Model-based systems engineering to design collaborative robotics applications

Hernandez, Carlos; Fernandez-Sanchez, Jose Luis

DOI
10.1109/SysEng.2017.8088258

Publication date
2017

Document Version
Accepted author manuscript

Published in
Proceedings of the 2017 IEEE International Symposium on Systems Engineering (ISSE 2017)

2017 IEEE International Systems Engineering Symposium (ISSE), Vienna, 2017, pp. 1-6.
doi: 10.1109/SysEng.2017.8088258 ©2017 IEEE

Model-Based Systems Engineering to Design Collaborative Robotics Applications

Carlos Hernandez
TU Delft Robotics Institute
Delft University of Technology
Mekelweg 2, 2628CD Delft, The Netherlands
Email: c.h.corbato@tudelft.nl

Jose Luis Fernandez-Sanchez
Industrial Engineering School
Universidad Politecnica de Madrid
Email: joselfernandez@ieee.org

***Abstract*—Novel robot technologies are becoming available to automate more complex tasks, more flexibly, and collaborating with humans. Methods and tools are needed in the automation and robotics industry to develop and integrate this new breed of robotic systems. In this paper, the ISE&PPOOA methodology for Model-Based Systems Engineering is applied for the development of robotic systems. The methodology is described through its application to reengineer a state-of-the-art collaborative robot application. The challenges that robotic systems present to model-based systems engineering are discussed, together with the benefits of MBSE methodologies.**

I. Introduction

Robotic solutions in traditional automation allow to automate simple, repetitive tasks over large volumes of products, with little variations, in completely controlled environments. Typical applications include welding, painting, or handling the same parts presented in fixed positions. Traditional engineering methods were sufficient to integrate these systems, which mainly consist of robot manipulators that execute fixed motions, with very little flexibility. However, currently there is a need to automate high-mix and low volume productions. New robotic systems with perception capabilities to adapt their behaviour at runtime, and easily reconfigurable for different tasks are demanded by industry [1]. To meet these needs, novel robot technologies are becoming available to automate more complex tasks, more flexibly, and collaborating with humans [2].

Developing these new robotic applications requires the integration of diverse components to implement advanced robot capabilities. An example is the winning entry in the international Amazon Robotics Challenge 2016 [3], led by one of the authors. The system integrated 3D cameras, a industrial manipulator and a custom gripper, to be able to detect and handle a large variety of products in a semi-structured warehouse environment.

These new robotic components include novel technologies in mechanical engineering, electronics, control, and embedded software. Traditional, document-centric engineering methodologies can no longer cope with the complexity and the engineering demands in these new robotic systems.

Methods and tools are needed in the automation and robotics industry to develop and integrate this new breed of robotic systems. Model-based systems engineering (MBSE) uses models to consistently integrate the diverse information and engineering decisions through the life-cycle, including requirements, functional and physical architectures, detailed design, implementation, and validation.

This paper presents the early results of applying the Integrated Systems Engineering and Pipelines of Processes in Object-Oriented Architectures (ISE&PPOOA) [4] to the re-engineering of a collaborative robotic system for a product handling application. The motivation is to obtain a functional architecture of the system, independent of concrete technical solutions. Analysing the robot's behaviour at this abstract level allows to detect issues and identify solutions in the design that can be reused across different implementations, for example in relation to safety, and modularity in the robot's physical architecture. The application of the methodology also allows to identify the design heuristics and patterns realised in the robot behaviour to address functional and non-functional requirements, such as safety, and their interfaces. This allows to decouple the design solutions for the different robot capabilities from their specific physical implementation, promoting their reuse.

The remaining of the paper is organized as follows: Section 2 presents the ISE&PPOOA MBSE process. Section 3 presents the early modelling results of its application to a collaborative robot. Section 4 discusses the main benefits of the approach and the contributions of this work. Finally, Section 5 provides some concluding remarks.

II. The ISE&PPOOA MBSE process

The ISE&PPOOA method can be used as a re-engineering process for software intensive autonomous systems [5]. Traditional functional based systems engineering is combined with MBSE using the functional paradigm to represent the system behavior. The systems engineering part of the ISE&PPOOA process, shown in Figure 1 as an activity diagram, is described below with more detail.

