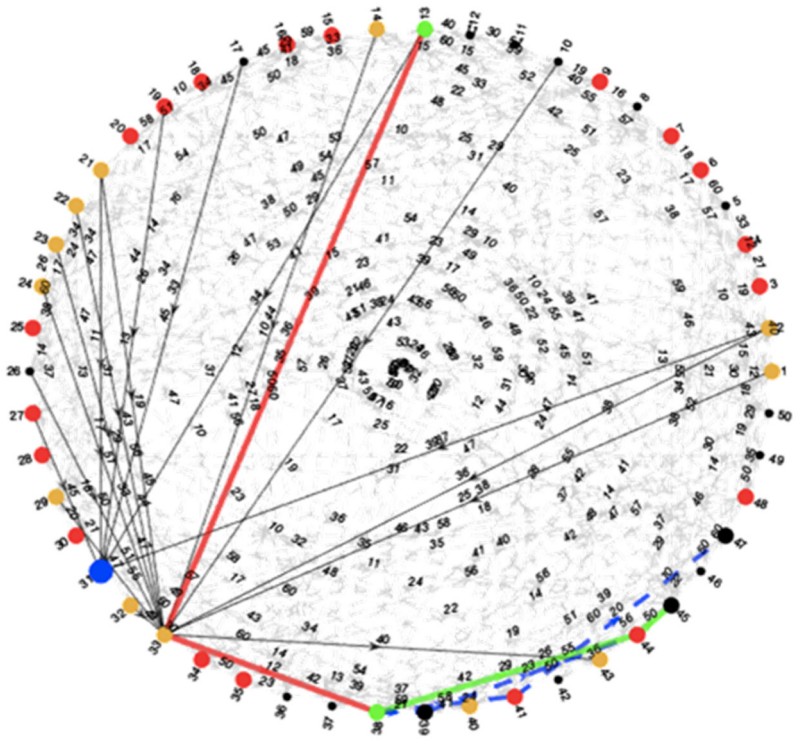


Figure 5.12: Graphical representation of the case-related recommendation generation using the RPP

are connected by the solid red line in Figure 5.12. The yellow dots represent the concerned segment of the process flow models (E_p), and the red dots represent the common knowledge elements (E_c) shared by the designer and the RPP.

We set the proportion of common knowledge to 0.5. Thus, there were 25 common knowledge elements. The green dots represent the knowledge elements shared in the recommendation by the designer. The extended design activity flow included two design entities $e_{44}, e_{45} \in E_x$ which are connected by the green solid line. The entities that connected by the dash blue lines are the alternative design activity flows. Based on the numerical value concerning the above explanation, the probability of the decision options was determined. These are shown in Table 5.7.

Table 5.7: Determination of the probability of the decision options according to the simulated case-related recommendation

<table>
<tr><th colspan="2" rowspan="4">aspects of evaluation</th><th colspan="4">recommendation provided by the ARF</th></tr>
<tr><td colspan="2">appropriate recommendation
(0.75)</td><td colspan="2">inappropriate recommendation
(0.25)</td></tr>
<tr><td colspan="2">acceptance by the designer</td><td colspan="2">acceptance by the designer</td></tr>
<tr><td>accepted
0.7977</td><td>rejected
0.2033</td><td>accepted
0.7977</td><td>rejected
0.2033</td></tr>
<tr><td rowspan="2">decision on design action</td><td>proper design action selected
= 2/19</td><td>justified objective decision I
0.1597</td><td colspan="3">unjustified subjective decision
0.3861</td></tr>
<tr><td>improper design action selected
= 4/19</td><td colspan="2">incorrect subjective decision I
0.4273</td><td>justified objective decision II
0.027</td></tr>
</table>

5.5 Evaluation of the usefulness of recommendations

5.5.1 Descriptive statistical analysis of the data for validation

The validation datasets were generated by using the SVA for simulation. Two scenarios were considered based on a constant number (50 and 100 elements, respectively) of the knowledge elements of the SVA. We aimed at analyzing the decisional behaviors of the simulated agents from two viewpoints: (i) comparing the particular decisional mode in the different scenarios, and (ii) comparing the different decisional mode in a particular scenario. Two dependent variables were considered: (i) the total number of instances of decision options – which represented the properness of recommendations, and (ii) the acceptance probability of the recommendations $p(aR)$.

Table 5.8 shows the descriptive statistics of the dataset according to the validation scenario I. The instances of the type I justified objective decision (JOD_I) and the unjustified subjective decision (USD) represented the total number of proper recommendations. Meanwhile, the instances of the incorrect subjective decision (ISD) and the type II justified objective decision (JOD_II) represented the total number of improper recommendations. In view to Δ_1, the total number of improper recommendations was bigger than the total number of proper recommendations.

Table 5.8: Descriptive statistical data of the validation scenario I – 'n_{max} = 50'

	decisional mode I			**decisional mode II**			**decisional mode III**		
	n(RPP) = 333& c_k = 0.15			n (RPP) = 100& c_k = 0.5			n (RPP) = 59& c_k = 0.85		
	instances	*p(aR)*		instances	*p(aR)*		instances	*p(aR)*	
		mean	std		mean	std		mean	std
JOD_I	11	0.079	0.133	42	0.266	0.053	66	0.483	0.066
USD	13	0.468	0.127	66	0.516	0.06	106	0.457	0.08
ISD	145	0.544	0.175	79	0.494	0.072	23	0.501	0.08
JOD_II	21	0.363	0.108	13	0.28	0.042	5	0.039	0.099

In contrast with the decision mode III, the total number of proper recommendations was much larger than the total number of improper recommendations.While the proportion of the total number of proper recommendations and the improper recommendations was also the same. This pattern was similar to the data of the validation scenario II, as shown in Table 5.9. Thus, we concluded that, in comparison, a decision mode which had the bigger proportion of common knowledge elements yielded in the higher number of proper recommendations.

Table 5.9: Descriptive statistical data of the validation scenario I – 'n_{max} = 100'

	decisional mode I			**decisional mode II**			**decisional mode III**		
	n(RPP) = 333& c_k = 0.15			n (RPP) = 100& c_k = 0.5			n (RPP) = 59& c_k = 0.85		
	instances	*p(aR)*		instances	*p(aR)*		instances	*p(aR)*	
		mean	std		mean	std		mean	std
JOD_I	15	0.092	0.085	35	0.168	0.035	63	0.353	0.507
USD	21	0.529	0.089	59	0.583	0.116	105	0.548	0.088
ISD	129	0.566	0.202	93	0.482	0.141	28	0.489	0.104
JOD_II	19	0.359	0.11	13	0.326	0.05	4	0.093	0.064

Before normalization, we analyzed the probability of the recommendation acceptance by considering the simulated data of the average value of acceptance probability for all decisional options. Therefore, the total sum of the probabilities within the same decisional mode was greater than one. Different decisional modes showed different patterns of decision options. In the case of Δ_1, the acceptance probability of JOD_I was very small compared to the other options. In the case of Δ_2, the smallest value of the acceptance probability of JOD_II was found in comparison with the other decisional options.

For all decision modes, the acceptance probability of the USD and the ISD were derived for the same proportions (by $p(aR_i) = 0.5$, approximately). This pattern seemed to be similar to the statistical data obtained in the case of the scenario II. It confirmed that the designers simulated by the SVA tended to accept the appropriate recommendations if the proportion of common knowledge elements was comparatively bigger. The total number of knowledge elements in the RPP seemed to have not influence on the acceptance of the recommendations with regard to the same decisional mode.

5.5.2 Investigation of correlations between the considered variables and the decision options

The aim of the work described in this section was to investigate the correlations between the considered variables (i.e., the decision modes, Δ_i and the total number of design entities included in the RPP) and the decision options, in particular the JOD_I and the USD. The indicator of usefulness relies on these two interrelated features. In our investigation, the indicators of usefulness were considered in two validation scenarios. The results are shown in Figure 5.13. In this figure, the black dashed line represents the indicator of usefulness of the recommendation in the case of the validation scenario I, and the blue solid line represents the indicator of usefulness in the case of the validation scenario II. The major findings are as follows:

Findings concerning the comparison of the particular decisional modes between different scenarios

- By comparing the results yielded by the particular decisional modes in the two validation scenarios, we found that, in the case of Δ_2 and Δ_3, the usefulness of the recommendations was higher when they were generated by a lower number of design entities in the RPP. At the same time, in the case of Δ_1, the indicators were found to be not significantly different in the two scenarios.
- A lower number of knowledge elements gave a higher probability value of usefulness in the case of Δ_2 and Δ_3. However, in the case of Δ_1, it seemed to be not significantly different in the two scenarios (n(RPP) = 333, and n(RPP) = 667). We also found that when the RPP contained a large number of design entities and the number of knowledge elements captured by the SVA increased, the usefulness of the recommendation was increased only with a small probability value.

