Natural Language Processing for systems engineering
Automatic generation of Systems Modelling Language diagrams

Zhong, Shaohong; Scarinci, Andrea; Cicirello, Alice

DOI
10.1016/j.knosys.2022.110071
Publication date
2023
Document Version
Final published version
Published in
Knowledge-Based Systems

Knowledge-Based Systems 259 (2023) 110071

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Natural Language Processing for systems engineering: Automatic generation of Systems Modelling Language diagrams

Shaohong Zhong [a], Andrea Scarinci [b], Alice Cicirello [a,c,*]

[a] Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, United Kingdom
[b] Uncertainty Quantification Laboratory, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, United States
[c] Depeartment of Engineering Structures, Delft University of Technology, Stevinweg 1, Delft, 2628 CN, The Netherlands

ARTICLE INFO

Article history:
Received 2 August 2022
Received in revised form 21 October 2022
Accepted 22 October 2022
Available online 3 November 2022

Dataset link: https://github.com/ShaohongZ/NLP-for-Systems-Engineering

Keywords:
SysML diagram
NLP
Structure diagram
Requirement diagram
Text-to-diagram
Automated diagrams from text

ABSTRACT

The design of complex engineering systems is an often long and articulated process that highly relies on engineers' expertise and professional judgment. As such, the typical pitfalls of activities involving the human factor often manifest themselves in terms of lack of completeness or exhaustiveness of the analysis, inconsistencies across design choices or documentation, as well as an implicit degree of subjectivity. An approach is proposed to assist systems engineers in the automatic generation of systems diagrams from unstructured natural language text. Natural Language Processing (NLP) techniques are used to extract entities and their relationships from textual resources (e.g., specifications, manuals, technical reports, maintenance reports) available within an organisation, and convert them into Systems Modelling Language (SysML) diagrams, with particular focus on structure and requirement diagrams. The intention is to provide the users with a more standardised, comprehensive and automated starting point onto which subsequently refine and adapt the diagrams according to their needs. The proposed approach is flexible and open-domain. It consists of six steps which leverage open-access tools, and it leads to an automatic generation of SysML diagrams without intermediate modelling requirement, but through the specification of a set of parameters by the user. The applicability and benefits of the proposed approach are shown through six case studies having different textual sources as inputs, and benchmarked against manually defined diagram elements.

1. Introduction

Systems engineering is a crucial process in the design and management of complex systems [1–5]. An integral part of designing and architecting engineering systems involves producing formal documentation to both record and support the development process (e.g. systems specifications [6], codified system diagrams etc.). An open debate within the systems engineering community exists about the notion of *completeness* of specifications (requirements in particular), and means to "check" for it. In this paper, the authors are not concerned with establishing a new paradigm for completeness verification. Instead, the authors aim at facilitating a crucial step in achieving such a goal: the processing of the large amounts of documentation and engineering knowledge from which specifications are drawn. Maintaining document consistency and/or completeness while gathering the necessary information to specify a system's behaviour is both challenging and crucial to avoid unexpected consequences that may result in uncontrolled hazards during the system's operations [7,8]. Errors and omissions in the initial design stages in particular can lead to costly product modifications after manufacturing [8].

Typically, the synthesis of available knowledge into appropriate engineering documentation and formats (e.g. requirements, diagrams) is a manual process. With this comes a certain degree of subjectivity and arbitrariness. In this paper, the authors propose a way of automating this manual process by using Natural Language Processing (NLP) techniques: on the one hand, having machine learning algorithms replacing manual work decreases costs and processing times; on the other hand, systematising the gathering of information and leaving the engineers with a purely supervisory task could provide for better coverage and consistency.

As a testbed for the concept just described, the authors focus on automating the generation of a very commonly used tool in systems design: SysML diagrams. SysML is a graphical modelling language that is a subset of the Unified Modelling Language

* Corresponding author at: Depeartment of Engineering Structures, Delft University of Technology, Stevinweg 1, Delft, 2628 CN, The Netherlands.
E-mail address: a.cicirello@tudelft.nl (A. Cicirello).

https://doi.org/10.1016/j.knosys.2022.110071

(UML) with extensions catered to systems engineering [2,4,5,9–11]. It is able to support the specification, analysis, and design of complex systems such as hardware, software, and facilities [2]. SysML has also gained importance in recent years as a critical enabler of Model-based Systems Engineering [2,3]. Like many other graphical modelling languages it still requires significant manual input [2–4]. To the best of the authors' knowledge, this is the first study that focuses on the automatic generation of SysML diagrams from unstructured natural language text.

The main contributions of the paper are as follows:

- The design of a versatile automated approach to generate SysML diagrams from natural language text documents;
- The combined use of NLP techniques and the lexical databases to facilitate and augment the generation of SysML models;
- The mapping of textual entities (phrases and relationships) to SysML model elements using NLP techniques and heuristic rules.

The rest of this paper is structured as follows: a detailed overview of the current state-of-the-art NLP-based strategies for automating systems engineering processes is provided in Section 2. The proposed approach is presented in Section 3, and the steps for its implementation are summarised in Section 4. The case studies and experimental procedures are then described in Section 5. In Section 6, the results are discussed.

2. Review of state-of-the-art NLP-based techniques for automating system engineering diagram generation

To the best of the authors' knowledge, the automatic generation of requirement diagrams and structure diagrams has not been addressed in the literature yet. Therefore, this section focuses on reviewing the state-of-the-art of three key topics: (i) NLP in System Engineering for the generation of system engineering models; (ii) Automatic generation of Unified Modelling Language (UML) diagrams; (iii) Ontology learning for automatic extraction of concepts, relations, attributes, and hierarchies from text.

2.1. NLP in systems engineering for the generation of system engineering models

NLP techniques have been used in requirements engineering, where the majority of requirements documents and their sources, such as user reviews, are written in natural language [12–15]. Past studies have used NLP for requirement elicitation [16], requirement enrichment [17], requirements tracing [18], requirement classification [19], requirement improvement [20,21], identifying the domain of requirement [22], ambiguity detection [23], generating test cases and use cases [24–26], and detection of low-quality requirements [27]. A more detailed review is done in [12], which classified the function of NLP in requirement engineering as detection, extraction, classification, modelling, tracing and relating, and search and retrieval. Of specific interest here are studies that have attempted to extract requirements and construct models from natural language text, which are briefly reviewed in what follows.

Recent works, used NLP for mining concepts from a mix of textual requirement assets such as documentation and manuals [28], product brochures [21], online reviews [13], app descriptions and reviews [15]. However, these works rely on the user to specify relations between identified concepts and to manually construct the system engineering models [21], even when the goal is to generate structure features diagrams [21].

The state-of-the-art approaches for the generation of systems engineering models from natural language text inputs are based on the specification of a set of heuristics rules and by predefining either a set of association phrases [29], or a mix of manually specified sentence patterns [14,30], or a predefined list of goal-specific keywords and syntactic patterns [31]. Other studies have also used NLP techniques such as dependency parsing and coreference resolution to identify type dependencies. The hierarchy was then identified using predefined rules or classifiers [32,33]. However, these approaches are limited by their reliance on heuristics rules [32]. Additionally, they also tended to put strict constraints on the forms of inputs, either specifying complex syntactic rules or restricting inputs to be of a certain format, for example, requirement specification documents [14,29–31]. Such constraints limit the flexibility of the approaches to adapt to the variety of textual assets available for a systems engineer and require a significant amount of prior work to produce structured requirement documents for processing.

2.2. Modelling in SysML and UML

Unlike requirement diagrams, which are unique to the SysML profile, structure diagrams in SysML bear a close resemblance to UML diagrams such as class diagrams and composite structure diagrams [2,9,11]. Thus, past studies aimed at generating UML diagrams from natural language text are also reviewed.

In [34], the authors proposed a method to automatically generate UML class diagrams from natural language requirements by employing a Recursive Object Model (ROM) as the intermediate step. The ROM is a graphical language that treats each word and each sentence as objects and assigns semantic relations from a predefined set to these objects using lexical and syntactic rules [35]. The ROM model was then traversed and converted to a class diagram using the noun objects and their relations [35]. Additionally, in [36], the authors used PoS tagging and semantic role labelling combined with manually defined rules to generate class diagrams from requirement specifications. However, similar to past studies on generating requirement models, these studies rely on using structure requirement documents of specific formats as inputs and are thus limited in their flexibility [34,36]. In contrast, in [37], the authors proposed a method to automatically generate UML class diagrams from natural language stakeholder requests. They used a set of predefined syntactic rules to decompose the inputs into simple sentences, then parsed the resulting sentences and identified classes and relations using a set of heuristic rules [37]. However, the reliance on predefined rules on syntactic features to classify relations also limit the wider applicability of this approach.

2.3. Ontology learning

Ontology learning refers to the automatic extraction of 'concepts, relations, attributes, and hierarchies' from text [38,39]. State-of-the-art methods to automatically extract ontology models from natural language text include Text2Onto [40], OntoGain [41], a graph-based algorithm [42], and CRCTOL [43]. Ontology Learning techniques could be used to enable the automatic generation of SysML diagrams. This is because, similarly to Ontology Learning, System Engineering modelling approaches employ an object-centred representation of the target domain and use similar constructs such as classes/blocks and relations [44]. However, as highlighted in [44], the techniques in ontology generation need to be adapted to suit systems engineering. For example, the primary source for generating an informative diagram is a limited corpus, instead of relying on the web. Moreover, the extracted structured information has to be classified into different types of SysML diagrams [9].

2.4. An approach to overcome the limitations of the state-of-the-art

Based on the above review, it has been identified the need for the development of an automatic approach to generate SysML diagrams from unstructured natural language text. The proposed approach is based on natural language processing techniques and the semantic web, and goes beyond the state-of-the-art by tackling the following research questions:

- How to extract information for developing system engineering models without specifying restrictive syntactic rules valid only for a specific domain?
- How to identify hierarchies with minimal reliance on heuristic rules and predefined patterns?
- How to automate the generation of different types of SysML diagrams with no human intervention or intermediate modelling from a limited corpus of textual resources?

3. Automatic generation of systems modelling language diagrams

The proposed approach focuses on automatically generating a subset of structure and requirements diagrams in SysML from a corpus of natural language text documents. Specifically, this paper focuses on Block Definition Diagrams (BDD), Internal Block Diagrams (IBD), and Requirement Diagrams (REQ) [9]. The basic elements of these three diagrams include blocks and their relationships. A block is an elemental modelling construct in SysML that represents both real entities, such as physical objects, and abstract entities, such as concepts [2]. Relationships between blocks in SysML can be further classified into categories such as association or generalisation for BDD, and trace or containment for REQ [2,9–11].

To generate these diagram elements, a parallel architecture is proposed to extract two kinds of textual entities from natural language text: (1) key phrases, where a key phrase is defined as a list of one or more words that represent an important entity described in the text. (2) key relationships between the key phrases, where a key relationship is defined as one that links two key phrases. The key phrases and key relationships are used to generate blocks and the relationships between blocks, respectively. The generated blocks and their relationships are then organised into desired SysML diagrams.

The procedures for automatic diagram generation are summarised in Fig. 1a and it consists of six steps:

1. The first step is the manual selection and upload of a corpus of text documents (Section 3.1).
2. The raw texts selected are then used as inputs for the key noun extraction (Section 3.2).
3. The raw texts selected are used also as inputs relation extraction (Section 3.3).
4. Steps 2 and 3 result in a collection of key nouns and a collection of relations, respectively. The two collections are then used to generate the list of key phrases and key relations (Section 3.4).
5. The list of key phrases and key relations are subsequently used to generate corresponding SysML model elements and augmented according to the required diagram type (Section 3.5).
6. Finally, the generated SysML model elements are then organised and plotted to the corresponding SysML profile (Section 3.6).