The main goal of using systems engineering process in this work is the creation of the functional and physical architectures of the robotic system, identifying the subsystems and their interfaces. The robotic system may have subsystems that are software intensive and/or non software intensive where physics conservation laws of mass, energy and momentum are

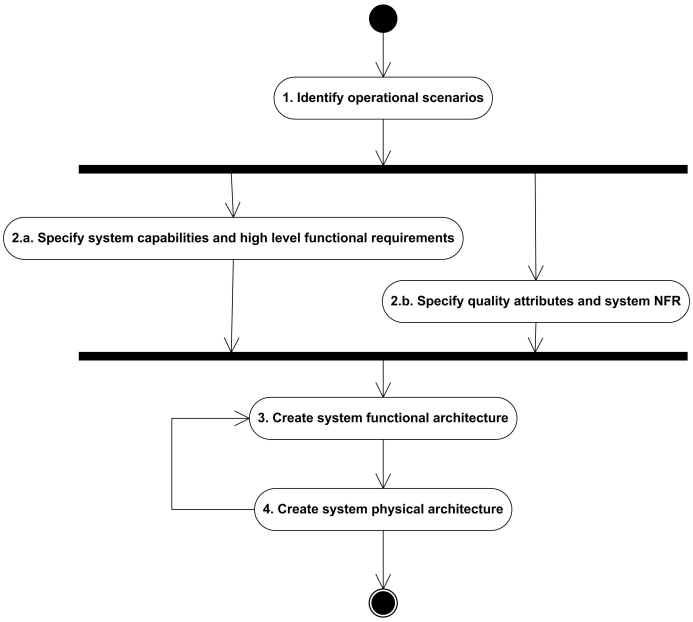

Fig. 1. ISE&PPOOA Systems Engineering Process.

an important issue that should be considered when representing the system views.

The process presented in Figure 1 has five main steps that are performed sequentially for steps 1, 2 and iteratively for steps 3 and 4. The Step 2 is split into two parallel or concurrent steps labeled 2a and 2b. For the sake of simplicity, we only represent the process steps avoiding the representation of the deliverables that are inputs and outputs of the different steps.

The goal of each step and the deliverables produced are described below.

A. Step 1. Identify the system operational scenarios

Goal: Identify the operational context of the system and describe its operational scenarios for different modes of operation.

Deliverable: The collaborative robotic system intended behaviors are described by the operational scenarios, where additionally to the preconditions, post conditions and steps of each scenario, the operational needs are identified. These operational needs are the inputs for the later identification of some system capabilities and quality attributes in the following steps of the subprocess.

B. Step 2a. Specify system capabilities and high level functional requirements

Goal: Transform the operational needs into a set of system capabilities and high level system requirements.

Deliverable: The deliverable is the representation of the robotic system capabilities with a hierarchical decomposition using the block definition diagram of SysML. System functional requirements specified in natural language but based on the hierarchical decomposition are obtained.

C. Step 2b. Specify quality attributes and system NFRs

Goal: Transform operational needs into a set of quality attributes for example efficiency, availability, safety and others including the associated non-functional requirements.

Deliverable: In decomposing a non-functional requirement, the systems engineer can chose to decompose its type (safety, reliability, other) based on a selected quality framework, or its topic considering if they apply to the whole system or one of its parts. It is possible and should be taken into account, that some non-functional requirement may be affected either positively or negatively at the same time. These interactions are very important because they have an impact on the following architecting steps of the SE process.

D. Step 3. Create system functional architecture

Goal: Transform functional requirements into a functional architecture identifying the functional hierarchy, functional behavior and functional interfaces.

Deliverable: The deliverable is the functional architecture representing the functional hierarchy using a SysML block definition diagram. This diagram is complemented with activity diagrams for the main system functional behavior flows. The N^2 diagram is used as an interface diagram where the main functional interfaces are identified. A textual description of the system functions is provided as well.

E. Step 4. Create system physical architecture

Goal: Transform the functional architecture into the architecture of the solution or physical architecture. In the ISE&PPOOA process, the selection of the solution is based on functions clustering and design heuristics. An heuristic here is a means of satisfying a quality attribute response measure by manipulating some aspect of a quality attribute model through design decisions.

Deliverable: The deliverable is the physical architecture representing the system decomposition into subsystems and parts using a SysML block definition diagram. This diagram is complemented with SysML internal block diagrams for each subsystem and activity and state diagrams as needed. A textual description of the system blocks is also provided as well. The heuristics used for the particular architecture solution are identified and documented. Optionally and if it is needed, SysML parametric diagrams are used to describe constraints in system properties to support engineering analysis.