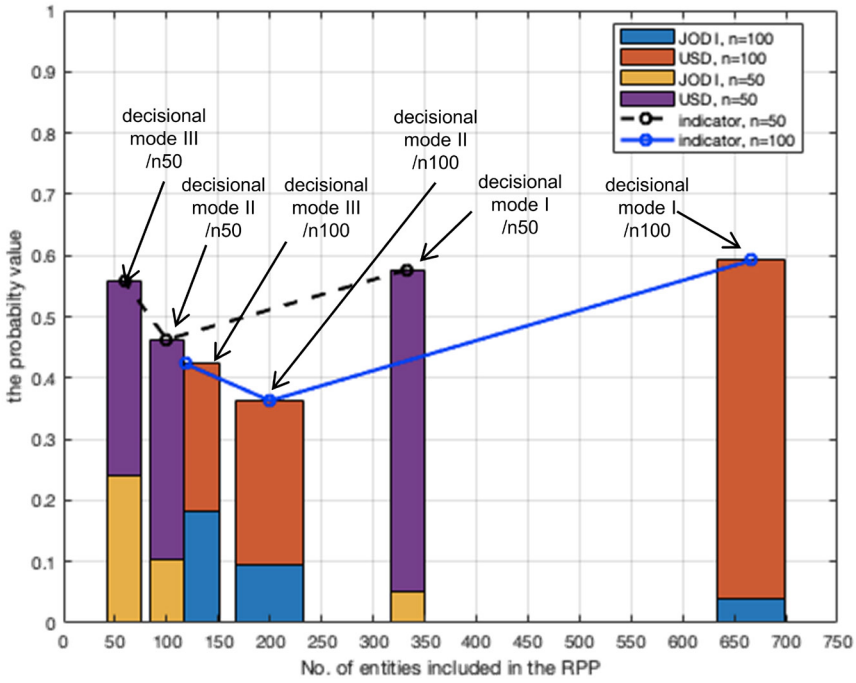


Figure 5.13: The correlations of the common knowledge elements and the probability of the usefulness indicator of recommendations

Findings concerning the comparison of the different decisional modes in the same scenario

- By comparing the results concerning the different decisional modes in the same scenario, we found that Δ_1 and Δ_3 yielded in a higher probability of usefulness than Δ_2. The patterns of the decision options followed what was assumed for the decisional model in the case of both tested scenarios.
- Having a constant number of knowledge elements in the SVA, whilst the proportion of common knowledge elements decreased due to the increased number of the design entities in the RPP, the usefulness slightly decreased in the transition from Δ_3 to the Δ_2 and then it increased in the transition from Δ_2 to Δ_1. This pattern confirmed that the dataset generated by the SVA was valid and sufficient for the validation study.
- Concerning the features of the usefulness indicator, the following can be stated. The probability value of the JOD_I was in a negative correlation with the total number of design entities included in the RPP. If the common knowledge elements presented by the SVA were fixed to be a given constant number and the total number of design entities included in the RPP increased, the probability of the JOD_I decreased. At the same time, the proportion of the USD increased as the total number of design entities included in the RPP increased.

- Furthermore, no matter if the offered recommendation was useful or not, there was a high probability that the SVA either accepted an inappropriate recommendation or rejected an appropriate recommendation. The explanation is that it did the acceptance or rejection without being aware of the appropriateness of the recommendation.

5.5.3 Opportunities for improving the recommendation generation process

Finding opportunities for improving the recommendation generation process is a vital issue for the ARF as a whole. From a knowledge engineering point of view, the improvement of the recommendation generation can be seen as a knowledge management issue. More concretely, the pursued increase in the usefulness of recommendations can be achieved by improving the knowledge management related to the RPP. This has legacy since the RPP is a computational process model, which contains knowledge about design entities and their relationships. As was argued earlier, the proposed RPP is flexible enough to represent even complicated (multi-trajectory) design processes. Based on the variable relationships of the incorporated design entities, multiple design activity flows can be captured in the RPP. This affordance was used as the basis of the recommendation generation in this promotion research. When it comes to the evaluation of the usefulness of the recommendations, this affordance can also be utilized. When improvement opportunities for the recommendation generation are sought for, two variables (i) the decisional modes, and (ii) the indicators of the usefulness can be taken into consideration. The former variable has direct influence on the actual decision of the SVA, and the latter can be used for the evaluation of the recommendation generated using the RPP. These will be discussed below in detail.

5.5.3.1 Improvement opportunities concerning the decisional modes

The concept of decisional modes was introduced to describe the proportion of common knowledge possession of the designer in the case of a particular design process. Three types of the designers were taken into account based on how competent they are to execute the design process. The ARF is able to diagnose computationally what the designer knows about a particular design action when an NUE has been detected. However, it cannot evaluate how much knowledge an individual designer possesses. The fact of the matter is that the knowledge possession of a designer is an uncontrollable variable. On the other hand, it strongly influences the acceptance of recommendations. Hence, when looking for improvement opportunities, a key issue is how to determine the optimal proportion of the common knowledge that is shared by the mimicked designer and the RPP. We argued that this information can be used to enhance the usefulness of recommendations.

Regarding the decisional model of the SVA, the decisional modes have direct relations with the acceptance probability of the recommendation. A higher probability of acceptance offers a higher possibility of having a useful recommendation. Taking this into consideration, the following opportunities seem to be possible for improving the quality of recommendation generation:

- In the case of Δ_1, although it yielded the highest value for the probability of the usefulness indicator, the probability of JOD_I was quite low (less than 0.1 as shown for both scenarios by the columns of decisional mode Δ_1 in Figure 5.13). The SVA was equipped with constant number of knowledge elements of (n_d = 50 and 100), while the RPP contained a large number of design entities and their relationships. These conditions implied that the RPP could represent a rather complicated design process. However, to enhance the probability of acceptance, it should also offer a practical and executable procedure as part of the recommendation.
- In the case of Δ_2, the lowest probability of the usefulness indicator was obtained in comparison with the other decisional modes (as shown for scenario II - n_d = 100 - in the column of Δ_2 in Figure 5.13). An increase in the number of design entities included in the RPP decreases the proportion of common knowledge possessed by the SVA. This adjustment implies a replacement of the SVA from the Δ_2 to Δ_1 decisional mode. As a result, the acceptance probability of recommendation is increased. However, the transition from the Δ_2 to Δ_1 decisional mode also increases the computational complexity of the RPP.
- In the case of Δ_2, the highest probability value of the JOD_I was obtained (as shown for scenario I - n_d = 50 - in the column of decisional mode Δ_3 in Figure 5.13). The reason was that the RPP was constructed for a less complicated design process and that the decisional mode Δ_3 represented a high proportion of common knowledge of the SVA (c_k = 0.85). As a consequence, the SVA was able to recognize the appropriate recommendations with a higher chance. Therefore, it became evidenced that the probability of recognition of the recommendations is a key factor of enhancing the acceptance probability of the recommendation.

Considering all different decisional modes together, we could conclude that there were three possible ways of enhancing the acceptance probability: (i) offering practical and executable recommendations, (ii) offering more recommendations options, and (iii) increasing the awareness of the appropriateness of the recommendations.

5.5.3.2 Improvement opportunities concerning the usefulness indicator

Concerning the usefulness indicator, it should be expected that the higher probability of the indicator is better. It indicates the higher possibility that the designer can continue the design process. However, if we consider two features of the indicator, it should be expected that the higher probability of JOD_I is better. On the other hand, the lower probability of USD should be better. These conditions ensure that the designer is able to recognize the appropriateness of the offered recommendation. If considering all expectations concerning the usefulness indicator, the preference conditions for the usefulness of the recommendation should be (i) the highest value of the probability of the indicator, (ii) the highest value of the probability of the JOD_I, and (iii) the lowest value of the probability of the USD.

Simulations were conducted to quantify the indicators of recommendation usefulness. The simulated results are shown in Table 5.10. It was found that no indicator satisfied the

expectations in all decisional modes. On the other hand, there were conflicts found between the expected values of the indicator and its features (see the underlined numbers in Table 5.10). Taking these constraints into consideration, there were three possible options for improving the quality of the recommendation generation.

(i) If the JOD_I is regarded as the major indicator of recommendation usefulness, then the expected probability value of the indicator should be approximately equal to, or greater than 0.24. The prognostic reasoning casted light on the fact that the RPP should be constructed so as to serve the incompetent designers in decisional mode Δ_3.

(ii) If the USD is regarded as the major indicator of recommendation usefulness, then the expected probability value of the USD should be approximately equal to, or less than 0.25, The conclusion is that the RPP should be constructed so as to serve the competent designers in decisional mode Δ_3.

(iii) If the overall usefulness indicator is regarded as the primary indicator of recommendation usefulness, then the expected value of its probability should be greater than 0.5. This implies that the RPP should be constructed so as to serve the incompetent designers in decisional mode Δ_1.

Table 5.10: The simulated results concerning the usefulness indicators

n_d	Δ_1 (ck = 0.15)			Δ_2 (ck = 0.5)			Δ_3 (ck = 0.85)		
	JOD_I	USD	IND	JOD_I	USD	IND	JOD_I	USD	IND
50	0.052	0.525	***0.577***	0.102	0.36	**0.462**	*0.24*	0.316	**0.556**
100	0.038	0.554	**0.592**	0.094	0.269	**0.363**	0.181	*0.241*	**0.422**

5.6 Discussion and interpretation of the findings

5.6.1 Assessment of the validation methodology

The methodology validation was a rather complicated design since several aspects had to be considered including (i) the underpinning theory, (ii) the procedure of validation, (iii) the use of methods and techniques, (iv) the use of instrumentation, and (v) the criteria for goodness (e.g., logical correctness of the validation, the reliability of validation process, and the consistency of findings). The underpinning theory of the validation study was actually not a formal theory (though several elements were captured in abstract models). The principles and the activities of the validation study were based on a set of interrelated assumptions. They were treated in the promotion research as (axiomatic, intuitively accepted) working hypotheses concerning (i) the decision behavior of the designer, (ii) the selection of the decision variables, and (iii) the specification of the decisional modes and the decisional model. In overall, we hypothesized that these were necessary, and that the designer decisional behavior could be described satisfactorily by these, and the decisional options could be properly captured. We did not find any evidence about the incorrectness of these assumptions neither at completing the research work, not at assessing the logical

and computational results.

The validation process was divided into three stages, (i) the preparation stage, (ii) the execution stage, and (iii) the evaluation stage. This was a necessary and useful action to reduce the innate complexity and make the flow of the specific actions transparent and supervisable. Since participation of human designers in the validation process could not be considered for technical reasons, a synthetic validation agent (SVA) was conceptualized as a means for generation a synthetic validation dataset. Thus, the SVA is the kernel instrument in the validation study. The construction of the SVA happened in the same programming environments as was used for programming of the demonstrative part of the ARF. The typical methods of using the SVA were (i) computational simulation of the SVA, and (ii) the analysis of multiple scenarios. The SVA was used both as a research instrument and as a flexible software tool in the simulative part of the validation study. Below, we revisit two aspects of the validation methodology: (i) the procedure of validation, and (ii) the use of the methods for the validation.