An illustrative example is provided in Fig. 1b. Each step is discussed in detail in what follows.

3.1. Raw texts selection

The algorithm takes as input a manually selected corpus of text documents. These documents do not need to be structured or follow a predefined writing style like requirement documents, and the proposed approach can be applied to documents typically available to systems engineers, for example manuals [2,4]. Additionally, the corpus should be of sufficient size. Providing a universal standard for the sufficient size of textual resources, covering different domains and writing styles, is a challenging task that goes beyond the present work. Nonetheless, based on the experiments carried out by the authors (with the most salient ones reported in Section 5), it is recommended that for the successful generation of diagrams, the document count should be larger than 100 and the average word count per document should be larger than 500. A corpus can also be obtained by splitting one large document, for example one manual book, into chapters and sections, as illustrated in the case study on UK government reports and a Windows manual in Section 5.

3.2. Key nouns extraction

This step of the approach aims to select key nouns, which are nouns that represent important entities and that can serve as constituents of the key phrases. The key nouns are automatically extracted from the corpus by employing preprocessing and term frequency–inverse document frequency (tf–idf) techniques [45].

3.2.1. Preprocessing

Preprocessing is needed to remove grammatical features in the text to support noun extraction [45]. In this study, a combination of preprocessing methods is applied to the text inputs:

Tokenisation: Tokenisation breaks down large textual content, such as paragraphs or documents, into smaller chunks [45]. In this study, the Natural Language Toolkit (NLTK) is used to tokenise each input text document into individual sentences and then tokenise each sentence into individual words. In particular, the adapted Punkt sentence segmenter for sentence tokenisation [46] and the Penn Treebank Tokeniser for word tokenisation [47] available in NLTK are used. In this work, only one-word unigrams are considered.

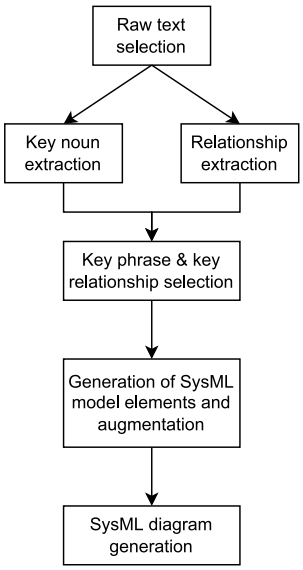
Part-of-speech (PoS) tagging: The NLTK package is used to classify each word of a sentence into different lexical categories. The PoS tagger assigns a PoS tag to each word based on its context sentence [45]. The Greedy Averaged Perceptron tagger and the Penn Treebank tagset are used [47]. Examples of PoS tags include present tense verbs ('VBP') and adjectives ('JJ'). After the PoS tag for each word is assigned, the nouns are used for further processing and words of other types are removed.

Lemmatisation: The individual nouns are further converted to lower case and reduced to their root forms through the NLTK WordNet Lemmatiser, e.g. from 'sensors' to 'sensor'. The WordNet Lemmatiser employs a combination of predefined rules and dictionary search in lemmatisation [47].

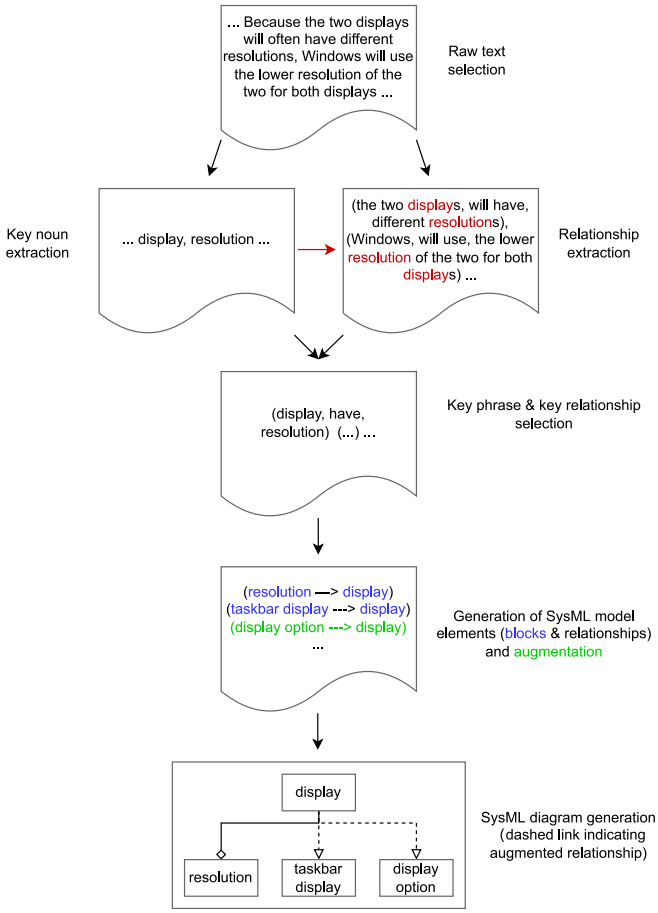
Stop word removal: As the last step, all stop words that are still in the text are removed, using the NLTK list of English stop words as a Ref. [47].

3.2.2. Evaluation of term frequency–inverse document frequency

The term frequency–inverse document frequency (tf–idf) metric is employed to evaluate the domain relevance of each noun after preprocessing [45,48,49]. Note that after the preprocessing steps, the documents contain only nouns. Tf-idf treats each word as unigram, and word orders or document orders are not considered. The tf–idf weighting w of a word t, in a document



(a) Steps for automatic SysML diagram generation



(b) Illustration of automated diagram generation steps using an excerpt from a Windows manual as example

Fig. 1. Procedures and examples for automatic SysML diagram generation.

d belonging to a corpus of documents c, is a value computed as [45,48,49]

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_{t,c} \tag{1}$$

$\mathrm{tf}_{t,d}$ is given by

$$\mathrm{tf}_{t,d} = \log_{10}(count_{t,d} + 1) \tag{2}$$

where $count_{t,d}$ is the total count of a word t in a given document d, and the inverse document frequency $\mathrm{idf}_{t,c}$ is given by [45,49]

$$\mathrm{idf}_{t,c} = \log_{10}(N_c/(1 + \mathrm{df}_{t,c})) + 1 \tag{3}$$

where N_c is the total number of documents in the corpus c, and the document frequency $\mathrm{df}_{t,c}$ is the number of documents in the corpus c that contain the word t [45]. In this study, the tf–idf weighting is normalised to the range $0 \leq w_{t,d} \leq 1$ by dividing by the largest tf–idf weighting in a document. The inclusion of inverse document frequency discriminates against words that are common across the corpus [45,49], which are assumed to be less representative of the document's content, analogous to a set of domain-specific frequency-dependent stop words. The weighting $w_{t,d}$ obtained for each word of a document is used to identify the collection of key nouns to be retained. In particular, a tf–idf weighting threshold σ_{tf-idf} is specified by the user (usually in terms of a user-defined percentile), and nouns with tf–idf weighting higher than the threshold are added to the collection of key nouns.

3.3. Relationship extraction

The input text is first tokenised into sentences using the NLTK package. After sentence tokenisation, open information extraction techniques are used to extract relationships from individual sentences [45,50].

3.3.1. Open Information Extraction (IE)

In this study, the OpenIE toolbox is used for relationship extraction [51–54]. The toolbox assigns a confidence value to each extracted relation, and in the proposed approach, a confidence threshold $\sigma_{relationship}$ between 0 and 1 is manually defined to preliminarily filter the extracted relationships.

For each sentence, the toolbox generates a set of possible relations, where each relation is a list of textual phrases $r = (p_1, p_2, p_3, p_4, \ldots, p_{N_r})$. In an extracted relation r, p_2 is the relation phrase, p_1 and p_3 are akin to subject and object phrases, and p_4 to p_{N_r} are secondary argument phrases, sometimes appearing in long sentences, where N_r is the number of phrases in an extracted relationship [50]. The OpenIE toolbox is implemented as a combination of four methods [51–54], which are briefly explained:

Semantic Role Labelling (SRL) based IE SRL consists of detecting semantic arguments and their roles associated with a verb in a sentence. For example, given the sentence 'I ordered a cake', SRL identifies 'I' and 'cake' as arguments for the verb 'order', where 'I' is the agent and 'cake' is the patient. This component of the OpenIE tool is based on the SRL system by [55,56].

Relational noun based IE Relational noun based IE identifies relations that are mediated by nouns. For example, given the phrase 'Rowing Club President James', it extracts the relation (James, be President of, Rowing Club). This is implemented by encoding a predefined set of relational nouns and nominal patterns in the OpenIE tool [52].

Numerical IE Numerical IE identifies the numerical relations in a sentence. For example, given the sentence 'The company has 100,000 employees', it extracts the relation (The company; has number of employees; 100,000). This is implemented by a combination of machine-learned patterns and customisations specific to numerical relations in the OpenIE tool [53].

Coordination analyser This method is used to split conjunctive sentences. It first uses a dependency parser to extract candidate conjuncts, and then score their coherence based on the Berkeley Language Model [57]. Additional linguistic constraints are also imposed to improve selection results [54].

3.4. Key phrases and key relationships selection

To obtain the key phrases and key relationships useful for generating SysML model elements, a three-step selection mechanism is used. The first step is to select and refine candidate key phrases using the key nouns obtained in Section 3.2. The second step is based on scoring the candidate phrases based on a combination of metrics to obtain key phrases [45,58,59]. The third step is to select key relationships based on the selected key phrases.

3.4.1. Selecting candidate key phrases

For simplicity, only subject and object phrases (p_1 and p_3) of the extracted relationships are used for identifying the key phrase candidates. The phrases are first preprocessed using procedures described in Section 3.2.1. This results in tokenised phrases where $p = (t_1, t_2, \ldots, t_{N_p})$, and each phrase comprises only nouns in root forms; N_p refers to the number of nouns in a phrase. Then, an intersection is taken between each processed phrase and the list of key nouns. Nouns in these phrases that do not map to any key nouns are removed. The user is further able to determine the specificity of the key phrases by setting a parameter L_{phrase} for the maximum number of nouns in one key phrase, where $length(p) \leq L_{phrase}$. If the number of nouns in a candidate key phrase exceeds L_{phrase}, the nouns are ranked according to tf–idf values and the top L_{phrase} key nouns are kept. This results in a collection of candidate key phrases for further processing.

3.4.2. Key phrase selection based on tf–idf, WordNet depth, and phrase frequency

The candidate key phrases are further selected by using a metric based on the frequency of the phrase, average tf–idf weighting, and average WordNet score [58,59].

The count of a phrase, $count_{p,d}$, refers to the total count of a phrase p in the 'bag of phrases' of a document d, normalised with respect to the most frequent phrase in the document. This is used to account for the importance of each phrase. Only the candidate key phrases that are outputs of Section 3.4.1 are considered.

The WordNet depth h' of a word t refers to its semantic depth assigned by WordNet based on its synsets, and is used to account for semantic meaning when evaluating a phrase [58,59]. A general term such as 'entity' would be assigned a small depth value, whereas a more specific term such as 'pancreas' would be assigned a high depth value. The WordNet is a large lexical database of English, where the meanings of each word are represented as a cognitive synonym set called synsets [58,59]. The LESK algorithm is used to identify the most relevant synset for words in the input document based on the document as context [60]. For simplicity, it is assumed that each word in one document has only one synset. The WordNet depth of each word is normalised to range $0 \leq h' \leq 1$ by dividing by the largest depth value in a document, and the one complement of the normalised depth $h = 1 - h'$ is used as the WordNet score.