III. Exemplary Case: Collaborative Robot Application

The ISE&PPOOA methodology has been applied to re-engineer a collaborative product handling robot application. The robot picks products from a packaged stack, and delivers them to a human operator by dropping them at a specified location (see Fig. 2). The robot prototype has been developed as part of the Factory-in-a-day European project [2], and has been demonstrated already in industrial and research events[1])

[1]The robot was showcased during RoboBusiness Europe 2017 delivering Lego sets to visitors (see http://www.factory-in-a-day.eu/successfull-trade-fair-for-factory-in-a-day

Fig. 2. Collaborative product handling robot application developed in Factory-in-a-day.

The following operation scenarios have been identified in step 1 of ISE&PPOOA methodology:
S0. System Configuration and Calibration: the robot is deployed in its operation environment, calibrate the reference locations to pick and deliver the products, and configure the stacking pattern in the product feeding container.
S1. System Start: all components in the system are booted up making them ready for standard operation.
S2. Pick product: the robot manipulator grasps and retrieves a product from the container, holding it.
S3. Deliver product: the robot moves to the deliver location and drops the item if the bag held by the operator is detected.
S4. Shutdown: all system activity is stopped and its components powered off.

As an example, the description of the *'Pick product'* scenario in ISE&PPOOA is presented in Table I.

From the operational scenarios the set of capabilities shown in Fig. 3 were identified in ISE&PPOOA step 2.a, which are described in the domain of the system's mission, totally independent of the concrete technical solutions implemented. Including general categories for capabilities helps analyse new applications without leaving out specific capabilities that could otherwise be forgotten. This is the case of *Quick deployment times* in this case, in relation to *increased availability*, a general capability for production systems. It also allows identifying different concerns in the desired behaviour. For example, avoiding obstacles is a behaviour intended for safety (*Harmless* capability), but it also relates to *contingency management*, and important issue for increased availability.

A key safety consideration is that robot and the operator share part of the workspace: it is a collaborative application. For this reason, the application design uses a UR5 Universal Robot collaborative manipulator and a light 3D printed suction-based gripper without sharp edges. A standard safety design provided by many collaborative manipulators, such as the UR5, is to trigger a robot safety stop when a collision is detected. However, this decreases productivity. Avoiding collisions is a more desirable. This is the approach taken in the Factory-in-a-day project, for which different technologies have been developed to augment collaborative robot manipulators with dynamic obstacle avoidance [6]. Two of these technologies have been integrated in the system presented here: a robot skin and a motion planning framework. The engineering effort discussed here includes the development of the application control system that integrates the different elements presented in Fig. 4 as external entities or actors.

The motion planning framework is a key part of the Robot Control. It generates collision-free trajectories for the task, using planning algorithms and a 3D model of the environment. The proximity-sensing skin provides dynamic obstacle detection information. A motion planning and execution module executes the task trajectories. When a trajectory crosses the shared space in the environment, the module also monitors the obstacle detection input to pause the execution of the current trajectory and if an obstacle is detected, to resume the robot motion once obstacles are cleared (e.g. see how interruptible regions are used in the application's activity diagram in Fig. 5).

The innovative proximity-sensing Artificial Robot Skin (ARS) developed by the Institute of Cognitive Systems Systems in the Technical University of Munich [7], [8]. This modular skin consist of identical 'cells' physically connected forming skin patches. These patches can be applied to cover the robots links and joints, while being electronically connected to work as a single, modular robot skin. Each cell in the skin produces 4 modalities of perception: 3D acceleration,

TABLE I
ISE&PPOOA DESCRIPTION OF THE *Pick product* SCENARIO.

Pick product	
Preconditions	System initialized. Non-empty feeder container.
Triggering event	New bag detected at the delivery location.
Description	The robot arm performs the actions required topick a product from the container. The system uses the skin sensing information to avoid any collisionswhile moving.
Postconditions	Robot gripper is holding a product. The stack has decreased by one unit.
Operational needs	*ON S2_1* The system shall pick cardboard boxes of dimensions,263 x 162 x 65 mm and lightweight. *ON S2_2* The system shall pick parts from a 500x330x250 mm,box, stacked in two rows and standing in vertical orientation. *ON S2_3* The system shall complete the task in less than 4,secs if there are no potential collisions while performing. *ON S2_4* The system should shall pick the item safely,,guaranteeing the safety of the operator sharing its workspace.