As briefly mentioned above, the decomposition of validation process into the sub-stages was advantages since it helped the clarification of the contents and the planning of the activities for validation. In each stage, a specific objective and a set of concrete activities were specific. It provided a kind of check list of what to do for the validation. The decomposition was challenging since it needed a strong mental model and a clear view on the logical sequencing of the planned activities and the information sharing between them. This challenge was present mainly in the preparation stage, since the functions of the SVA needed to fulfill the functional requirements were not well-identified. On the other hand, the programming of the SVA was tested and, technically, it worked properly. We were able to generate the validation dataset in the execution stage by using the SVA for simulation, but the dataset proved to be insufficient for the evaluation of the usefulness of the recommendation in the evaluation stage. Therefore, the simulation had to be repeated on a broader base. Another procedural inconsistency was observed concerning the specification of the validation scenarios and development of the SVA.

As a time-consuming task, the conceptual development of SVA happened in the preparation stage. However, the specification of the validation scenarios happened in the execution stage, based on the logical sequence of the planned activities. It was not clear in the preparation stage what validation scenarios would the SVA be used. Hence, the issue was that the simulation with the help of the SVA in the execution stage might not produce the expected data and the testing scenarios ought to have been revised. Consequently, modification in the programing of the SVA was needed. Together with the detailed functional testing of the SVA in the development phase, it resulted in some repetitive and unnecessary extra work.

As mentioned above, SVA-based simulation was used as the method of generating the validation dataset. Concerning the latter, two testing scenarios were identified. The expected results were the correlations of the considered variables (e.g., decisional modes, probability of acceptance of the recommendation, the probability of the decision options,

and the usefulness indicator). The results produced by the repeated simulation showed that the SVA-based simulation was an appropriate method when participation of human designers in the validation process could not be realized. Notwithstanding, we had to face some limitations in terms of this approach, namely: (i) the decisional behaviors mimicked by the SVA were limited by the underpinning assumptions, and (ii) a comparative empirical evaluation of the obtained simulation results was not possible in the lack of 'in vivo' experimentation and testing of multiple samples.

Considering the need for scientific rigor, it means that evaluation of the usefulness was correct and acceptable only in the context of using the SVA. The results cannot be applied or scrutinized in other contexts without the modification of the assumptions and fundamentally changing the research design. Moreover, it must be mentioned, that the validation dataset was generated based on the sample data compiled for the particular validation scenarios. Since not actual empirical data were available to evaluate the correctness of the simulated results, only logical validation could be applied.

5.6.2 Evaluation of the usefulness of recommendation in action

The evaluation of the usefulness of the recommendation happened based on the simulation dataset generated by the SVA. The data generation was combined with the simulation of the case-related recommendation generation. The validation dataset allowed us to explore the relationships of the considered variables, in particular the relationship of the decisional modes and the patterns of the decisional options. This analysis provided an informational basis for the evaluation of the usefulness of the recommendations (including the derivation of the usefulness indicator and its features). However, the evaluation of usefulness was limited only to the quantitative aspect of the recommendations since it was done based on the comparison of the proportions of common knowledge shared by the SVA with regard to that of the RPP. As a variable, these proportions were supposed not only to measure how many knowledge elements of the SVA possessed related to the design process, but it also referred to the (cognitive and rational) competency of the SVA. It implied that the SVA was able to infer about the contents of the recommendation before making decision to accept or reject the recommendation. However, this feature was not yet included in the decisional model of the SVA since it needs further insightful studies and a more comprehensive model.

Considering the evaluation of usefulness in actual design processes, the qualitative aspect of the recommendation may be more important than the quantitative aspect. However, this is a complex research issue on its own. It concerns the meaning, interpretation and comprehension of the recommendation, as well as semantic, pragmatic, and apobetic computing. Due to the above characteristics, it is not possible to directly measure how much knowledge is possessed by an individual human designer, as it was done in the case of the SVA. The pragmatism and contextual dependence of content-wise recommendation should be taken into consideration in specific (real-life) design processes, as well as the communication and collaboration of multiple designers in a team. Task complexity is

also an aspect to consider in future studies. However, we believe that the concept of the decision options and the usefulness indicator can be applied to evaluate the usefulness of recommendation in real-life. It remains a challenge to learn what and how to capture the qualitative aspects of the recommendations and to determine the acceptance probability and usefulness with the consideration of this.

5.6.3 Some improvement opportunities for the validation

Concerning the validation process:

- Creating an activity-flow visualization for the whole validation process to show the planned sequence and interrelatedness of the activities.
- Specification of the information for the required planned activities and creating an information flow visualization to ensure that the required information is available when it is needed.
- Identifying the critical activities, which has the largest influence on the development of SVA and the evaluation of usefulness of the recommendation.
- Considering of external and internal communication and search opportunities to create a robust informational basis.

Concerning the fidelity of SVA

- Developing a more comprehensive (and more articulated) computational model that is based on behavior mapping of real-life human designers.
- Representing the qualitative aspects of the decisional behavior in the SVA (i.e., modifying the decisional model of the SVA by adding new factors and parameters and conducting experiments to see how much the SVA is able to recognize the usefulness of the recommendation contents with the enhancement).
- Creating semantic relationships between the contents of the recommendation and the decisional model of the SVA in order to enhance the likelihood of the decision behavior of the designer.

5.6.4 Some recognized limitations

In the previous section, we highlighted those actions which would improve the use process and the fidelity, respectively, of the SVA. Though these enhancements are unquestionably useful, they do not resolve some issues that originate in the fundamental concepts and assumptions related to the implementation of the SVA implied validation methodology and the SVA as a smart validation agent itself. According to our view, these need additional research and development and therefore were decided as outstanding from the focus of this thesis. Nevertheless, we identified the following limitations the elimination of which could improve both the quality and the efficiency of the validation. These are as follows:

- The lack of published literature on the decisional behavior of designers, in particular in context of designing reasoning mechanisms, restricted and made it difficult to enrich

the decisional models of the SVA with available research data.

- The current decisional behavior of the SVA has been generated based on pragmatic assumptions, rather than a comprehensive and tested underpinning theory. This influences the reliability of both the SVA as a computational agent and it application methodology (which is now simulation focused rather than reasoning focused).
- There were no empirical data available for a comparative evaluation of the dataset generated by the SVA. The evaluation of the usefulness based on a full-scale and fully-fledged dataset in demanded and feasible if further research is done.
- At using the SVA, the lack of the consideration of the qualitative aspects of the evaluation of usefulness of recommendations is still a limitation. However, it leads to a completely different (intellect- rather than data-based) reasoning model.
- Although there was an effort to consider the recommendations content-wise, the application context was however not considered in the used validation process.
- The validation was done only using one methodology, but using multiple approaches in concert would result in a more robust and reliable methodology, though the overheads would also be increased.

References

[1] Mendoza, M., & Torres, N. (2020). Evaluating content novelty in recommender systems. *Journal of Intelligent Information Systems, 54*(2), 297–316.

[2] Kang, L. T., & Wang, Y. (2014). Seven factors in evaluating recommender system. In *Applied Mechanics and Materials* (Vol. 472, pp. 443–449. Trans Tech Publications Ltd.

[3] Jain, S., & Patel, A. (2021) Semantic Contextual Reasoning to Provide Human Behavior. Retrieve from http://arxiv.org/abs/2103.10694.

[4] Holmgren, J., Davidsson, P., Persson, J.A., & Ramstedt, L. (2012). TAPAS: A multi-agent-based model for simulation of transport chains. *Simulation Modelling Practice and Theory, 23*,(2012), 1–18.

[5] Rizzi, D.A. (1993). Medical prognosis - Some fundamentals. *Theoretical Medicine, 14*(4), 365–375.

[6] Oh, J., Meneguzzi, F., Sycara, K., & Norman, T.J. (2013). Prognostic normative reasoning. *Engineering Applied Artificial Intelligence, 26*(2), 863–872.

Chapter 6

Reflections, conclusions, propositions, and recommendations

6.1 Reflections on the scientific and professional contributions of the research

The general objective of this Ph.D. research was to provide evidence that the Ph.D. student can do self-governed research and can contribute to the disciplinary development with novel tested knowledge. The specific objectives were to aggregate knowledge for, and to develop some demonstrative implementation of, an active recommendation framework (ARF) as a design enabling tool. The purpose of development was to support the design of application specific reasoning mechanisms (integral sets of computational algorithms) for S-CPSs. Driven by these objectives, four research cycles were planned and completed.

The four cycles addressed different aspects of the ARF development in the application context of an automated parking assist system as the target ASRM. The activities included: (i) knowledge aggregation, demarcation of the domain of interest, and specification of requirements; (ii) functional and architectural conceptualization of the active recommendation framework; (iii) computational implementation and operationalization of the demonstrative modules; and (iv) validation of the usefulness of the recommendations generated by the implemented demonstrative modules of the ARF. The following sub-sections present my personal reflections on the work and its findings, as well as a self-evaluation of the results of the completed research. They discuss the contributions made to: (i) the academic and practical knowledge; (ii) the development of active recommendation frameworks; (iii) the conceptualization of ASRMs; and (iv) the development of mechanisms for real-life automatic parking systems.

6.1.1 Contribution to the academic and practical knowledge

The contribution of the completed research to the academic and practical knowledge concerned two scientific areas: (i) the field of intellectualized design support tools; and (ii) the field of cognitive engineering of reasoning mechanisms for practical applications.

Contribution to the field of intellectualized design support tools

The promotion research contributed to the field of intellectualized design support tools from three aspects. First, we proposed a theory to underpin the conceptualization of the ARF through the integration of two fundamental concepts, namely: (i) the concept of active frameworks; and (ii) the concept of recommender systems. The concept of active frameworks assumes that a computational system is able to actively monitor behavioral changes of the designer in the design process at runtime. The concept of recommender systems assumes that context-sensitive recommendations can be generated to support design problem solving, for example, in the development process of ASRMs.

Second, we applied an application-oriented approach to demonstrate the crucial elements of the conceptualized ARF. At the demonstrative implementation of the ARF, the functionality needed for an automatic parking system was considered and tested. Using the domain knowledge related to the WPE design session, the implemented modules of the ARF support the design of an ML-based complex selection algorithm. This algorithm is able to search for proper parking plans based on a set of stored parking scenarios. All of these are in line with our working hypothesis stating that an ARF cannot investigate the state of design actions without domain specific knowledge and contextual information about the design process.