As both tf–idf weighting w (already introduced in Section 3.2.2) and the WordNet score h are assigned to individual words t, an average value is taken for each candidate key phrase. Hence, for each candidate key phrase p, containing N_p number of

nouns, from a document d, a score λ is computed by using the following formula:

$$\lambda_{p,d} = \sum_{i}^{N_p} w_{t_i,d}/N_p + \sum_{i}^{N_p} h_{t_i,d}/N_p + \text{count}_{p,d} \tag{4}$$

The first term computes the average tf–idf weighting, the second term computes the average WordNet depth, and the last term refers to the normalised count of the phrase. A score threshold σ_p can then be specified by the user, where candidates with scores higher than the threshold are selected as key phrases. It is worth mentioning that when documents contain many broken sentences (as in the document 7 of Table 2), many key phrases might not be found. Since this is an important step in the procedure, in order to support the engineers using the proposed tool, it is recommended to setup an alarm to warn the user when the tool fails to extract relations from more than 50% of the sentence tokens in one document.

3.4.3. Key relationship selection

Relationships are selected as key relationships only when both their subject and object phrases are key phrases. This is to prevent open-ended relationships and 'floating' blocks that do not connect to any other block in SysML diagrams.

3.5. Mapping and augmentation

The selected key phrases and key relationships are *textual*. For plotting SysML diagrams, they need to be first mapped to SysML model elements. The authors focus on the Block Definition Diagrams (BDD), Internal Block Diagrams (IBD), and Requirement Diagrams (REQ) in this study. The key phrases are used to generate blocks B, which form a fundamental unit of SysML [2]. The textual relationships are first classified into different categories according to the SysML diagram type, and then used to generate the different types of relationships R between blocks. The textual relationships are also used to generate requirements that the blocks satisfy.

To make diagrams more complete in both structure and semantics, the textual phrases and relationships are also augmented according to the required SysML diagram type. The augmentations are used to generate additional blocks and relationships for the diagrams.

3.5.1. Block definition diagrams

Block Definition Diagrams are the most common diagrams used in SysML and are used to define the types of elements of other diagrams [2,9]. Blocks in BDD can have many features such as parts, references, and operations, whereas relations in BDD can belong to categories such as associations, generalisations, and dependencies [2,9]. For simplicity, the authors choose to focus on the composite association, generalisation, and reference association, and on identifying the operations features of blocks in BDD. The composite association relationship represents a structural composition where one block is part of another block [2,9]. On the other hand, generalisation relationship means an inheritance relationship, where one block is the generalisation of the other, more specialised, block [2,9]. Additionally, reference association conveys a connection between blocks where one block can access another [2,9].

Relationship mapping for BDD

The steps used to map textual relationships to inter-block relationships in BDD are:

1. Identification of operation: for every subject phrase p_1 in the textual relationships, the relation phrase p_2 between the subject and its object phrase is considered as the operation of the block B_{p1} defined by p_1.
2. Classification based on relation phrase: the textual relationships are first classified according to the meaning of the relation phrase p_2 in the relationship. A pre-selected list of WordNet synsets that define composite associations are used, for example 'include.v.01'. If the relation phrase's synset belongs to the predefined list, then the relation is mapped to a composite association with the corresponding hierarchy.
3. Classification based on overlap: the rest of the textual relationships are then classified by the string representations of the subject and object phrases (p_1, p_3). If the string representation of one phrase is contained in the other phrase in the relation (e.g. the phrase 'prediction model' includes the single word phrase 'prediction' in string representation), then the relationship is mapped to a generalisation relationship, with the block defined by the shorter phrase being the generalisation (the other being specialisation).
4. Classification based on score: the remaining textual relations are then classified according to the score of the subject and the object phrase according to Eq. (4). If the difference in score is above a user-defined threshold $\sigma_{rel-difference}$, the relation is mapped to a composite relationship where blocks defined by the lower-scored phrase is considered a part of the block defined by the higher-scored phrase.
5. Classification of remaining relations: the remaining textual relations which are not yet classified are then mapped to reference associations.

Augmentation for BDD

The steps used to augment the list of textual phrases and relationships for plotting BDDs are:

1. Identification of top-level phrases: The key phrases whose corresponding blocks do not form sub-blocks of another are first compiled to a set P, where a sub-block is defined as the block at the part end of a composite relationship, or at the specialised end of a generalisation relationship. These phrases are used for the next abstraction step.
2. Abstraction: Algorithm 1 is used to iteratively abstract higher-level phrases from the top-level phrases based on a per-word score γ that is a combination of the tf–idf weighting w and WordNet score h of a word t in a document d.

Algorithm 1: Phrase Abstraction

```
Identify set of top-level phrases P
for phrase p in P do
    if length of phrase N_p > 1 then
        Initialise score set Γ
        for noun t in p do
            γ_{t,d} = w_{t,d} + h_{t,d}
            Γ_{p,d} = Γ_{p,d} ∪ {γ_{t,d}}
        end
        sort(Γ_{p,d})
        p_abstract = p \ t, where γ_{t,d} is smallest in Γ_{p,d}
        P = P ∪ {p_abstract}
    end
end
```

Additionally, generalisation relationships are assigned to the blocks defined by the top-level phrases and their abstractions.

3. Relationship augmentation: After the abstraction step, all of the resultant sets of top-level phrases are now unigrams. The WordNet is then used to identify hypernym/hyponym (corresponding to generalisation) and holonym/meronym (corresponding to composite association) relationships between these phrases and corresponding relationships are assigned between their blocks [58,59].

4. Phrase augmentation: As the final step, the lowest common hypernyms between the phrases that remain at the top level are found using WordNet [58,59]. The blocks defined by these remaining top-level phrases are assigned a generalisation relationship with blocks defined by their hypernyms.

3.5.2. Internal block diagrams

Internal Block Diagrams are used to specify the internal structures of blocks, and display the parts and references of a specific block as well as the connections between its parts and Refs. [2,9]. In this study, the parts of a block are defined as the block's composites, and the references of a block are defined as other blocks connected to it through the reference association. A connection means that two blocks are able to access each other in an operational system [2,9]. To leverage the connection between IBD diagrams and BDD diagrams, the augmented list of textual relationships and phrases from the BDD are used as inputs to the IBD algorithm.

Relationship mapping for IBD

The steps to map textual relationships to inter-block relationships for IBD are:

1. Parent block generation: The user can first choose to specify a parent block to draw the IBD for. Then, all blocks that are sub-blocks or sub-sub-blocks of the user-specified block are selected. If no parent block is manually selected, IBDs will be drawn for all blocks.
2. Relationship selection: The textual relationships whose subject or object phrases correspond to a sub-block of the user-specified block are selected, except in cases where the phrase is in the lower hierarchy in the textual relationship.
3. Connection classification: For each of the selected textual relationship, if the relation phrase is not empty, then blocks defined by its subject and object phrases are assigned a port connection between each other.

Augmentation for IBD

The steps used to augment the list of textual phrases and relationships for plotting IBDs are:

1. Phrase augmentation: Pairs of key phrases that share a non-empty intersection (i.e. include one or more identical nouns) are identified and are added to the list of phrases for IBD, except when the intersection corresponds to the user-specified block.
2. Relationship augmentation: After the phrase augmentation step, a generalisation relationship is assigned to the block defined by the intersection and the intersection's parent blocks, with the intersection block at the generalised end. The new relationships are added to the list of IBD relationships.

Importantly, the augmentation mechanism used in IBD is different to the abstraction mechanism in BDD. This is done as a redundancy measure to ensure the completeness of the plotted diagrams. For example, a phrase 'prediction mode' may be only abstracted to 'prediction' in the BDD step. However, in the IBD step, it could be additionally abstracted to 'model' if any of the other selected key phrases also include the word 'model'.

3.5.3. Requirement diagram

Requirement Diagrams are a unique feature of SysML diagrams that represent the requirements of the system and its components [2,9]. Relationships relevant to requirements in SysML include containment, trace, derive, refine, satisfy, and verify [2,9]. For simplicity, the authors focus on the satisfy relationships and the trace relationships between the requirements, which represents a weak dependency where a change at one end may result in the need to modify the requirement at the other end [2,9].

Mapping for REQ

The following steps are used to map textual relationships to requirements and requirement relationships for REQ diagrams.

1. Requirement identification: For each of the key relationships, if the relation phrase is not empty, then its corresponding raw relationship output from OpenIE is considered a requirement.
2. The requirements are assumed to be satisfied by their subject phrase blocks, and their requirement blocks are named after the subject phrases. There can be multiple requirements with the same subject phrase, and these are grouped together in the same requirement block.
3. Relationship identification: A trace relationship is assigned to requirements whose corresponding relationships share the same subject phrases or object phrases, or have subject phrases equal to object phrases of other relationships, and vice versa.

Relationship augmentation for REQ

The WordNet is used to identify any hypernym, hyponym, entailment, and causes relationship between the relation phrases of relationships that have been identified as a requirement [58,59]. If a relationship is identified, the corresponding pair of requirements are considered to have a trace relationship

3.6. Diagram generation

Prior to plotting the diagram, the user can choose to select a parent block for plotting the desired diagram. For IBD, this would be the parent block used to select sub-blocks and sub-sub-blocks. For BDD, all blocks at a lower hierarchy than the parent block (e.g. the parent block's sub-blocks and their sub-blocks, etc.) will be iteratively selected, and the resultant selected blocks and their relationships will be plotted instead of the full diagram comprising all generated blocks. For REQ, all requirements that contain the phrase corresponding to the selected block, and the relationships between these requirements, are extracted for plotting.

An open-source diagram generation tool (PlantUML [61,62]) is used to plot the generated blocks and their relationships. The augmented blocks and relationships are plotted with dotted lines to indicate that these do not directly map to phrases and relationships in the input text documents. The diagram generation tool uses GraphViz as its graphical engine [63].

4. Steps for the implementation of the proposed approach

The following steps are performed for the generation of SysML diagrams using the proposed approach:

1. Select the corpus of textual materials;
2. Select the document to draw the SysML diagram for;
3. Set threshold values for five hyperparameters
   (a) $0 < \sigma_{tf-idf} < 1$ for key noun selection. This sets the tf–idf threshold for a noun to be considered a key noun.
   (b) $0 < \sigma_{relationship} < 1$ for relationship extraction. This sets the confidence threshold for a relation extracted by OpenIE and can be used to remove an excessive number of duplicated relationships.
   (c) $0 < \sigma_p < 3$ for key phrase selection. This sets the minimum score for a phrase to be considered a key phrase via the proposed formula in Eq. (4).

(d) $0 < \sigma_{rel-difference} < 3$ for composite relationship classification. This sets the minimum score difference for connected phrases to be mapped to composite relationships.
(e) $L_{phrase} > 0$ for length of phrases. This sets the maximum number of words in a phrase;

4. Select the type of SysML diagram to generate (optional);
5. Select the phrase to define the parent block in the SysML diagram (optional);
6. Run the end-to-end six-steps approach as described in Section 3 to automatically generate SysML diagrams, which will generate the following outputs in sequence:
   (a) List of key nouns;
   (b) List of extracted relationships;
   (c) List of key phrases and key relationships;
   (d) Mapping of key textual relationships to SysML relationships;
   (e) Required diagrams.
7. Evaluate the quality of the results through the following steps:
   (a) Evaluation of key phrase selection;
   (b) Evaluation of key relation mapping;
   (c) Evaluation of diagram generation.

4.1. Quantification of the success of phrase extraction

To evaluate the success in key phrase extraction, manually selected lists of key phrases based on OpenIE extractions are created for each of the selected documents and used as the ground truth. The automatically extracted lists of phrases are then evaluated by precision and recall. Precision is defined as the percentage of extracted key phrases that match a phrase in the ground truth. Recall refers to the percentage of ground truth phrases that match a phrase in the extracted list. As the key relationships are extracted together with the key phrases, the key relationships are not evaluated separately because it is strongly coupled with the key phrases.

4.2. Quantification of the success of relationship mapping

To evaluate the effectiveness of the approach in mapping textual relationships to SysML relationships, the ground truth for the type of relationship between blocks defined by the subjects and objects is defined manually. This is used as a benchmark to evaluate the results from the relationship mapping algorithm. The algorithm is evaluated for its accuracy in determining the type of SysML relationship (e.g. composite association, generalisation or reference association) and the relative hierarchy between the blocks defined by the subject and the object phrase in the textual relationship (e.g. whether block defined by the subject phrase is a composite of that defined by the object, or vice versa).

4.3. Hyperparameter selection

The following values for the hyperparameters are suggested, and applied to the six case studies investigated:

1. $\sigma_{tf-idf} = 0$. This means that all the nouns in the input text are treated as key nouns. This is done to illustrate the effect of selection in the subsequent steps. However, it is envisioned that the parameter can be adjusted according to the user's needs;
2. $\sigma_{relationship} = 0.5$. This is to filter out relationships with low confidence of being valid relationships.
3. $\sigma_p = 0.6$. This implies most phrases are selected as key phrases, to ensure the completeness of results.
4. $\sigma_{rel-difference} = 0.5$. This ensures that only connected phrases with a sufficient score difference are mapped to composite relationships.
5. $L_{phrase} = 3$. This sets the maximum length of a phrase to be three nouns.

4.4. User interface

An illustrative example of the user interface for the proposed tool is shown in Fig. 2. The main panel features the list of uploaded documents on the left. On the top, the users are able to vary the hyperparameters according to need, and generate the desired SysML diagram at the bottom, below the hyperparameters.

5. Experiments

Six case studies are investigated to evaluate the effectiveness of the proposed approach in automatically generating SysML diagrams from texts. The details of the datasets used, the experimental procedure, the evaluation procedure, and the choice of hyperparameters are discussed in this section. The results from the extraction of key phrases and classification of relationships are validated against manually constructed benchmarks. Additionally, the generated diagrams are also compared against manually designed ground truth diagrams.

It is worth emphasising here that the aim of the proposed approach is not to replace systems engineers but rather to aid them in gaining an overview of the system. Therefore, the goal is not a perfect extraction of phrases or relationships. Rather, it is to provide an overview of the selected document and system through the profiles in SysML diagrams.

5.1. Datasets

The applicability of the proposed approach is tested on six case studies to cover different dataset sizes, domains, and writing styles:

(1) Patent descriptions from the European Patent Office (EPO) [64]. Patent files in English are downloaded from the EPO public database in txt format. Each patent includes sections such as patent name, abstract, and patent description. The patent files are first parsed to remove constructs such as HTML tags and figure references. Then, only the patent description section is used as textual input to the algorithm. Each description is treated as a separate document, and 345 separate patent description documents are obtained. The patents cover areas such as data transmission devices, devices for drug admission, and bioinformatics.

(2) Windows manual. A digital book named Windows 10 Field Guide in pdf format is used [65]. The book is split into 28 sections according to the chapter, and converted to individual txt files with images removed. Each file is treated as a separate document. The book is a manual for the Microsoft operating system Windows 10 and describes its different functions and applications, for example Skype and Maps.

(3) UK government report. A series of UK government report on the Future of Manufacturing is downloaded as pdf files [66]. A total of 37 reports are downloaded, and each of them is further split into different chapters, with cover pages, acknowledgements, content pages and references removed. This results in a total of 284 chapters. Each chapter is converted to txt format and treated as a separate document. The contents cover areas

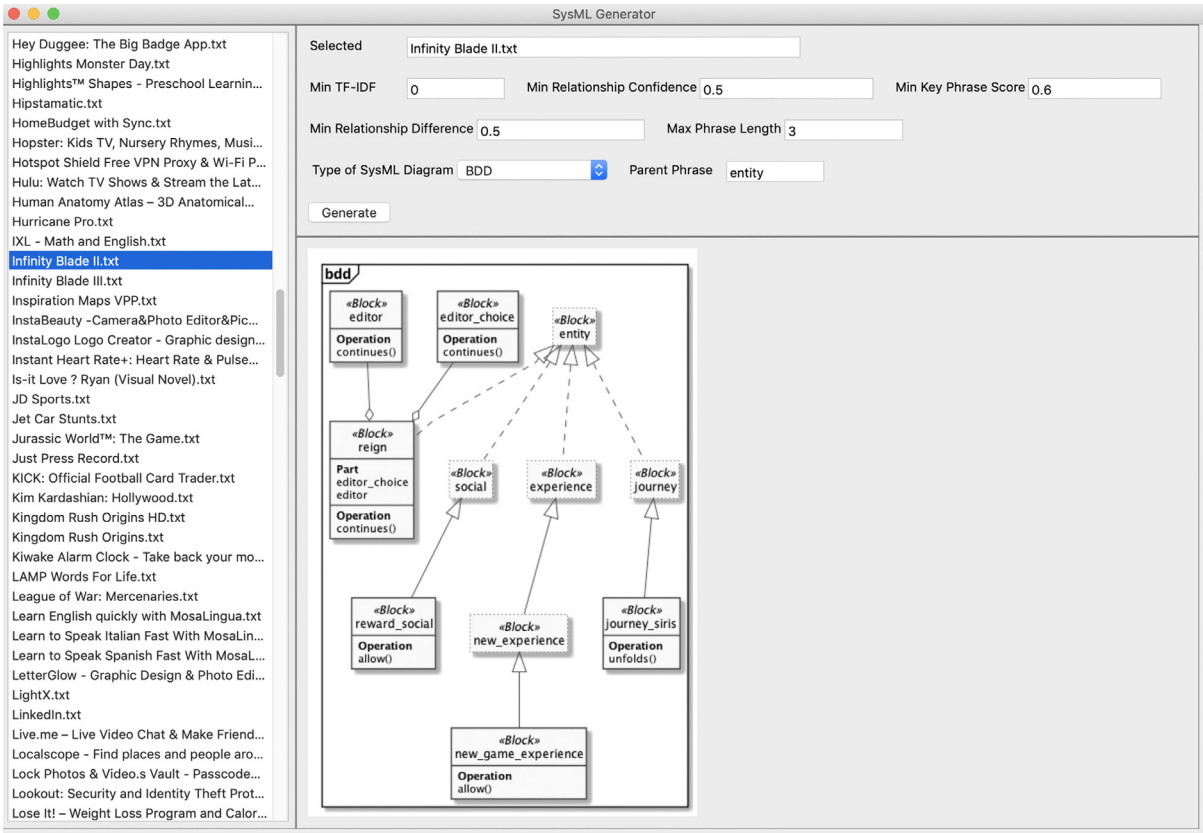


Fig. 2. An example of the user interface for the proposed tool.

such as technology trends in manufacturing and investment in innovation.

(4) App descriptions. App descriptions from the Apple App Store are downloaded from an online public dataset [67]. A total of 346 app descriptions in txt format with a file size larger than 4 KB are selected from a dataset of 4075 app descriptions. This is done to ensure that each document has sufficient length. Each description is treated as a separate document. The app descriptions included apps of different genres and included contents such as user reviews and feature descriptions.

(5) Research publications from IEEE. 285 papers are downloaded using IEEE Xplore [68]. The search criteria were to have the words 'design' and 'manipulator' in the title of the paper and the range was from 2015 to 2021. The papers were downloaded as pdfs and converted to txt files. Each paper is treated as a separate document. The papers covered areas in the design of robotic manipulators, for example mechanics and control.

(6) Description of countries on Wikipedia. 193 English Wikipedia pages about countries are parsed from the parent Wikipedia page *Member states of the United Nations* [69]. Each page is saved as a separate txt file and all formatting and images were removed. Each page is also treated as a separate document. The descriptions include each country's economy, geography, government, etc.

Details of each dataset are summarised in the table below. Two documents are selected from each of the datasets for demonstration.[1] The documents are selected based on their word counts being close to the average word count of the dataset. Additionally, one block is selected for each document as the parent block, and the phrases that correspond to the parent blocks are listed in Table 1. The use of these phrases is explained in Section 3.6.

[1] The demonstration documents are available at https://github.com/ShaohongZ/NLP-for-Systems-Engineering

6. Results

6.1. Key phrase extraction

The results from the key phrase extraction are detailed in Table 2. It can be seen that the number of phrases extracted from a document is around the same order of magnitude as the number of unique nouns in the document. The possible reason why the number of phrases tends to be less than the number of nouns, except for one case, is that many of the phrases are in descriptive sentences or sentences with pronouns. For example, in a descriptive sentence such as 'The device is expensive', the candidate phrase 'device' will not be considered as a key phrase as no relationship can be extracted from the sentence. Additionally, in sentences that involve pronouns such as 'You can turn on the computer' and 'It is used to predict motion', the candidate phrase 'computer' or 'motion' will not be selected because detecting pronoun references is out of the scope of this paper. Requiring all key phrases to be related to at least one other phrase is useful because it avoids the case of 'floating blocks' where a given block is not connected to any other block in the SysML diagram. Such floating blocks are not conducive to helping the user understand the dependencies within the system or the hierarchies between the different blocks within the system, and may confuse the user.

As can be seen from Table 2, the precision and recall rates indicate that the algorithm has successfully extracted key phrases from the document. The algorithm is especially useful with documents that tend to be more carefully written, as can be seen from the higher success rates in extracting phrases from patent descriptions. The failure case of key phrases that are not identified includes those in descriptive sentences, those connected to pronouns, and incomplete sentences. Incomplete sentences are probably the main reason for the low recall in extracting key

Table 1
Details of datasets, selected documents, and selected phrases.

	Doc. no.	Word count	Selected phrase
EPO patent		8370 (average)	
A state estimator	1	8418	Prediction
A body fluid leakage detection aqueous composition	2	8343	Starch
Windows manual		3027 (average)	
Devices	3	3061	Display
Maps	4	2859	Map
UK government report		1982 (average)	
Knowledge spillover — knowledge sources manufacturing Part5	5	1993	Intangible asset
De-industrialisation and balance-of payments Part3	6	1958	Manufacturing
App description		559 (average)	
Infinity Blade II	7	559	Entity
Ebates: Cash Back, Coupons & Rebate Shopping App	8	559	Entity
Research paper		3414 (average)	
Design of a weight-compensated and coupled tendon-driven articulated long-reach manipulator	9	3430	Actuator
Multi criteria design of a spherical 3-DoF parallel manipulator for optimal dynamic performance	10	3413	Optimisation
Wikipages		9830 (average)	
Bhutan	11	9867	Industry
India	12	9811	Religion

Table 2
Results from key phrase extraction.