Third, a synthetic validation agent (SVA) was conceptualized to simulate the decision-making behavior of (human) designers as a computational validation means and approach. The validation study focused on the quality of the recommendations in term of their usefulness. This characteristic can be interpreted in a broader and a narrower meaning. We used the term in the narrower meaning that considers an offered recommendation useful if a designer facing a procedural obstacle can resolve the problem and can continue the design process based on the recommendation. The SVA generated the dataset to mimic the (human) designers' decisional behavior as a follow-up on the proposed recommendations. The principle of prognostic reasoning was used to predict the possibilities of decision options based on the mimicked decisional behavior. As a result, a socially-based indicator could be realized to express the usefulness of the obtained recommendations based on the justified objective decisions of the designer.

Contribution to the field of cognitive engineering of reasoning mechanisms for practical applications

On the one hand, cognitive engineering (CE) means equipping systems with system-level decisional intellect. On the other hand, CE means adapting the intellectualized systems to the physical and cognitive behavioral processes of the stakeholders. These two dimensions of CE are present concurrently at the development of the ARF that offers design support

means and tools for the development of smart reasoning mechanisms for S-CPSs. From an information engineering point of view, the knowledge about the design actions and the part of the design process in the WPE session was captured and processed into system knowledge in the form of a reference process protocol (RPP). In the particular application context of an APAS, the use of the RPP involves causal probabilistic reasoning in combination with ML-based reasoning. Concerning the knowledge and inference methods, CE equips the ARF with the capability of supporting design decisions.

6.1.2 Contribution to the development of active recommender frameworks

The concept of an active recommendation framework is novel and initially proposed by researchers at the hosting Section. The term "framework" means a purposeful enabler that arranges and rationalizes design activities, information processing, and designer-system interaction. The term "recommender" expresses that, as a complex system, the ARF derives context-dependent advices for the designer based on a comprehensive system model of the concerned (specific) design process. The term "active" refers to the fact that the ARF continuously monitors the design process and spontaneously interacts with the designer wherever it is needed in the design process.

From the viewpoint of a computational system, the ARF was proposed as a design-action driven context-sensitive recommender system. The ARF is capable of (i) monitoring what is happening in the design process, (ii) identifying where a procedural obstacle is, and (iii) offering personalized recommendations to the designer to help proceed in the design process. This assumes not only monitoring of the process, but also dealing with the information contents of the design activities. From a computational point of view, it has an innate complexity. Consequently, it could not be implemented in full scale. A demonstrative part was realized that is able to present the novel functional abilities of a fully-fledged implementation. This demonstrative part operationalizes two interoperating mechanisms, namely the process monitoring and the decision support mechanisms.

Considering the development process of the ARF, four aspects of its methodological contribution are as follows.

Contribution concerning the development methodology

There was no standard approach to the development of application-orientated engineering recommender systems with specialized processing monitoring capability. We proposed and used a multi-layer methodological approach for conceptualization and implementation of the ARF. It is a top-down approach, which relies on systematic decomposition of the constituents of the ARF system according to four perspectives: (i) functionality; (ii) architecture; (iii) computational algorithms and data constructs; and (iv) computational workflow. In addition, through the applied multi-layer abstraction of the ARF, it provides a logical scenario of the activities. We posit that, in comparison with the traditional methodologies used at the development of similar frameworks documented in the literature, this approach effectively handles the intrinsic complexity of recommender system development.

Contribution concerning the conceptualization of the novel system functionality

Another novel methodological contribution of this work is the conceptualization of the functionality for recommendation generation using the reference process protocol. To facilitate a proper continuation of the design process, recommendations are created by the cooperative use of two inference methods: (i) probabilistic reasoning by means of joint probability distribution (JPD); and (ii) model-based reasoning using a decision tree model. The RPP represents the procedural network of design actions making up the considered design process. The ARF generates the process-based recommendation by considering a segment of the flow of the design consisting of three design actions: (i) the preceding design action, (ii) the current design action, and (iii) succeeding design action. The RPP is implemented in the form of a Bayesian network.

Based on the probabilistic relationships captured in the Bayesian network, the ARF can find the proper preceding design action by investigating the RPP retrospectively. To modify the current design action, the ARF uses the decision tree model to select the most appropriate method. Relying on preceding design actions and the (modified) design action at hand, the ARF determines the joint probability distribution and finds the next design action in the RPP toward the completion of design process. The cooperative use of a probabilistic graphical model and ML-based reasoning with regard to the conceptualization of the inference engine is a novel functionality of the ARF system.

Contribution concerning computational implementation

The MATLAB package was used for the computational implementation of the demonstrative modules of the conceptualized ARF. This developer package provided many reusable resources for programing. The reuse of pre-programmed algorithms was useful to reduce the workload and allowed us to concentrate on the development of brand-new algorithms from scratch. Some noteworthy elements of computational implementation can be summarized as follows:

- The functional specification was done with a view to the interrelated computational components. In line with the stated requirements, the computational component was designed and implemented to realize one primary function – preferably, by one comprehensive algorithm. The functionally critical algorithms were identified based on three criteria: (i) functional complexity, (ii) data sensitivity, and (iii) intensity of human interaction. This resulted in simplicity and transparency. In addition, this approach ensures that modification of an individual algorithm has the lowest possible impact to the system-level functionality.
- From a structuring point of view, the architecture of the whole ARF reflects a hierarchical structure. The highest-level architectural element is the whole ARF itself, dedicated to the system-level functionality. The lower-level components manifest on (i) mechanism level; (ii) module level; (iii) sub-module-level; and (iv) computational component (interrelated algorithms) level. Encapsulated in computational components, the algorithms are the lowest-level architectural elements.

- In order to fulfil the structural requirements, a hierarchical composition approach was employed that supports the handling of the implementation complexity. Accordingly, the lowest possible number of components and their relationships were strived for at each level of architecting. By doing so, the lowest possible dependencies among the elements were achieved. In turn, it ensures that the efforts needed for structural adaptation at the framework level will remain low (if modification of the algorithms is needed).
- The reference process protocol was proposed as an essential means for the implementation of the process-based recommendation generation. It is system of knowledge, which is supposed to known by the ARF. The RPP lends itself to a systematic exploration of the potential design flows as well as to the investigation of design process and its action elements. The hybrid inference utilizes the RPP to infer the most proper design activity flow and propose it to the designer.
- As a 'domain of interest', the WPE selection session of the design process of the APAS was considered. Within this, special attention was paid to the critical design task, $\mathfrak{D}_{1.0}$, and it was used in the functional testing of the implemented demonstrative components. The contribution to the WPE selection session was an ML-based algorithm, which is able to predict the most appropriate parking case to start with. The ML-algorithm is also able to select the applicable motion path for real-life parking scenarios.
- validation of the system-level functionality of the ARF. Based on the simulations, it was confirmed that the selected parking trajectories were appropriate and doable in the various parking cases studied. Based on these, it is fair to claim that the support services provided by the ARF were efficient for this design task.

Contribution concerning the validation of the implemented demonstrative part of the active recommender framework

The goal of validation of the results of ARF development was to check if the proposed solution does, or not, what it was supposed to do. Concerning the method of validation, it is important to mention some pieces of background information here. It is traceable in the literature that using 'artificial humans' in repetitive validation tasks has become a trend in the literature in the last few years. This approach offers not only new opportunities, but also many benefits. Though remote on-line testing of networked systems has also become a daily practice, the experiences showed that this cannot be so well controlled as the on-site (participatory) experimentation. In the context of validation of the implemented parts of the ARF, the following novel methodological elements were considered:

- A logical and computational model of a synthetic validation agent (SVA) was elaborated as a means for simulation of human decisional behavior. The SVA generates a validation dataset that mimics the decision patterns of human designers. Three influential factors of the decisional behavior of human designers (as problem solving agents) were considered: (i) the appropriateness of a recommendation, (ii) the recognition of its properness by the designer, and (iii) the possibility of selecting the proper recommendation. These not only have an influence on the decision-making

behavior, but they are also correlated with the usefulness of the recommendation. Based on these factors, the decisional behavior of the designer was modelled and represented by eight decisional options. These decisional options were classified into three groups: (i) justified objective decision, (ii) unjustified subjective decision, and (iii) incorrect subjective decision.

- Having the dedicated SVA, the usefulness of recommendations could be tested without direct participation of human designers. The analysis of usefulness usually required semantic interpretation from the designers' points of view. The assumption was that, depending on their decisional behavior, the designers would take different decisional options given the same recommendation. The sequence of their decisions forms a decisional path including a combination of negative and positive decisions on the recommendations. Thus, usefulness was defined more narrowly. A received recommendation is useful if the designer can continue the design process with the proposed design flow and action. With this interpretation, the operation of the SVA generated the dataset upon which the assessment of the usefulness of the recommendation was possible.
- To assess the usefulness of the individual recommendations, indicators were generated based on the principle of prognostic reasoning. These indicators were derived from the correlations of common knowledge elements and the collective decisional behaviors of the designers. The aggregated probabilistic of the justified objective decision and the unjustified subjective decision can be defined as the metric to evaluate the usefulness of the recommendation.

6.1.3 Contribution to the design methodology of application-specific reasoning mechanisms

Oftentimes, application-specific reasoning mechanisms are more complex structures than generic inferring and reasoning mechanisms. This complexity is typically originated in the complicatedness of real-life problems as well as in the concomitant data, information and knowledge acquisition, pre-processing, and verification/validation tasks. Thus, the design methodology of ASRMs should focus on the complete set of (possible or optimized) problem solving procedures that are relevant for the application problem at hand. This is in contrast to the conceptualization methodology, where the generic reasoning mechanisms are developed based on logical inferring schemes or analogical search processes. The traditional design methodology of ASRMs starts out from a logical and/or a procedural model of operation without the consideration of the dynamic application context. A typical example is a cruise control.

In our conceptualization, the design methodology of ASRMs covers the whole process of application problem solving, and the varying application context. The contribution of the promotion research to the design methodology of application-specific reasoning mechanisms manifests in the systematic approach to service-oriented problem interpretation and in the high-level decomposition of the design process into purposeful sessions. The proposed approach to APAS conceptualization includes the following sessions: (i) computational

formulation of the problem, (ii) situation modelling and analysis, (iii) working principle exploration, and (iv) decision logic modelling. It is complemented by (v) component-level computational detailing, (vi) design and interfacing of the algorithms, and (vii) system-level prototype integration. Focusing on the particular WPE session, we demonstrated the working principle exploration, activities in the context of APAS development.