Document no.	Word Count	Unique nouns	No. of key phrases	Precision %	Recall %
1	8418	345	401	80.8	76.8
2	8343	430	351	91.5	77.3
3	3061	214	141	70.2	57.9
4	2859	193	119	66.4	49.7
5	1993	214	157	55.4	81.3
6	1958	223	140	70.7	71.2
7	559	122	12	66.7	34.8
8	559	105	33	84.8	63.6
9	3430	216	131	61.8	54.0
10	3413	367	211	81.0	82.6
11	9867	1198	992	77.3	72.9
12	9811	1224	854	58.7	69.3

Table 3
Results from key relationship extraction.

Document No.	No. of sentences	No. of extracted relations	No. of key relations
1	409	670	425
2	463	681	309
3	186	257	102
4	174	289	107
5	92	286	137
6	98	158	100
7	20	21	7
8	34	43	25
9	301	251	115
10	217	207	150
11	508	1179	799
12	443	1106	689

phrases from Document No. 7 because the descriptions contained many broken sentences such as 'The best game ever'. Additionally, the algorithm is able to achieve relatively high precision, across most of the document types, validating the proposed algorithm for selecting key phrases. The precision and recall rates can be further improved by adjusting the hyperparameters. The success in extracting key phrases contributes to helping the user gain an initial understanding of the system.

As no other approach automatically generates SysML diagrams directly from text, the comparison with the literature can only be made by considering the extraction of key phrases for similar purposes. It was observed, that the precision and recall results are comparable with or outperform most prior works. Specifically in [7], which aimed to extract domain terms as UML classes, a precision of 40% and recall of 50% were reported. In the ontology domain, the CRCTOL [43] reports more than 90% in precision and less than 5% in recall (in different ablations). The proposed method also achieves comparable results to [13] (precision of 62% and recall of 82.2%), [29] (precision between 40% and 73%, and recall between 57% and 93%), and [70] (precision between 82.5% and 85.0%, and recall between 68.0% and 53.2%).

On the other hand, the proposed approach under-performs compared to [14] (precision and recall rates around 90%) and [36] (precision of 83.82% and 91.01%). This is because, [14] assumes a fixed format for the user stories, thus making the extraction process much simpler. Similarly, [36] assumes a structure in the input text, arguably simplifying the extraction.

6.2. Key relationship extraction

The results from key relationship extraction are detailed in Table 3. As can be seen from the table, the OpenIE tool extracts multiple semantic relations from the same sentence even after applying a relation likelihood threshold of 0.5 on the OpenIE tool. This is likely attributed to the combination of different methods

Table 4
Results from key relationship classification.

Document No.	Generalisation	Composite association	Reference association	Classification accuracy %
1	26	168	226	75.8
2	11	92	204	70.9
3	3	31	68	64.7
4	6	45	55	70.1
5	4	39	93	84.7
6	1	26	72	71.0
7	0	2	5	57.1
8	0	11	13	68.0
9	7	34	73	62.6
10	1	33	115	73.3
11	12	209	577	73.0
12	5	187	497	73.6

used in OpenIE [51–54]. This is helpful in capturing the complete semantic meaning of each sentence. It can also be seen that the key relation selection algorithm is effective in removing from 27% to 65% of the extracted relations. This can also be manually adjusted by tuning the parameters $sigma_{tf-idf}$, σ_p, and $\sigma_{relationship}$. The use of multiple parameters is aimed at giving the user more control over the completeness of the generated SysML diagrams.

6.3. Relationship mapping

The results from the mapping of key relationships for BDD diagram are detailed in Table 4. A majority of the extracted textual relationships are mapped to either composite associations or reference associations. This is expected because a textual relationship is only mapped to generalisation if one phrase contains another, which is assumed to be a stricter criterion. At a difference threshold of 0.2, the number of composite associations and the number of reference associations are around the same order of magnitude, with the number of composite associations generally larger. At a difference threshold of 0.5, the number of composite and reference associations are around the same order of magnitude, with the number of reference associations generally larger.

Having a sufficient number of composite relationships and generalisation relationships is useful because they also serve as taxonomic relationships which provide structures to the SysML diagrams. This helps the algorithm to identify the hierarchy between blocks and facilitates the graphic layout algorithm. Having such a hierarchical structure is useful in helping the user understand the overall structure of the system. Additionally, the algorithm can also be seen to have achieved high accuracy in mapping the correct relationship types. The outlier in the app description dataset is potentially due to the small number of key relations for the document. This contributes to creating a more objective starting point in SysML diagrams for users.

Due to the differences in the proposed approach, comparisons are made with methodologies that aim to leverage NLP techniques for extracting/classifying requirements and relationships. For extracting relationships, the results are comparable to those reported in [43] (F-score of around 69%) and [29] (F-score from 48.0% to 81.1%). Similar results were also obtained in [41], reporting results in taxonomy construction with F-scores of between 66.7% and 69.2%. The proposed approach under-performs compared to [14], which reports a precision between 67.2% and 83.7% in extracting relationships (F-score∼80%). This is because in [14] is assumed a fixed input format, thus making the extraction process simpler. Similarly, [33] reports results with F-score of 80.3%, by making assumptions on the structure of the input documents. Additionally, when considering requirement classification results, the results obtained are comparable to [19] that uses 5%–15% of the dataset for training.

6.4. Augmentation

The results from augmentation for BDD diagrams are detailed in Table 5. As can be seen, the abstraction step adds a high number of augmented relationships and phrases. It helps provide a higher-level overview of the entire system and more structure to the SysML diagrams through taxonomic generalisation relationships. This also enables the user to quickly identify higher-level phrases within the document to plot other diagrams, for example package diagrams. Additionally, abstraction is based on the extracted phrases, ensuring that the abstracted phrases and relationships are relevant to the original corpus. This provides users with a more complete starting point to understand and design the complex system.

It can be seen that only a small number of relationships are identified between the top-level unigram phrases after the abstraction step, potentially due to the criteria used as only WordNet hypernyms/hyponyms and meronyms/holonyms relationship are identified [58,59]. It can also be seen that a large number of common hypernyms and hypernym relationships are augmented. The common hypernyms are able to provide another level of abstraction to the selected phrases based on semantics that are different from the abstraction based on lexical terms, helping the system engineer to gain a more complete high-level picture of the system and its different abstract categories in addition to contributing more structure to the SysML diagrams.

6.5. Requirement identification

The results from requirement identification are detailed in Table 6. The number of requirements identified is the same as the number of key relationships, which is expected because the requirements in this study are derived from relationships. This contributes to a more complete set of candidate requirements for the user to consider. Additionally, it can also be seen that a large number of relationships between requirements are identified and augmented through the algorithm. This leads to most of the requirements being connected using the augmented relationships, which is useful because even though relationships between phrases may be found through using sentences explicitly stated in the text, relationships between requirements, which are derived from sentences, can be difficult to identify. The proposed algorithm is able to suggest candidate relationships between requirements, allowing the user to build on the knowledge to classify relationships between requirements further. Additionally, the relationships also provide structure to the requirement diagrams by connecting the requirements and bringing related requirements to the same cluster.

6.6. BDD diagram generation

A few examples of BDD diagrams generated in the case studies are illustrated in Fig. 3 below. It can be seen that the augmentation steps were useful in building the structure of the BDD diagrams. For example, many blocks have corresponding phrases that share the same abstracted phrase, which defines the parent block that clusters these blocks together. This can also be tuned by the choice of specificity L_{phrase}. This is useful because these phrases are not explicitly related in the original document, yet they are connected via the parent blocks in the generated diagrams. For example, in Fig. 3(b), the block defined by the abstracted phrase 'display' clusters the sub-blocks such as 'display option' and 'taskbar display' into one branch. In Fig. 3(a), the block defined by the abstracted phrase 'prediction' clusters multiple sub-blocks into one branch. Additionally, blocks corresponding to top-level unigram phrases can also be connected to the same

Table 5
Results from phrase and relationship augmentation.

Document no.	No. of abstraction relations	No. of abstracted phrases	No. of augmented top-level relationships	No. of lowest common hypernym relationships	No. of lowest common hypernyms
1	246	125	1	276	96
2	224	145	3	569	168
3	98	61	1	159	58
4	66	45	0	112	53
5	104	62	1	235	85
6	107	69	0	249	80
7	10	10	0	14	9
8	19	16	0	43	23
9	83	51	2	121	51
10	182	117	0	316	104
11	736	462	26	1678	403
12	595	376	37	1672	399

Table 6
Results from requirement identification and augmentation.

Document no.	No. of requirements	No. of relationships between requirements
1	425	636
2	309	450
3	102	84
4	107	52
5	137	85
6	100	56
7	7	1
8	25	2
9	115	149
10	150	161
11	799	469
12	689	575

block defined by their common hypernyms such as 'entity' in Fig. 3(c), where 'entity' is a high-level common hypernym for many synsets in the WordNet [58,59].

These high-level phrases are augmented based on semantics, which ensures that they are relevant to the document, in addition to providing a hierarchical structure that aids user understanding. Without the augmented phrases, these extracted key phrases may become open-ended, which makes it harder for the user to understand the structure of the system from the SysML diagrams. Additionally, the two levels of abstraction capabilities, both lexical and semantic, are useful to help the systems engineer in gaining an overview of the higher-level abstract concepts in the system that are not explicitly stated in the text, and offer suggestions for candidate high-level phrases and their corresponding blocks to the systems engineers.

It can also be seen that there is a mix of extracted relationships and augmented relationships in the diagram, which implies that both steps are useful for the construction of the SysML diagrams. By combining the taxonomy derived from both generalisation and composite association relationships and both extracted and augmented relationships, the algorithm ensures a strong structure in the generated diagrams.

6.7. IBD diagram generation

The IBD diagram specifies the connection between different sub-blocks of a given parent block. An example of the IBD diagram is illustrated in Fig. 4. It is assumed that two blocks are connected with ports if an action phrase exists in the relationship whose subject and object phrases define these two blocks. The presence of action phrases and corresponding ports is useful to show the interaction between different blocks, in addition to their hierarchical relationship described by the BDD diagram. Additionally, it can be seen that there are more blocks in the IBD than in the BDD diagram, because the IBD diagram uses a different algorithm to identify sub-blocks. This acts as a redundancy measure for the identification of sub-blocks that are not captured by the BDD algorithm, to help generate a more complete overview of the system. Additionally, reference relationships with blocks that are outside the user-specified block are also included in the IBD to highlight potential interactions and communications with external blocks.