6.1.4 Contribution to solution generation for a real-life automatic parking problem

Known to be time-consuming and frustrating, the search for a parking space in urban areas needs long-range search and navigation activities, as well as short-range maneuvering activities. Long-range search is connected to car park guidance systems (CPGSs) that provide information about location and vacancy by digital communication. Thus, CPGSs support drivers in their search for and navigation to an available parking space. Short-range search can also be based on local CPGSs or, alternatively, on monitoring by self-observation. The real-life need for automatic parking may appear in the special contexts of (i) street (roadside) parking, and (ii) zone (multi-lot) parking.

Automated parking may be combined with both human driving and automated driving. In the former case, the assumption is that the driver presses the park key after entering the parking zone, and the car will store itself either on the roadside or in a parking area. In the latter case, the passenger informs the car about her/his intention to stop and park, and all follow-up actions, including payment, are done by the self-driving car. The execution of parking needs computations concerning (i) parking vacancy detection and evaluation, (ii) reasoning about the best parking lot, (iii) parking the car in the selected lot, and (iv) completing the follow up activities that are needed. For input data generation, video and sensor technologies are used individually or jointly.

The above description shows that vehicle/car parking is a complex task and its automated execution requires multiple smart and collaborative computational reasoning mechanisms. These computational mechanisms are functionally connected not only to regional CPGSs, but also to (i) input sensing (e.g., using magnetometer and radar for a dual detection and monitoring the occupancy of a parking space) and (ii) steering of the car (e.g., fine control of the motor, steering mechanism, and applying/releasing the breaks). As it can be seen in the literature, current development efforts mainly focus on the physical part or the software development part of the parking process, and not on the situated reasoning part.

However, deep learning has recently been used in building practical applications, but this can only support the related lot recognition and classification problem. The other activities, like car control generation, are yet not sufficiently covered by computational problem solving. On the other hand, the proposed ARF has been conceptually equipped with the abilities that are needed to support the design of these dedicated reasoning mechanisms. It has been developed based on typifying the design activities concerning the various reasoning mechanisms, and representing the design activity flow in a reference process

protocol. This is a kind of forward-looking conceptualization that considers the end goal and identifies the phases (steps) of the process that are needed to achieve the target.

6.2 Main conclusions

6.2.1 Conclusions concerning research cycle 1

Reflections on the methodological approach

The main objectives of the first research cycle were to systematically review and to criticality investigate the state of the art of system engineering frameworks (SEFs) related to the computational implementation of system-level reasoning. A comprehensive literature study was done by applying both quantitative and qualitative methods. The quantitative study aimed at exploring the landscape of publications related to the research phenomenon. The bibliometric map was constructed based on the wide range of key terms, using the VOSviewer software. The primary domain of discourse of the literature study was associated with the notion of "framework". In addition, the interrelatedness of the key terms was analyzed.

The strength of the relationships was exposed and displayed in the form of a power map. A formal reasoning model was derived from the power map for the qualitative study, which included content analysis and interpretation of findings. The requirements for the ARF were derived by considering the implications of the findings. The requirements were formulated in the regular textual form and their relationships were explored and represented as semantic maps. The major reflections on the methodological approach are as follows:

- This research cycle took advantage of data analytic techniques and software tools which could be used for visualization of data and their relationships. This helped create a visio-graphical overview of a huge number of publications and the quantification of the interrelationships of the key terms. However, the map did not convey the semantic meaning of their relationships directly. The interpretation of the map required the background knowledge of the researchers. Our intention was to exclude or at least to reduce any bias that might have happened during the process of filtering, combining, and merging of key terms. However, the subjective interpretations could not be fully scrutinized.
- In the phase of requirement engineering for the different levels of the ARF, the sematic map helped us identify the key requirements. A major concern was their feasibility (i.e., the possibility of fulfilling their expectations at specification of the functionality of the ARF. The investigation of the sematic relationships among the requirements cast light on possible inconsistencies among them. We found that the semantic mapping was an effective means for the feasibility assessment of the various requirements.

Reflections on the results

The knowledge exploration and aggregation process was supported by a transparent mental model. This reasoning model focused on system-level reasoning about frameworks and

introduced six domains of interests as contents. The five included system-level knowledge domains are: (i) synthetic knowledge, (ii) system awareness; (iii) reasoning mechanisms; (iv) decision-making; and (v) system adaptation. These were regarded as enablers for system-level reasoning operation. The sixth domain, recommendation generation, was considered as a manifestation of system-level service. As mentioned above, in the process of content analysis, the implications of the findings were analyzed with the intention to identify the requirements for the conceptualization of an ARF. The main reflections on the knowledge aggregation phase are as follows:

- We used the term '*framework*' as the main key term to perform the search for relevant publications and to get to a bibliometric map. Based on the follow up data analysis, a network of the related key terms was created and graphically presented. The related key terms were extracted from 2,096 publications. The term 'framework' appeared as the intermediary link between the other key terms in the network. This was a kind of confirmation that the collected publications were relevant to the interest in frameworks. However, it could not be guaranteed that all publications on frameworks were indeed taken into account in the literature study.
- In the later stage of the promotion research, we focused on the design support for the development of smart reasoning mechanism. The necessity of monitoring the design process was not a part of our mental model yet. After conducting the literature study, it became clear that providing support service only to solving the application-specific design problem was not sufficient. Most of the traditional SEFs operated in a static manner. This means that they were not developed to be sensitive to the changes that may occur in the context of operation and the behavior of the system at runtime. If these traditional principles would have been applied, then a design support framework could not recognize what is happening during runtime operation. To avoid this situation, it is necessary to equip an ARF with real-time process monitoring capability. Considering this fact, we concluded that process monitoring and design decision support had to be seen not only as essential, but also interrelated functionalities with regard to the operation and computational implementation of the targeted ARF. Their interoperation makes it possible to generate context-sensitive recommendations adapted to the runtime events and situations in real time.
- The finding related to the reasoning mechanism development made it clear for us that widely-based additional research would have been needed to: (i) explore semantic relationships of data and information elements of emerging situations and unknown operation based on analogies or ontologies, (ii) create belief networks for representing potential associations of knowledge elements in order to fill in incomplete knowledge and information over processes, and (iii) adapt reasoning strategies to ill-defined problems and heterogeneous knowledge and information representations. However, due to the time and capacity constraints, these could not be considered during our research.

6.2.2 Conclusions concerning research cycle 2

Reflections on the methodological approach

The objective of the second research cycle was to conceptualize the whole, and in particular a testable demonstrative part, of an ARF to support the development of ASRMs. The main research question of this research cycle was associated with the conceptualization of a feasible and efficient ARF. The conceptualization of the whole and the constituents of the ARF happened in terms of functionality, architecture, computational algorithms, and computational workflow. To cope with the complexity of the whole system, a top-down decomposition approach was used. As discussed above, the functionality specification and testing of the proposed concepts were done by concentrating on the WPE session of the design process of an APAS.

The methodological approach of conceptualization relied on a four-layer decomposition of the ARF development process, this included (i) specification of functionality, (ii) allocation of the functionality into system architecture, (iii) specification of computational algorithms and data structure, and (iv) organization of the operation workflow, including communication with the designer. The reflections on the methodological approach are as follows:

- The complexity of the mechanisms of the ARF was challenging from a realization point of view. By using the multi-perspective conceptualization approach in the ARF development, we could effectively handle the challenge.
- The advantage of the multi-perspective conceptualization approach was that it allowed us to consider that the constituents of the ARF were interrelated both on the same layer and across layers. It also facilitated a kind of pathfinding to the most appropriate conceptual elements of the ARF.
- The dependencies introduced by the four-layer logical decomposition structure created a non-linear design process, as opposed to the traditional water fall models. Execution of a non-linear design process is a time-consuming task. Rectifying the concept on one layer will most probably require the correction of the other layer too.

Reflections on the results

The main findings of the conceptualization of the ARF can be summarized as follows: (i) type B observation of non-usual events was chosen as the basis of the conceptualization of the ARF, (ii) two main functionalities (i.e., process-monitoring and decision-support) were needed and allocated to two mechanisms in a one-to-one manner, (iii) the mechanisms were composed of the lowest possible number of modules and their interactions, (iv) the lower-level architecting elements were constructed following the same expectations, (v) the architecture of whole ARF was constructed by two mechanisms, six main modules, twenty sub-modules, and sixty-three enabling algorithms, (vi) the reference process protocol (RPP) was introduced as an essential component of the ARF, (vii) the RPP was both a conceptual means of capturing system knowledge and a computational means for the generation of process-based recommendations, (viii) two inference approaches

(i.e., exact inference and hybrid inference) were employed to generate context-sensitive recommendations, and (ix) concentrating on the WPE session only, the ASRM of an APAS was used as contextualization of the RPP.

The reflections on the results obtained in the second research cycle are as follows:

- Compared with similar SEFs, the proposed concept is novel in the field of software system design and engineering for three reasons: (i) handling the interdependence by a multi-layer design methodological approach; (ii) offering a context-sensitive decision support mechanism for the development of ASRMs; and (iii) providing recommendation services using a runtime process monitoring mechanism.
- Even in the conceptualization phase of the ARF, we had to realize a large complexity challenge. This became obvious in the implementation phase. Therefore, we tried to keep the number of elements and their relationships to the lowest possible level. Nevertheless, on the level of computational algorithms, sixty-three algorithms were required to realize the system-level functionality of the demonstrative part of ARF. Due to resource limitations and time constrains, it was not possible to implement the whole ARF and to test it in a real-life environment.
- Due to the technical issues experienced in terms of accessibility and monitoring of the software execution data of the used design tool, we decided to use a designer activity-based monitoring approach. This type of monitoring allowed not only to observe the executed design actions, but also to observe the designer's facial expressions. Interestingly, we found that the latter enabled capturing the cognitive thinking of the designer with regard to the design activities as well as the events in the design process. It offers an opportunity to generate recommendations based on the pattern of design activities. However, this remained an open issue for future works.
- In the conceptualization phase, we considered two separate inference approaches for generation of recommendations. The exact inference approach infers a solution based on the information obtained from a dialogue with the designer. If a solution cannot be found this way, then hybrid inference is necessary. Hybrid inference generates context sensitive recommendations in an automated operation based on the RPP. It must be mentioned that, in a retrospective investigation of the RPP, it might happen that information about the state of preceding design entity is not available. Updating the current state of design process needs acquiring state and/or context information from the designer. This is a computational gap of the automated recommendation generation (and of context-sensitive recommendation generation).

6.2.3 Conclusions concerning research cycle 3

Reflections on the methodological approach

In short, the third research cycle intended to (i) implement the demonstrative modules of the ARF, and (ii) test the system-level functionality of the ARF in the application context. To avoid an uncontrollable complexity of the implementation, the divide-and-conquer strategy

was applied. It involved using (i) a multi-layer structure, (ii) modular design technique, and (iii) object-oriented programming. For the purpose of demonstrative implementation, MATLAB package was used. The reflections on the methodological approach are as follows:

- In order to handle computational complexity successfully and to realize the implementation in a constrained timeframe and with the available resources, we had to apply a strict scoping on the implementation. We could consider only a demonstrative part of the ARF, in particular, with regard to context-sensitive recommendation generation using the RPP. Nevertheless, some other related modules were also needed to complete the operation workflow.
- The functional validation of the demonstrative part was completed with a focus only on the design process elements in the WPE design session. This aimed at the development of search algorithms for selecting the proper motion path for the actual parking problem. Thus, the testing of the functionality was done based on a scenario of the relevant design actions, and not throughout the entire process of designing all algorithms required for the operation of the ARF. Therefore, the usability of the ARF in the application context and the end result of the programmed search algorithm could not be tested without considering the complete design process.
- We applied the reasoning with consequences principle to test the system-level functionality of the ARF. This approach is based on a combined logical and analogical validation. If the ultimate output is correct, then the hypothesis will be valid. Notwithstanding the abovementioned limitations, we found that this logical validation approach was beneficial in our case.

Reflections on the results

Four demonstrative modules (i.e., the DOI, ROI, RPC, and ACG modules) were implemented. They collectively included thirteen sub-modules and thirty-two computational components. Thirteen algorithms have been discussed in detail. The algorithm A4.04 (selecting the best usable method) was considered as the critical algorithm, based on its foreseen overall impacts. It plays a role in the hybrid inference, which is executed based on the interoperation of three other algorithms (including the algorithm A4.01, A4.03 and A4.06). The following reflections can be made concerning the results of the third research cycle.

- The specification of functionality revealed the opportunity of using existing algorithms and increasing the utilization of standard algorithms. The organization of the workflow and the interaction processes provided opportunity for the development of optimized algorithms.
- With a view toward a future intelligent (automated) ARF, we decided to operationalize the ARF with a limited amount of interaction with the designer. This explains the relatively low number of algorithms for human interaction.
- To be able to sufficiently support application-specific design tasks, the construction of a reference process protocol required domain-specific knowledge. In the functional

testing, we assumed that all relationships of the design entities in the RPP, as well as the decision variables for selecting the usable method, were theoretically correct. The recommendations were generated based on this assumption. However, the correctness of the theoretical basis of the recommendations was not experimentally tested. This must be noted because the quality of the recommendations might not satisfy the requirements completely, but it was sufficient for the demonstrative purpose.

6.2.4 Conclusions concerning research cycle 4

The objective of the fourth research cycle was to validate the usefulness of the recommendation provided by the ARF. We defined the term 'usefulness' to specifically mean that the recommendation is useful if the designer can unblock the obstacle and continue the design process. Concerning the validation, we faced difficulty with an assessment with the involvement of practicing designers. One of the difficulties was caused by the coronavirus pandemic, which did not make possible to invite designers to conduct on site studies. This orientated our attention to other methodologies and computational solutions. This is the main reason why we discovered and introduced the concept of a synthetic validation agent as the surrogate of the designer to handle the situation. The synthetic validation agent aims at mimicking the decision behavior of the designer and generating a quasi-experimental dataset for the purpose of validation. To capture the possible decisional options of the designers, three decision variables were identified. The interplay of the decision variables offered eight options, which were sorted into four classes, (i) justified objective decision; (ii) unjustified subjective decision; (iii) incorrectness subjective decision; and (iv) negatively justified objective decision.

The agent-designer made the decision based on the assumptions of the possession of common knowledge elements shared by the SVA and the RPP. The decision model was derived from the relationships of the probability of shared knowledge and the probability of knowing the knowledge elements included in the recommendation by the SVA. Three decisional modes were the components of the decisional model. The prognostic reasoning was applied when deriving the indicator of the usefulness. The decisional options were analyzed and based on the interpretation of usefulness. The first two decisional options were selected as indicators of the usefulness.

Reflections on the methodological approach

- A synthesis validation agent (SVA) is an effective means for the generation of the synthesis dataset. In our work, it provided the meaningful dataset which can be used for the analysis of the correlations of variables for different points of view. The challenges for the development of the SVA are how to capture the decision behaviors of the human and how to validate the decisional model which operationalizes the decision-making of the SVA. The latter is even more challenge from our point of view.
- Three decision variables were sufficient for capturing the designer decision options. Their interplay simplified the human decisional behaviors and limited the total number of the decisional options. This handled the challenge of how to capture the collective

behaviors of the designers without the participation of the human designers.

- The quantitative analysis is typically used when deriving the decisional model of the SVA. We introduced new variables which were related to the relationship of the knowledge possession of the SVA and the knowledge elements included in the RPP. Three decisional modes were described and quantified as components of the decisional model. However, in many problems, it is difficult to collect the data for evaluating the model. The logical validation was applied in this situation. It is assumed that the decisional model is valid if it gives the reasonable results.
- In our study, the SVA-based simulation was used as the method of generating the validation dataset. The expected results were the correlations of the considered variables (e.g., decisional modes, probability of acceptance of the recommendation, the probability of the decision options, and the usefulness indicator). The results produced by the repeated simulation showed that the SVA-based simulation was an appropriate method when participation of human designers in the validation process could not be realized.
- Notwithstanding, we had to face some limitations in terms of this approach, namely: (i) the decisional behaviors mimicked by the SVA were limited by the underpinning assumptions, and (ii) a comparative empirical evaluation of the obtained simulation results was not possible in the lack of 'in vivo' experimentation and testing of multiple samples.

Reflections on the results

- The findings seem to contradict the trend of developing the recommender systems, which should contain a huge number of recommendation items. In our work, we concluded that the recognition of the appropriate recommendation by the designer is more important than its quantitative matter. According to the proportion of common knowledge shared by the SVA and the RPP, if it was very low, there was a high probability that the SVA could not recognize the appropriateness of the recommendation. This condition implied that if the RPP could represent a rather complicated design process, there was a high possibility that an unfamiliar recommendation was generated with regard to the specific context information. This made the designer uncertain about the offered recommendation. Therefore, for the development of the specialized recommender system, it should consider a sensible number of recommendation items to be able to yield the optimal proportion of common knowledge of the SVA. As a consequence, the SVA was able to recognize the appropriate recommendations with a higher chance.
- Although new aspects of the evaluation metrics concerning the novelty of the recommendation items have been introduced in the recent academic publications, this aspect might not be suited to the development of the specialized recommender system (the ARF for instance). Based on the prognosis of the proportion of common knowledge of the agent designer and the usefulness indicator, the recommendations should be well-known by the designers. In practical terms, it is impossible to evaluate how much possessed knowledge an individual designer. Hence, the offered recommendation

should be practical and executable rather than novel or sophisticated in order to enhance the probability of acceptance of the recommendation.

6.3 Propositions

6.3.1 Scientific propositions

Proposition 1: *The growing complexity of application-specific reasoning mechanisms of smart cyber-physical systems implies the need for active support of software designers.* (RC1)*

The growing complexity of real-life application problems raises the need for sophisticated reasoning mechanisms for S-CPSs. Application-specific reasoning mechanisms (ASRMs) are often characterized by functional complexity, architectural complexity, and computational complexities. In fact, procedural reasoning processes go through multiple stages of information processing (e.g., building awareness, situated reasoning with incomplete context information, informed decision-making, and runtime adaptation). Therefore, their design process usually includes a more complicated set of activities than that of the generic reasoning mechanisms. On the other hand, minor errors and mistakes occurring in the conceptualization stage may lead to unexpected faults in the operation of the designed software mechanisms. As indicated by the increased number of publications in the field of system engineering and CPS development, researchers and system designers are looking for efficient approaches for designing S-CPSs. Many support tools have been developed based on the paradigm of CAD/E systems. The number of publications that propose novel design-support tools for designing smart systems is also growing. Not only multi-disciplinary collaboration and complexity management are addressed, but also setting up application-orientated design scenarios and operationalization of artificial intelligence methods. Providing runtime support for reasoning mechanism design is a new challenge that implies the need for consideration of process-related recommender systems. These developments led to the idea and conceptualization of our active recommender framework for providing active support.

Proposition 2: *Recommendation services must be integrated into the decision support mechanism to guide designers in the development of application specific reasoning mechanisms.* (RC1)*

As members of the family of decision-support systems, the emerging engineering recommender systems are supposed to provide recommendation services according to the state of the process and the solution contents. The provided personalized recommendation service depends on the context information, which is either directly provided by the designer or captured by the system based the decisions made by designers. This trend provided the conceptual idea for the development of an ARF that integrates the add-on recommendation services with the decision support mechanisms.

Proposition 3: *Because of multiple limitations, a fully intelligent (automated) active recommender framework is currently not feasible. * (RC2)*

The state-of-the-art efforts in cognitive engineering are aimed at the development of human-like intelligent systems. System intelligence is the ability of systems to derive possible solutions and to make appropriate decisions even in uncertain situations. What it means in our context is that a fully intelligent (automated) ARF would take over problem-solving when the designer fails and would explore the space of possible solutions. A fully intelligent and automated system is currently not feasible due to lack of knowledge, for instance the limitations of technologies, the task complexity, and the trade-off (balance between developmental investment and achieved practical support/gain) issues. It might be interesting from a scientific point of view, but cannot be operationalized in the case of complex application problems due to computational limitations. Therefore, the most appropriate strategy is to look for pragmatic, but effective solutions based on partial intelligence. In our view, partial intelligence can provide procedural recommendations when the designer is hindered. The level of involvement of the ARF and the goal and form of the contribution of the ARF to the designer's activities is still in an early stage of understanding. Expert designers can cope with the challenges of the design process, but may use multiple, time-consuming iterations, and may ignore mishaps and potential threads. The latter can be the entry points for an ARF to provide support services. This requires processing context information about the problem at hand, the reasoning of the designer, and the actual state of the design process.