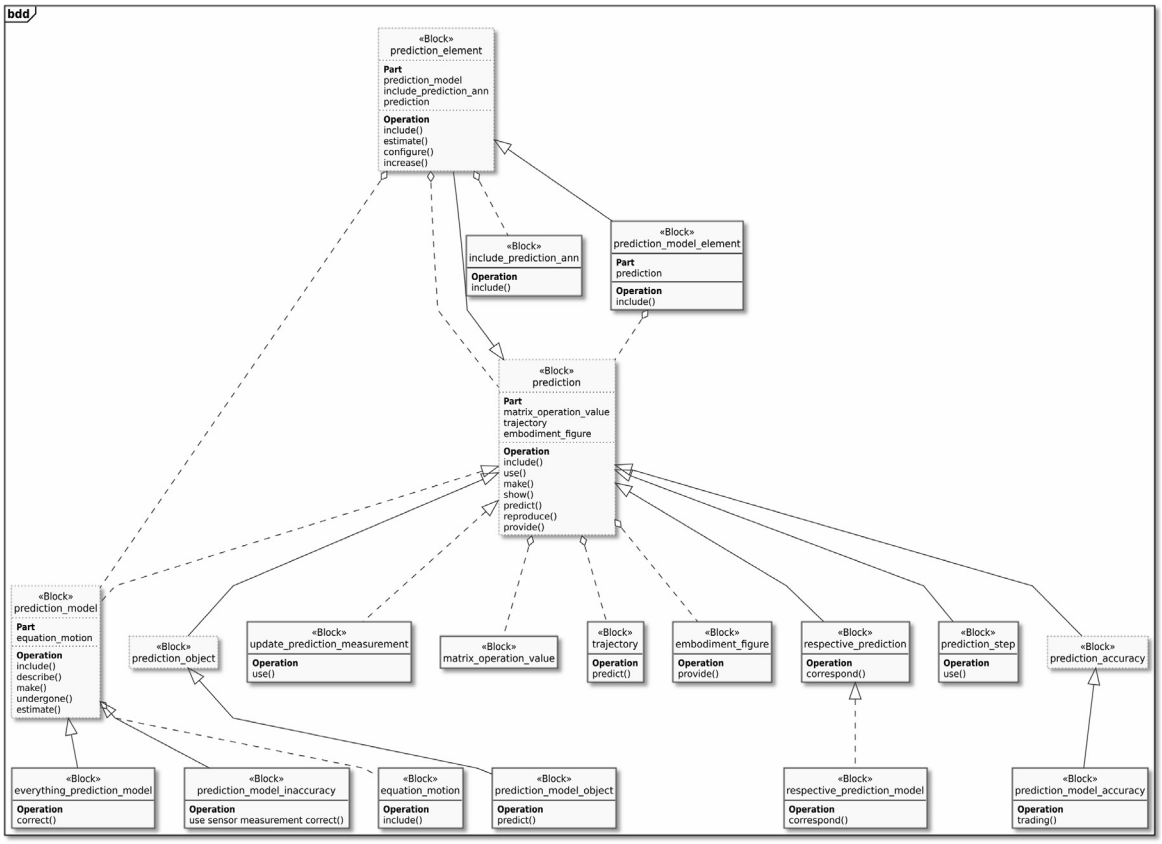
6.8. REQ diagram generation

The REQ diagram specifies the requirements satisfied by the blocks and the relationships between different requirements. An example of the REQ diagram is illustrated in Fig. 5. It can be seen that some of the requirements share the same name and. are satisfied by the same block, implying that the algorithm has successfully clustered them together. Additionally, most of the requirements are connected to each other via augmented trace relationships. These can serve as candidates for more specific types of relationships for the user. The relationships also help create a hierarchical structure in the requirement diagram, where the most connected requirement is assumed to be placed higher in the hierarchy. Moreover, because the requirements are derived from relationships extracted from individual sentences, they are mostly functional requirements. The extraction of non-functional requirements is out of the scope of this study. By presenting the extracted relationships as requirements to the user, the algorithm is also able to provide more sentence context to help the users understand the phrases and relationships presented in BDD and IBD diagrams.

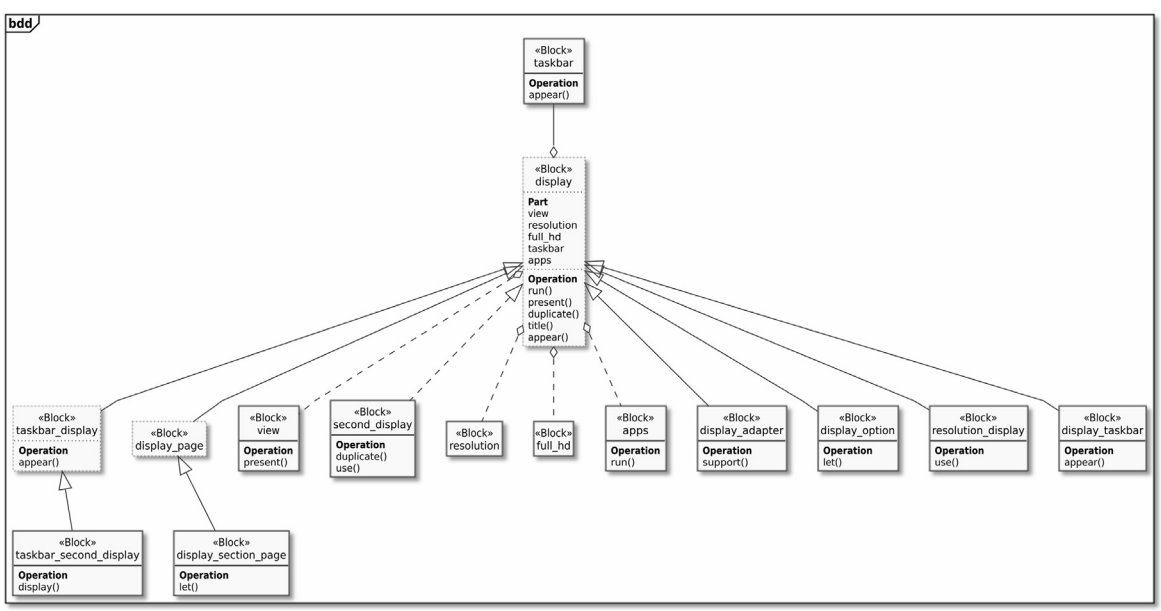
From the generated figures, it can be seen that the algorithm is able to generate structured SysML diagrams that can potentially aid engineers in designing and architecting complex systems, alongside any existing diagrams and documentations they already possess. By defining the steps and procedures in extracting key phrases and classifying relationships, the approach provides a standardised, and arguably objective starting point for the user to understand and design different systems. Additionally, the approach also provides a degree of versatility in allowing the user to freely choose and mix the textual materials to upload, and to adjust the multiple parameters to achieve the desired specificity.

6.9. Time and space complexity of the SysML diagrams generation

The time taken for the automatic generation of SysML diagrams depends on the length of the document and on the computational resources available. For example, in the patent documents with around 9000 words and using a laptop machine



(a) EPO patent dataset, selected phrase = 'prediction'

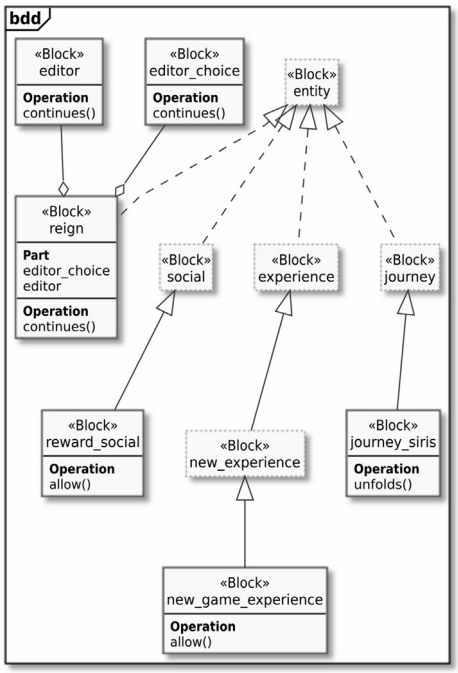


(b) Windows manual dataset, selected phrase = 'display'

Fig. 3. Automatically generated BDD diagrams.

with 8 GB RAM, the generation of SysML diagrams took around 5 min. However, the majority of the time was spent on extracting relationships using the OpenIE tool, which only needs to be done once for each document, and which does not require human involvement. In deployment, once the relationships are extracted (and saved to separate files), the time required to generate SysML diagrams is less than 1 min, enabling the user to adjust the hyperparameters and visualise updated SysML outputs quickly.

Additionally, in deployment, the operational space required for generating the SysML diagrams is also minimal, as the intermediate files are all text documents, and the resolution of the generated SysML images can be adjusted according to the user's need. The space required is less than 9 GB in total. However, most of the space needed is from the OpenIE toolbox and language models (8.76 GB), which can live on a server and not necessarily on the user's local machine.



(c) App description dataset, selected phrase = 'entity'

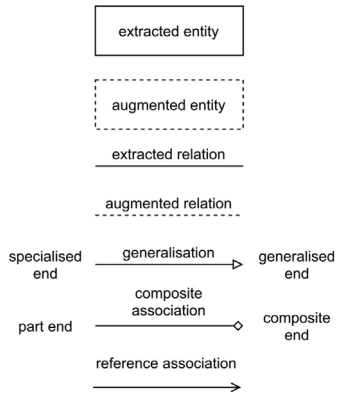


Fig. 3. (*continued*).

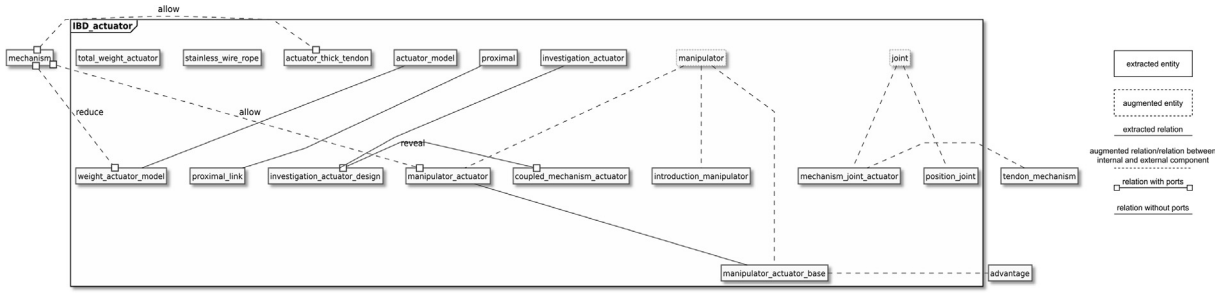


Fig. 4. IBD diagram for research paper dataset, selected phrase = 'actuator'.

7. Discussion

The work presented addresses the three research questions set in Section 2.4 by yielding an approach that is

- Flexible and open-domain. By using techniques from open information extraction, it does not require the input document to follow specific syntactic rules or to belong to a specific domain. Instead, it is able to extract the required information for plotting from free, unstructured natural language text.
- Automated: Takes natural language text as input and directly generates different types of SysML diagrams with no human intervention or intermediate modelling required.
- Parameter-based: The proposed approach has minimal reliance on heuristic rules and predefined patterns to identify hierarchies. Instead, the approach parameterises the extraction of key entities and relationships and allows the

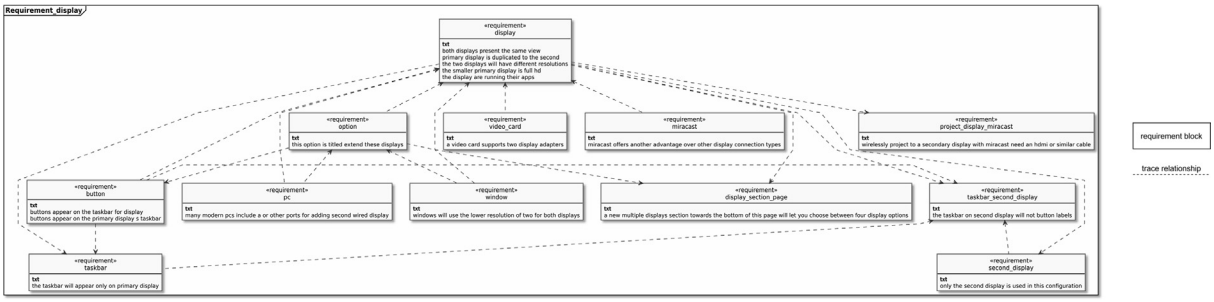


Fig. 5. REQ diagram for Windows manual dataset, selected phrase = 'display'.

user to adjust the degree of specificity using a set of parameters.