Proposition 4: *An active recommender framework needs knowledge about the application domain and the design context to provide proper recommendation services. * (RC2)*

In a typical recommendation system, the so called 'cold start problem' occurs if information about the profile and preferences of the user, and her/his ratings of recommendation items are not (sufficiently) known. These pieces of information are part of the explicit knowledge possessed by the system. Without this knowledge, recommendation systems cannot produce reliable recommendations. To eliminate this problem, context aware recommender systems (CRSs) tend to utilize knowledge about the preferences of the users as well as context information to produce personalized recommendations. In other words, knowledge is an essential element of CRSs that supports situated reasoning and recommendation generation based on the context information. The contents of recommendations depend on the knowledge acquired by the system. In the development process of ASRMs, the application domain and the design context should be converted into system knowledge for the ARF. In our specific case, a reference process protocol (RPP) was the means of capturing and representing formal knowledge about the design process of ASRMs. Recommendations are generated in the actual context of design flow, which is derived from the RPP. It also has to be mentioned that the ARF cannot contextualize the design actions and infer about the progress in the design process without sufficient domain knowledge. Otherwise, there is a high possibility that the delivered recommendations are not proper.

Proposition 5: *The active recommender framework can provide the requested support services by concurrently using runtime process monitoring and context-sensitive decision support. * (RC2)*

To be alert and ready to advise, the ARF must be at least active (in an ultimate case, even proactive), rather than only reactive. The ARF should be aware of both regular and irregular conduct of the design process and the behavior of the designer. According to our proposal, the process monitoring mechanism detects an unusual event during the execution of the design process, and offers the requested support services in (quasi)real time. The support service (recommendation) is generated according to the actual situation and to what is needed to resolve a procedural or cognitive blockage. Therefore, the ARF has been developed to concurrently use the runtime process monitoring and the context-sensitive decision support mechanisms. Both the process monitoring and the decision support mechanisms use the RPP for a context-sensitive generation of recommendations. The proper connection between these mechanisms should be guaranteed by the computational implementation.

Proposition 6: *Either consistency of the information flow or change of the designer's behavior provides sufficient clues for detecting non-usual events.* (RC2)*

An event is indicated by the changes in the state of the system and/or the behavior of the designer observable at a moment in time. Events allow inferences about something that might go or have gone wrong in the design process. A non-usual event (NUE) is a sub-class of these events. Conceptually, inferring an obstacle in the design process can be based on detecting a non-usual event. Two approaches could be used to detect an event. One is process-based monitoring and another is activity-based monitoring. The former one observes changes in the information flow of the system. Our experiments showed that, concerning the data needed for the observation of an event and the accessibility of data sources for this purpose, accessing the software execution data in a set of design tools was more complicated than obtaining the data concerning the designer's facial expressions. The latter one observes changes in the behavior of the designer. More specifically, the activity-based monitoring observes the designer's facial expressions. The pattern of changes in the facial expression can be associated with the various types of events in the design process. The state of the art of ML-based recognition of facial expressions offers the opportunity to continuously monitor the designer's behavior. The ML-based recognition approaches can provide sufficient clues to recognize a NUE. To handle the complexity of the computational implementation, the activity-based monitoring was chosen in the process of conceptualization of the ARF.

Proposition 7: *Rule-based reasoning or pattern-based reasoning does not result in significantly different outcomes in the case of non-probabilistic inference.* (RC3)*

Rule-based reasoning infers a solution based on a rule set and the decision conditions specified in the form of Boolean parameters. Pattern-based reasoning infers a solution by finding the best match in the patterns of decision variables stored in the decision table and

considering the valuation of the variables provided by the designer. For the implementation of the dialogue-based obstacle identification (DOI) module, the principle of exact inference was applied to infer a solution based on a set of answers provided by the designer. An exact inference is considered as a sub-class of non-probabilistic inferences. Fundamentally, if a set of conditions in the rule set and the pattern of decision variables are identical, both approaches provide a similar result. This means that both rule-based reasoning and pattern-based reasoning can be applied alternatively for the execution of exact inference. Considering the requirements and constraints (i.e., computational complexity and resource usability), the pattern-based reasoning was preferred for the demonstrative implementation of the ARF.

Proposition 8: *A reference process protocol is the right means for the ARF to provide context-sensitive recommendations for the development of ASRMs.* (RC3)*

The reference process protocol (RPP) was introduced as a crucial constituent of the ARF for generation of process-based recommendations. The RPP models and contextualizes the design process of ASRMs and, as such, represents the system-level knowledge of the ARF. By definition, RPP is a prescriptive instrumental model of the design process or a specific part of the process. In the case of hybrid inference, the RPP is used to generate context-sensitive recommendations. The probabilistic reasoning offers multiple choices for creating design activity flows. At a decision point in the RPP, the decision tree model supports the selection of the most appropriate (usable) method for a particular design entity. The crucial constituent nature of the RPP originates in the fact that hybrid inference cannot be executed in the lack of system knowledge in the application context.

Proposition 9: *The probabilistic relationships need to be included in the reference process protocol to make the active recommender framework capable of offering case-related recommendations.* (RC3)*

The relationships of design entities can be captured in and modelled by the RPP. They make the ARF capable of exploring the possible design flows to complete the design process for a development of ASRMs. Technically, the ARF finds the design activity flow that includes the current design entity by considering the target design entity, if they are in one way or other connected in the RPP. It is also possible that multiple design flows exist that include connection of two considered design entities. Therefore, probabilistic relationships should be included in the RPP to be able to determine the best matching design activity flow. In the computational implementation of the RPP, the relationships of design entities were quantified by the frequency of co- pairwise occurrences of the entities, which were supposed to be used in historical cases. The best design activity flow includes the most popular design entities according to the highest value of their joint distribution probability. Based on this interpretation, the ARF offers case-related recommendations considering the most popular design activity flow.

Proposition 10: *Traceable logical reasoning model is needed for hybrid inferencing.* (RC3)*

Hybrid logical inference was used to infer the most informative segment of the PFM and to rectify the current design action. Furthermore, decision tree classifiers were applied as decision support models for this purpose. The decision tree classifier used selects the most appropriate methods for the intended design actions at any decision points in the reference process protocol. The selection of the usable methods is based on the conditions of the decision criteria that are captured through the dialogue from the designer's answers. However, it was also possible to implement other machine learning-type algorithms (e.g., ANN, KNN, or SVM) for the hybrid inference component, which would probably make the performance of the hybrid logical inference more efficient than the decision tree model. However, the ML-type algorithms are constructed as black box models – that is, their decision process is not logically traceable and explainable. Meanwhile, the decision tree is a traceable logical reasoning model, which, on the one hand, provides a technically equivalent model to black box models, and on the other hand, they offer a monitoring and a better understanding opportunity of how decision is made. It is of a high probability that the designer would like to know and requests the ARF to learn/present how the decision was made. In addition, we argue that it increases the possibility that the recommendation proposed by the ARF is accepted by the designer, if she/he understands the logic of the decision-making process. This way, the quality of the recommendations is increased in term of their usefulness. Thus, it is important that the knowledge acquired by the ARF should be presented in the traceable logical reasoning model.

Proposition 11: *A synthetic validation agent needs to properly model and simulate the decisional behavior of sample designers.* (RC4)*

Current cognitive and computing technologies allow using synthetic software agents for various purposes. In the promotion research, a synthetic software agent has been created to model the decisional behavior of non-accessible designers in the validation study. The programmed agent modelled multiple designers, who variously responded to the offered recommendations. The synthetic software agent was supposed to act like the human designer in the decision-making process. The agent was designed to recognize the appropriateness of recommendations and to model the execution of the proposed design activity flow. Without the on-site participation of a human designer in the validation process, we could acquire synthetic data that reflected the collective decisional behaviors of designers. From the viewpoint of validation, if the agent was not able to properly simulate the decisional behavior of the designers, then a meaningful dataset could not be generated. In order to ensure a proper operation of the agent, we deployed both quantitative analysis (to derive the decision model of the agent) and qualitative analysis (to validate the properness of the model logically). As a result, the agent generated meaningful data, which could be effectively used for the validation of the usefulness of the recommendations.

Proposition 12: *Prognostic reasoning that considers the probabilities of decision options is an effective means for evaluating usefulness of case-related recommendations.* (RC4)*

Prognostic reasoning is an effective means for forecasting probabilities. In our validation study, we applied it in combination with generating case-related recommendations. By definition, the term 'case-related' means the recommendation was generated based on historically-used cases of the concerned design process. Prognostic reasoning was applied to derive indicators of usefulness. The indicator was used not only for the target design sub-task, but it was also applied to similar tasks. The usefulness of the case-related recommendation could be indicated by prognostic reasoning that could consider the probabilities of the decision options. We found that the proposed indicator can be applied effectively, if the forecasting of the probability of the acceptance of the recommendation hint at a likely outcome.

6.3.2 Socially-contextualized propositions

Based on the research work and the implication of the results, the following socially-contextualized propositions have been formulated:

Proposition A: *Design is gradually transferred to systems.**

Proposition B: *Due to the accelerated achievements of the research communities and to the diversification of science, no one knows what the state-of-the-art is.**

Proposition C: *Intelligent systems need to detect logical fallacy by their own intelligence in order to reach the level of human intelligence.**

Proposition D: *Without the support of a predictive conceptual framework, a system designer gets anchored to the first feasible solution, instead of exploring and constructing the most beneficial ones.**

Proposition E: *Developing a proper model for a computational agent mimicking the collective decisional behavior of people is a complicated task, but validation of the decision-making model of one single human agent is even more complicated.**

6.3.3 Self-reflective propositions

As basic thoughts emerged while conducting the promotion research, the following self-reflective propositions have been formulated:

Proposition X: *The democratic value is the most important issue in non-democratic societies.**

Proposition Y: *The process of doing a Ph.D. study is like walking three-steps forward and two and a half steps backward.**

Proposition Z: *No one knows that authentic Dutch food is one of the most delicious meals in the culinary world.**

6.4 Recommendations and future works

6.4.1 Possible short-term research

The research project was conducted for five years of the Ph.D. study. We found new and interesting scientific findings when we changed the points of view with regard to the research problems. Because of the limitations of the available resources and the time constraints, the research topic can be improved upon and short-term follow-up research can be proposed. The possible topics may be as follows:

- A short-term future research can focus on an all-embracing implementation and testing of the complete ARF with the improvement opportunities in recommendation generation, the utilization of reference process protocol, and algorithm modifications.
- The implemented functions can be improved, considering the limitations. More sophisticated algorithms can be considered to make these functions more intelligent and autonomous.
- More aspects of validation may be considered in testing the usability of the ARF with the participation of designers and testing the applicability in other sessions of the design process of ASRMs.
- The theoretical correctness of the RPP could be checked before testing the usability of the ARF. This may extend to (i) the contents of design entities, (ii) the relationships of design entities, and (iii) the prediction variables for training the decision tree model for a particular design entity.
- Instead of using Bayesian networks and probability-based reasoning, an ontology-based approach may also be used for capturing the relationships of the design entities in the RPP. Ontological reasoning can facilitate semantic inferencing when using the RPP. The probabilistic and ontological approaches may be combined with the objective of achieving improvement in the reasoning performance of the ARF.
- Due to the lack of data concerning the designers' decisions, a quantitative validation of the synthetic validation agent (SVA) is a challenge. Research may explore if a surrogate method could be developed.

6.4.2 Possible long-term research

As a prototype of a specialized engineering recommender system, the ARF could be studied from many dimensions in the future. For instance, its implementation as an intelligent and autonomous system is on top of the list of possible long-term research activities. In addition, extending it to support the design process of different ARSMs is also a direction with high potential, which may also include the exploitation of novel affordances. Other concrete proposals are as follows:

- Investigation of computational thinking concerning activity-based monitoring considering the principles of smart self-adaptive cyber-physical systems.

- An analysis of a broader range of factors influencing the decision-making behavior of the designer seems to be a sensible research effort. This could contribute to increasing the usefulness of the recommendations and to improving the quality of the recommendations provided by the ARF.
- Instead of reasoning with the designer's behavior exclusively, augmenting the ARF by CASE tools and monitoring the information flow between the designer and the tools is also a possibility. Eventually, these two event-monitoring approaches can be combined in order to offer the recommendation services in a more effective way.
- Investigation and application of other, not ML-type algorithms, for the runtime development of the RPP.
- Using 5GLs for a fully-fledged implementation of the ARF and combining it with next generation CASE tools.

List of figures

List of tables

List of acronyms

0G-CPSs	zeroth generation cyber-physical systems
1G-CPSs	first generation cyber-physical systems
2G-CPSs	second generation cyber-physical systems
3G-CPSs	third generation cyber-physical systems
4G-CPSs	fourth generation cyber-physical systems
ACG	Advisory Content Generation
AI	Artificial Intelligence
AIRM	Application-Independent Reasoning Mechanisms
APAS	Automated Parking Assist System
ARF	Active Recommender Framework
ASRM	Application-Specific Reasoning Mechanism
BDI	Belief-Desire-Intention
BM25	Best Matching 25 Algorithm
BNs	Bayesian Networks
CAD&D	Computer Aided Detection and Diagnosis
CAD/E	Computer Aided Design and Engineering
CARE	Classify-Asses-Resolve-Enact
CASE	Computer Aided Software Engineering
CIR	Context Information Reference
CMs	Cognitive Maps
CNN	Convolutional Neural Network
CPSs	Cyber-Physical Systems
DA	Design Action
DeM	Decision Modelling
DIR	Design Inclusive Research
DOI	Dialogue-based Obstacle Identifier
DTM	Decision Tree Model
ECA	Event-Condition-Action

EMR	Electronic Medical Record
FCMs	Fuzzy Cognitive Maps
FER	Facial Expression Recognition
FuNN	Fuzzy unsupervised Neural Network
HMM	Hidden Markov Models
ID3	Iterative Dichotomiser 3 algorithm
ISC	Integral System Concept
ISD	Incorrect Subjective Decision
JOD_I	Type I Justified Objective Decision
JOD_II	Type II Justified Objective Decision
JPD	Joint Probability Distribution
KIDS	Knowledge Intensive Data System
KG	Knowledge graph
k-NN	k-Nearest Neighbours Algorithm
LUT	Look-Up Table
MAPE-K	Monitoring-Analysing-Planning-Executing with knowledge
MAC	Memory-Attention-Composition
MES	Manufacturing Execution System
ML	Machine Learning
MM	Meta-Model
NUE	Non-Usual Event
NUE-D	Non-Usual Event Detector
OODA	Observe-Orient-Decide-Act
OWL	Web Ontology Language
PAR	Procedural Abductive Reasoning
PBR	Practice Based Research
PF	Problem Formulation
PFM	Process Flow Model
PRS	Procedural Reasoning System
QE	Quality Examiner
RPC	Reference Protocol Creator
RiDC	Research in Design Context
RG	Recommendation Generation
RPP	Reference Process Protocol
RMD	Reasoning Mechanism Development
ROI	Reference protocol-based Obstacle Identifier
RSs	Recommender Systems

RsP	Reasoning sub-process
S-CPSs	Smart Cyber-Physical Systems
S-CPSoSs	Smart Cyber-Physical Systems of Systems
SDPs	System Development Principles
SEFs	System Engineering Frameworks
SELF DAS	Synthetic, Evolving, Life Form – Dialectic Argument Search
SFM	Spatial Reference Feature
SitM	Situation Modelling
SLR	System-Level Reasoning
SoSs	System of Systems
SVA	Synthetic Validation Agent
SVM	Support Vector Machines
TAM	Timed (design) Action Model
TF/IDF	Term Frequency/Inverse Document Frequency
UML	Unified Modelling Language
USD	Unjustified Subjective Decision
WPE	Working Principle Exploration

Acknowledgements

Without the supports of the supervision team, my research fellows, friends, and family, it is impossible to complete my Ph.D. research by solely myself.

First of all, I would like to express my gratitude to my Promotor, Professor Imre Horváth for giving me an opportunity to work on the specular research topic. Without your advice, encouragement, and exhortation, I could not handle the complicated contents of the thesis and complete this project by myself. Your intensive consultations pushed me to complete this thesis successfully. I learned a lot from the criticized discussions with you. These are the invaluable experiences which I can get from no one else. You are master Yoda for me. It is my honor to be your student. My sincere acknowledgments also go to my daily supervisor, Zoltán for your kindly support. Your guidance and consultations helped me a lot when I got stuck. Thank you for your calm supervision. I will miss the time when we were at Locus pub for drinking beer and having enjoyable talks.

A special gratitude to the committee members, thanks for dedicating the time to read through the thesis and providing me with the positive responses.

Special thanks to our friends at the CPSs group, Gareth, Tima, Yongzhe, Peng and everyone for the good times we shared. Although when we were working, we did not talk much to each other, but when you were not there. I was feeling blues.

Many thanks to my lovely friends, Erk, Tal, Prang, Pump, Baikhoa, Nut, Mel, Milk&Alex, Anouk&Yi Chien, N, Paan, Ratio, Jay, Pond, Fon, Tuptip, Taw, Janet and everyone who shared the enjoyable moments during my five years in Delft. Thanks to P'Top and Naim for our challenging works under the Urban Chitchat project. Thank you everyone I cannot mention all here. I remember all of you when we shared the wonderful moments in the Netherlands. Thank you for hanging out at my house, Ternatestraat 173. When you were here, I was never feeling lonely.

For my family, I would like to dedicate my PhD. title to my dad who did not have a PhD. degree, but everyone usually called him 'Dr. Jin'. You gave me an inspiration to think differently from the traditional social norms. Although you are not here, but I know you are always with me. There is no word to express my gratitude to my mom. Love is not enough for everything you do for me. No matter how crazy I am, you always support me. Million

thanks to my lovely sisters, P'Aon, and Aom, for your safeguard especially the financial support. Special thanks to Brent for your help, for proofreading my thesis, and for your supports. You made a wonderful job. Please take my respect.

Lastly, to P'Amp for everything you do for me. No one can understand me but you. When I was young, beside getting PhD., being homeless is one of my dreams, but you destroy it completely. You are now my home.

about the author

Sirasak Tepjit was born in Yala, a province located in the deep-south of Thailand. In 1994, he moved to Bangkok to study in a Pre-Engineering Technical school. He started his undergraduate study in Production Engineering at King Mongkut's Institute of Technology North Bangkok (KMUTNB) in 1997. Taking five and a half year for the bachelor's degree, he continuted a master's study in Industrial Engineering at the same university. At that time, he was interested in system dynamics modeling, systems thinking, system engineering and management, and complext adaptive systems.

Since obtaining a master's degree in 2006, he spent years as a researcher and participated in several research projects funded by Thai government agencies. He had gained his research experiences in public policy-making, logistics infrastructure planning, and urban planning. In 2012, he went to the UK for studying the second master's degree in Logistics and Supply chain management at University of Portsmouth. After graduation, he went back to Thailand and started his academic career in the Faculty of Engineering and Technology at KMUTNB (Rayong campus). In 2015, he got a scholarship for a Ph.D. study funded by the Thai government. A year later, his Ph.D. journey had been begun at the Faculty of Industrial Design Engineering, Delft University of Technology, the Netherlands. His current research interests are in various topics of system design engineering including design methodologies, knowledge engineering, smart systems, and decision-support system.

Delft, 2022

TU Delft

ISBN 978-9-46-384299-0

Figure 2 System-level architecting

proper representation and
reasoning, which potentially
handle the changes in the design
process, (iv) responsiveness to the
changes in runtime.

Idention of an Active recommender framework

Active recommender framework
(ARF) is a conceptual idea that
allows 'framework' facilitating
a designer to develop
systems by monitoring the
of a user,
understanding