From the results presented in Section 6 it is possible to state that according to both quantitative and qualitative criteria the proposed approach is capable of outputting reasonable SysML diagrams, similar enough to those that could emerge from an engineer-made first draft. It is difficult to directly compare the results to similar work since, to the best of our knowledge, no attempts have been made, so far, to automatically generate SysML diagrams from natural text documents. Nevertheless, we regard this work as an encouraging first step towards enabling further and better automation of other systems engineering processes.

Some extensions of the current work could include:

- Development of an interactive tool from which engineers can correct and provide learning feedback to the algorithm as to what constitutes a good result or not. Such a tool could also allow for more explicit calibration and experimenting by the user of the various parameters that need tuning in order for the results to achieve sufficient quality;
- Quantification of the notion of *completeness and consistency* in some meaningful way to allow comparison against human-designed diagrams;
- Adaptation of the method to other types of system modelling languages (e.g. Object Process Methodology (OPM)) [71];
- Adaptation of the method to include Named Entity Recognition functionality.

One other aspect that is worth exploring (and that would perhaps call for an experimental setup including actual engineers working on a given task) would be to test how much of a detrimental impact could the reliance on automation have on the engineers' capacity to acquire and, most importantly, structure technical knowledge in the minds. As often happens when a previously manual task is automated, there is a risk of progressively deteriorating the level of preparation of the involved operators.

8. Conclusions

In this study, an approach to automatically generate SysML diagrams end-to-end directly from unstructured natural language text was proposed. This approach consists of six steps, and it leverages open-access tools such as NLTK, OpenIE, and PlantUML. One of the key strengths of the proposed approach is the combined use of Natural Language Processing techniques and the WordNet (available in NLTK) with the OpenIE toolbox. This enabled the automatic extraction of key textual relationships and phrases, and for the first time, the mapping and augmentation of these phrases and relationships to SysML model elements. The results are successfully leveraged to plot different structured SysML diagrams, by using the PlantUML tool. This approach requires five hyperparameters to be specified by the user, and it is open-domain. The versatility of the approach was demonstrated through the use of six case studies from different domains and using different writing styles. The results obtained with the proposed approach were validated against manually extracted results, which achieved high recall and precision in key phrase extraction, high accuracy in relationship classification, and success in generation of high-quality SysML diagrams.

This approach would benefit systems engineers in gaining a detailed graphical overview of the system dependencies at the initial design phase, since it provides a standardised, comprehensive and automated starting point from textual resources (e.g., specifications, manuals, technical reports, maintenance reports). This will support systems engineers in understanding complex systems and in quickly assessing and designing the SysML diagrams for these systems. As far as the authors are aware, this study is the first attempt at directly generating SysML diagrams from natural language text.

CRediT authorship contribution statement

Shaohong Zhong: Methodology, Software, Data curation, Validation, Visualization, Investigation, Writing – original draft. **Andrea Scarinci:** Conceptualization, Methodology, Validation, Supervision, Writing – original draft. **Alice Cicirello:** Conceptualization, Methodology, Validation, Writing – original draft, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is made available at https://github.com/ShaohongZ/NLP-for-Systems-Engineering.

Acknowledgements

The authors would like to thank Prof. Janet Pierrehumbert (University of Oxford) and Prof. Youssef Marzouk (MIT) for the helpful comments and insights at the early stages of this project. Part of this work was supported by the Department of Engineering Science at the University of Oxford, United Kingdom through the Engineering Undergraduate Research Opportunities (EUROP) in 2019 (recipient SZ, supervisors AC and AS), and by Balliol College, University of Oxford, United Kingdom through the Career Development Fellowship in Engineering Science of AC and a research bursary to SZ.

Appendix. Table of symbols

See Table A.7.

Table A.7
The definition of symbols used in the paper.

Symbol	Meaning
w	Tf-idf weighting
t	Word
d	Document
c	Corpus of documents
$w_{t,d}$	Tf-idf weighting of a word in a document
$\mathrm{tf}_{t,d}$	Term frequency of a word in a document
$\mathrm{idf}_{t,c}$	Inverse document frequency of a word in a corpus of documents
N_c	Total number of documents in a corpus
$\mathrm{df}_{t,c}$	Document frequency, the number of documents in the corpus that contain a particular word
σ_{tf-idf}	Manually specified tf–idf weighting threshold for selection of key nouns
$\sigma_{relationship}$	Manually specified confidence threshold to preliminarily filter extracted relationships
r	Textual relationships
p	Phrases
$p_{1,2,3,\ldots}$	Ordered phrases in a textual relationship
N_r	Number of phrases in an extracted relationship
$t_{1,2,3,\ldots}$	Ordered words in a phrase
N_p	Number of words in a phrase
L_{phrase}	Manually specified maximum number of nouns in a key phrase
$count_{p,d}$	Frequency of a phrase, referring to the total count of a phrase in a document
h'	WordNet depth, the semantic depth assigned by WordNet to an individual word given a context
h	WordNet score, 1-complement of the normalised WordNet depth of a word
$\lambda_{p,d}$	Score of a candidate key phrase given a document
σ_p	Manually specified score threshold for selection of key phrases from candidate key phrases
B	SysML block
R	SysML relationship between blocks
B_p	SysML block defined by a phrase
$\sigma_{rel-difference}$	Manually specified score threshold for classifying between composite and reference relationships
P	Top level phrases, the set of key phrases whose corresponding blocks do not form a sub-block of another
γ	Score of a word, which is a sum of tf–idf weighting and WordNet score
$\gamma_{t,d}$	Score of a word in a document
$\Gamma_{p,d}$	List of the scores of words in a phrase of a document
$p_{abstract}$	Abstracted phrase

References

[1] A. Kossiakoff, W. Sweet, S. Seymour, S. Biemer, Systems Engineering Principles and Practice, second ed., John Wiley & Sons, Ltd, 2011, http://dx.doi.org/10.1002/9781118001028.

[2] S. Friedenthal, A. Moore, R. Steiner, A Practical Guide to SysML, Third Edition: The Systems Modeling Language, third ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014, http://dx.doi.org/10.1016/C2013-0-14457-1.

[3] L.E. Hart, Introduction to model-based system engineering (MBSE) and sysml, in: Delaware Valley INCOSE Chapter Meeting, vol. 30, Ramblewood Country Club Mount Laurel, New Jersey, 2015.

[4] L. Delligatti, SysML Distilled: A Brief Guide to the Systems Modeling Language, first ed., Addison-Wesley Professional, 2013.

[5] E. Huang, R. Ramamurthy, L.F. McGinnis, System and simulation modeling using sysml, in: 2007 Winter Simulation Conference, IEEE, 2007, pp. 796–803, http://dx.doi.org/10.1109/WSC.2007.4419675, ISBN=978-1-4244-1305-8. ISSN=1558-4305. INSPEC Accession Number= 9847802.

[6] D. Dori, N. Korda, A. Soffer, S. Cohen, SMART: System model acquisition from requirements text, in: J. Desel, B. Pernici, M. Weske (Eds.), Business Process Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 179–194, http://dx.doi.org/10.1007/978-3-540-25970-1_12, Lecture Notes in Computer Science, vol 3080.

[7] P. Sawyer, P. Rayson, K. Cosh, Shallow knowledge as an aid to deep understanding in early phase requirements engineering, IEEE Trans. Softw. Eng. 31 (11) (2005) 969–981, http://dx.doi.org/10.1109/TSE.2005.129, Electronic ISSN: 1939-3520. INSPEC Accession Number: 8727391.

[8] A. Arellano, E. Zontek-Carney, M.A. Austin, Frameworks for natural language processing of textual requirements, Int. J. Adv. Syst. Meas. 8 (2015) 230–240.

[9] Object Management Group, OMG System Modeling Language Version 1.6, Standard, Object Management Group (OMG), 2019, URL https://www.omg.org/spec/SysML/1.6.

[10] S. Friedenthal, A. Moore, R. Steiner, OMG Systems Modeling Language (OMG SysML) Tutorial, INCOSE Int. Symp. 18 (2008) 1731–1862, http://dx.doi.org/10.1002/j.2334-5837.2008.tb00914.x.

[11] M. Hause, The SysML Modelling Language, in: Fifteenth European Systems Engineering Conference, 2006.

[12] L. Zhao, W. Alhoshan, A. Ferrari, K.J. Letsholo, M.A. Ajagbe, E.-V. Chioasca, R.T. Batista-Navarro, Natural language processing for requirements engineering: A systematic mapping study, ACM Comput. Surv. 54 (3) (2021) http://dx.doi.org/10.1145/3444689.

[13] N.H. Bakar, Z.M. Kasirun, N. Salleh, H.A. Jalab, Extracting features from online software reviews to aid requirements reuse, Appl. Soft Comput. 49 (2016) 1297–1315, http://dx.doi.org/10.1016/j.asoc.2016.07.048.

[14] M. Robeer, G. Lucassen, J.M.E.M. van der Werf, F. Dalpiaz, S. Brinkkemper, Automated extraction of conceptual models from user stories via NLP, in: 2016 IEEE 24th International Requirements Engineering Conference, RE, 2016, pp. 196–205, http://dx.doi.org/10.1109/RE.2016.40, ISBN:978-1-5090-4121-3. ISSN: 2332-6441. INSPEC Accession Number: 16519370.

[15] T. Johann, C. Stanik, A.M. Alizadeh B., W. Maalej, SAFE: A simple approach for feature extraction from app descriptions and app reviews, in: 2017 IEEE 25th International Requirements Engineering Conference, RE, 2017, pp. 21–30, http://dx.doi.org/10.1109/RE.2017.71, ISBN: 978-1-5386-3191-1. ISSN: 2332-6441. INSPEC Accession Number: 17207660.

[16] Z. Shakeri Hossein Abad, V. Gervasi, D. Zowghi, B. H. Far, Supporting analysts by dynamic extraction and classification of requirements-related knowledge, in: 2019 IEEE/ACM 41st International Conference on Software Engineering, ICSE, 2019, pp. 442–453, http://dx.doi.org/10.1109/ICSE.2019.00057, ISBN:978-1-7281-0869-8. ISSN: 1558-1225. INSPEC Accession Number: 18938395.

[17] S.J. Körner, M. Landhäuß er, Semantic enriching of natural language texts with automatic thematic role annotation, in: C.J. Hopfe, Y. Rezgui, E. Métais, A. Preece, H. Li (Eds.), Natural Language Processing and Information Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 92–99, http://dx.doi.org/10.1007/978-3-642-13881-2_9, Lecture Notes in Computer Science, vol 6177..

[18] J. Hayes, A. Dekhtyar, J. Osborne, Improving requirements tracing via information retrieval, in: Proceedings. 11th IEEE International Requirements Engineering Conference, 2003., 2003, pp. 138–147, http://dx.doi.org/10.1109/ICRE.2003.1232745, ISBN:0-7695-1980-6. ISSN: 1090-705X. INSPEC Accession Number: 7913997.

[19] A. Casamayor, D. Godoy, M. Campo, Identification of non-functional requirements in textual specifications: A semi-supervised learning approach, Inf. Softw. Technol. 52 (4) (2010) 436–445, http://dx.doi.org/10.1016/j.infsof.2009.10.010.

[20] D.V. Rodriguez, D.L. Carver, A. Mahmoud, An efficient wikipedia-based approach for better understanding of natural language text related to user requirements, in: 2018 IEEE Aerospace Conference, 2018, pp. 1–16, http://dx.doi.org/10.1109/AERO.2018.8396645, ISBN:978-1-5386-2014-4.

[21] A. Ferrari, G.O. Spagnolo, F. Dell'Orletta, Mining commonalities and variabilities from natural language documents, in: Proceedings of the 17th International Software Product Line Conference, SPLC '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 116–120, http://dx.doi.org/10.1145/2491627.2491634.

[22] J.S. Thakur, A. Gupta, Identifying domain elements from textual specifications, in: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, in: ASE 2016, Association for Computing Machinery, New York, NY, USA, 2016, pp. 566–577, http://dx.doi.org/10.1145/2970276.2970323, INSPEC Accession Number: 16358075.

[23] A. Ferrari, A. Esuli, An NLP approach for cross-domain ambiguity detection in requirements engineering, Autom. Softw. Eng. 26 (2019) 559–598, http://dx.doi.org/10.1007/s10515-019-00261-7.

[24] G. Carvalho, D. Falcão, F. Barros, A. Sampaio, A. Mota, L. Motta, M. Blackburn, Test case generation from natural language requirements based on scr specifications, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 1217–1222, http://dx.doi.org/10.1145/2480362.2480591.

[25] B.C.F. Silva, G. Carvalho, A. Sampaio, Test case generation from natural language requirements using cpn simulation, in: M. Cornélio, B. Roscoe

(Eds.), Formal Methods: Foundations and Applications, Springer International Publishing, Cham, 2016, pp. 178–193, http://dx.doi.org/10.1007/978-3-319-29473-5_11.

[26] S. Tiwari, D. Ameta, A. Banerjee, An approach to identify use case scenarios from textual requirements specification, in: Proceedings of the 12th Innovations on Software Engineering Conference (Formerly Known As India Software Engineering Conference), ISEC '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1–11, http://dx.doi.org/10.1145/3299771.3299774.

[27] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, S. Gnesi, Detecting requirements defects with NLP patterns: An industrial experience in the railway domain, Empirical Softw. Eng. 23 (6) (2018) 3684–3733, http://dx.doi.org/10.1007/s10664-018-9596-7.

[28] N. Loughran, A. Sampaio, A. Rashid, From requirements documents to feature models for aspect oriented product line implementation, in: J.-M. Bruel (Ed.), Satellite Events At the MoDELS 2005 Conference, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 262–271, http://dx.doi.org/10.1007/11663430_27, Lecture Notes in Computer Science, vol 3844..

[29] A. Sree-Kumar, E. Planas, R. Clarisó, Extracting software product line feature models from natural language specifications, in: Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1, SPLC '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 43–53, http://dx.doi.org/10.1145/3233027.3233029.

[30] L.A. Al-Safadi, Natural language processing for conceptual modeling, Int. J. Digit. Content Technol. Appl. 3 (3) (2009) 47–59.

[31] E. Casagrande, S. Woldeamlak, W.L. Woon, H.H. Zeineldin, D. Svetinovic, NLP-KAOS for systems goal elicitation: Smart metering system case study, IEEE Trans. Softw. Eng. 40 (10) (2014) 941–956, http://dx.doi.org/10.1109/TSE.2014.2339811, ISSN: 1939-3520. INSPEC Accession Number: 14652000.

[32] U. Thayasivam, K. Verma, A. Kass, R. Vasquez, Automatically mapping natural language requirements to domain-specific process models, Proc. AAAI Conf. Artif. Intell. 25 (2) (2011) 1695–1700, URL https://ojs.aaai.org/index.php/AAAI/article/view/18863.

[33] T.H. Nguyen, J. Grundy, M. Almorsy, Rule-based extraction of goal-use case models from text, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, in: ESEC/FSE 2015, Association for Computing Machinery, New York, NY, USA, 2015, pp. 591–601, http://dx.doi.org/10.1145/2786805.2786876.

[34] L. Chen, Y. Zeng, Automatic generation of UML diagrams from product requirements described by natural language, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, in: 29th Computers and Information in Engineering Conference, Parts A and B, Vol. 2, 2009, pp. 779–786, http://dx.doi.org/10.1115/DETC2009-86514.

[35] Y. Zeng, Recursive object model (ROM)-modelling of linguistic information in engineering design, Comput. Ind. 59 (6) (2008) 612–625, http://dx.doi.org/10.1016/j.compind.2008.03.002.

[36] H. Afreen, I.S. Bajwa, Generating UML class models from SBVR software requirements specifications, in: 23rd Benelux Conference on Artificial Intelligence (BNAIC 2011), Citeseer, 2011, pp. 23–32.

[37] D.K. Deeptimahanti, R. Sanyal, Semi-automatic generation of UML models from natural language requirements, in: Proceedings of the 4th India Software Engineering Conference, ISEC '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 165–174, http://dx.doi.org/10.1145/1953355.1953378.

[38] T.R. Gruber, Toward principles for the design of ontologies used for knowledge sharing, Int. J. Hum.-Comput. Stud. 43 (5–6) (1995) 907–928, http://dx.doi.org/10.1006/ijhc.1995.1081.

[39] M.N. Asim, M. Wasim, M.U.G. Khan, W. Mahmood, H.M. Abbasi, A survey of ontology learning techniques and applications, Database 2018 (2018) http://dx.doi.org/10.1093/database/bay101.

[40] P. Cimiano, J. Völker, Text2Onto: A framework for ontology learning and data-driven change discovery, in: Proceedings of the 10th International Conference on Natural Language Processing and Information Systems, NLDB '05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 227–238, http://dx.doi.org/10.1007/11428817_21.

[41] E. Drymonas, K. Zervanou, E.G.M. Petrakis, Unsupervised ontology acquisition from plain texts: The OntoGain system, in: Proceedings of the Natural Language Processing and Information Systems, and 15th International Conference on Applications of Natural Language to Information Systems, NLDB '10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 277–287, http://dx.doi.org/10.1007/978-3-642-13881-2_29.

[42] P. Velardi, S. Faralli, R. Navigli, OntoLearn Reloaded: A Graph-Based Algorithm for Taxonomy Induction, Comput. Linguist. 39 (3) (2013) 665–707, http://dx.doi.org/10.1162/COLI_a_00146.

[43] X. Jiang, A.-H. Tan, CRCTOL: A Semantic-Based Domain Ontology Learning System, J. Am. Soc. Inf. Sci. Technol. 61 (1) (2010) 150–168, http://dx.doi.org/10.1002/asi.21231.

[44] M. Mejhed Mkhinini, O. Labbani-Narsis, C. Nicolle, Combining UML and ontology: An exploratory survey, Comp. Sci. Rev. 35 (2020) 100223, http://dx.doi.org/10.1016/j.cosrev.2019.100223.

[45] D. Jurafsky, J.H. Martin, Speech and Language Processing (2nd Edition), Prentice-Hall, Inc., USA, 2009, ISBN=0131873210.

[46] T. Kiss, J. Strunk, Unsupervised Multilingual Sentence Boundary Detection, Comput. Linguist. 32 (4) (2006) 485–525, http://dx.doi.org/10.1162/coli.2006.32.4.485, arXiv:https://direct.mit.edu/coli/article-pdf/32/4/485/1798345/coli.2006.32.4.485.pdf.

[47] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit, O'Reilly Media, Inc., 2009.

[48] H.P. Luhn, A statistical approach to mechanized encoding and searching of literary information, IBM J. Res. Dev. 1 (4) (1957) 309–317.

[49] K.S. Jones, A statistical interpretation of term specificity and its application in retrieval, J. Doc. (1972).

[50] M. Mausam, Open information extraction systems and downstream applications, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016, pp. 4074–4077.

[51] J. Christensen, Mausam, S. Soderland, O. Etzioni, An analysis of open information extraction based on semantic role labeling, in: Proceedings of the Sixth International Conference on Knowledge Capture, in: K-CAP '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 113–120, http://dx.doi.org/10.1145/1999676.1999697.

[52] H. Pal, Mausam, Demonyms and compound relational nouns in nominal open IE, in: Proceedings of the 5th Workshop on Automated Knowledge Base Construction, Association for Computational Linguistics, San Diego, CA, 2016, pp. 35–39, http://dx.doi.org/10.18653/v1/W16-1307.

[53] S. Saha, H. Pal, Mausam, Bootstrapping for numerical open IE, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 317–323, http://dx.doi.org/10.18653/v1/P17-2050, URL https://aclanthology.org/P17-2050.

[54] S. Saha, Mausam, Open information extraction from conjunctive sentences, in: Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 2288–2299, URL https://aclanthology.org/C18-1194.

[55] V. Punyakanok, D. Roth, W.-t. Yih, The importance of syntactic parsing and inference in semantic role labeling, Comput. Linguist. 34 (2) (2008) 257–287.

[56] R. Johansson, P. Nugues, The effect of syntactic representation on semantic role labeling, in: Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1, COLING '08, Association for Computational Linguistics, USA, 2008, pp. 393–400.

[57] A. Pauls, D. Klein, Faster and smaller N-gram language models, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT '11, Association for Computational Linguistics, USA, 2011, pp. 258–267.

[58] C. Fellbaum, WordNet: An Electronic Lexical Database, Bradford Books, 1998.

[59] Princeton University, About WordNet, Princeton University, 2010, URL https://wordnet.princeton.edu/.

[60] M. Lesk, Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone, in: Proceedings of the 5th Annual International Conference on Systems Documentation, SIGDOC '86, Association for Computing Machinery, New York, NY, USA, 1986, pp. 24–26, http://dx.doi.org/10.1145/318723.318728.

[61] Plant UML Team, PlantUML, 2021, URL http://plantuml.com/.

[62] PlantUML Team, Drawing UML with PlantUML: PlantUML Language Reference Guide, 2021, URL http://plantuml.com/guide.

[63] J. Ellson, E. Gansner, L. Koutsofios, S.C. North, G. Woodhull, Graphviz—Open source graph drawing tools, in: P. Mutzel, M. Jünger, S. Leipert (Eds.), Graph Drawing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 483–484.

[64] European Patent Office, EP full-text data for text analytics, 2021, URL https://www.epo.org/searching-for-patents/data/bulk-data-sets/text-analytics.html licensed under Creative Commons Attribution 4.0 International Public License https://creativecommons.org/licenses/by/4.0/.

[65] P. Thurrott, R. Riveram, M. McClean, Windows 10 Field Guide, Leanpub, 2021, [accessed 25-Mar-2021].

[66] Arup, S. Broadberry, T. Leunig, J.R. Bryson, J. Clark, R. Mulhall, H.-J. Chang, A. Andreoni, M.L. Kuan, S. Deakin, et al., Future of manufacturing (2013), 2021, URL https://www.gov.uk/government/collections/future-of-manufacturing#project-report (Accessed 25 March 2021).

[67] finnqiao, Apple Appstore Descriptions, 2021, URL https://github.com/finnqiao/apple_appstore (Accessed 25 March 2021).

[68] IEEE, IEEE Xplore, 2021, URL https://ieeexplore.ieee.org/ (Accessed 19 March 2021).
[69] Wikipedia contributors, Member states of the united nations — Wikipedia, the free encyclopedia, 2021, URL https://en.wikipedia.org/w/index.php?title=Member_states_of_the_United_Nations&oldid=1052617810 (Accessed 13 November 2020).
[70] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, B. Mobasher, Supporting domain analysis through mining and recommending features from online product listings, IEEE Trans. Softw. Eng. 39 (12) (2013) 1736–1752.
[71] D. Dori, Object-process methodology, in: Encyclopedia of Knowledge Management, Second Edition, IGI Global, 2011, pp. 1208–1220.