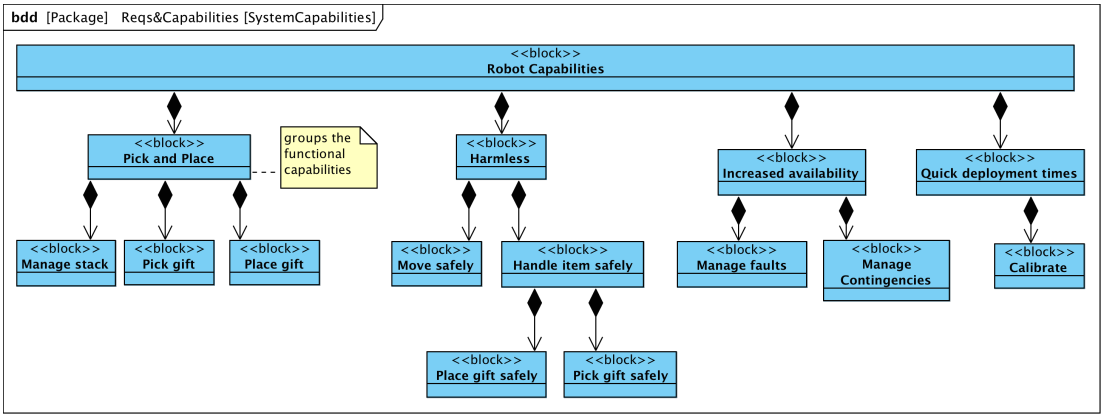

Fig. 3. Capabilities identified for the collaborative robot application using the ISEE&PPOOA method.

force, temperature, and distance. In the current application the distance readings are used to detect obstacles.

The motion planning and execution module is implemented using the MoveIt! framework [9]. The complete application control has been integrated using the Robot Operating System (ROS) framework [10] and using available open-source ROS packages, e.g. to interface the different hardware devices. The application logic is implemented as a state machine to control the execution of the different robot actions.

Contrary to a new development, reverse engineering of the robotic system has been applied to obtain the functional architecture. The initial functional behavior of the robotic system can be represented by the activity diagram in Fig. 5, which shows the basic actions in the system. The identification of the interruptible regions in the different activity flows are the basis for the safety analysis at the functional level, and it is additional value of ISE&PPOOA. As shown in Fig. 5, only the execution of two specific motion segments can be interrupted by the detection of obstacles. These are the two trajectories in which the robot moves inside the workspace shared with the operator. The behaviour triggered by the interruption is a discrete sequence of actions. To guarantee the verifiability of the behaviour, the initial design of the functions to generate collision-free trajectories, initially based on stochastic online planners, will be modified to use a database of trajectories computed and verified offline.

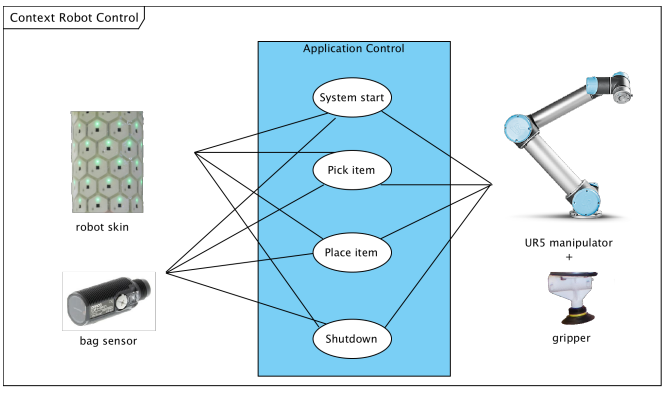

Fig. 4. Context diagram for the robot control system developed.

A preliminary functional decomposition of the system has been obtained by identifying functional interfaces and grouping the low level actions in the activity diagram in Fig. 5. The functional interfaces of the high level functions of the robot control system are represented with a N2 chart (see Table II). The upper row represents the inputs from other subsystems, while the last column represent its outputs. This tabular representation facilitates the modular design of the physical architecture by clustering techniques to be applied at the appropriate level of the functional architecture. For example, table III represents the functional interfaces of F3 subfunctions.

IV. Main Contributions and Benefits

The main benefits of the ISE&PPOOA methodology applied to the control of the robotic system are related to the modelling of the robot capabilities and functional interfaces. This facilitates reuse, maintenance and scalability.

Firstly, the modelling of the functional interfaces allows to identify the functional dependencies amongst the robot capabilities, modularizing the design solutions and thus promoting their reuse for other applications, or modifying their implementation to address new operational needs. For example, if in a new application the robot has to pick the product from a human who hands it over,the subfunctions of *F3 Move Robot Safely* can be reused in a different design of the pick behaviour, in which interruptible trajectories and obstacle information are used to implement the motions for picking the products. Another example could be a new requirement for the robot to perform faster. In this case the implementation of the functions to plan the motions and execute the trajectories could be modified with faster planning algorithms or better performing robot control parameters.

Secondly, design heuristics application is one of the main issues to develop the system's physical architecture. Heuristics

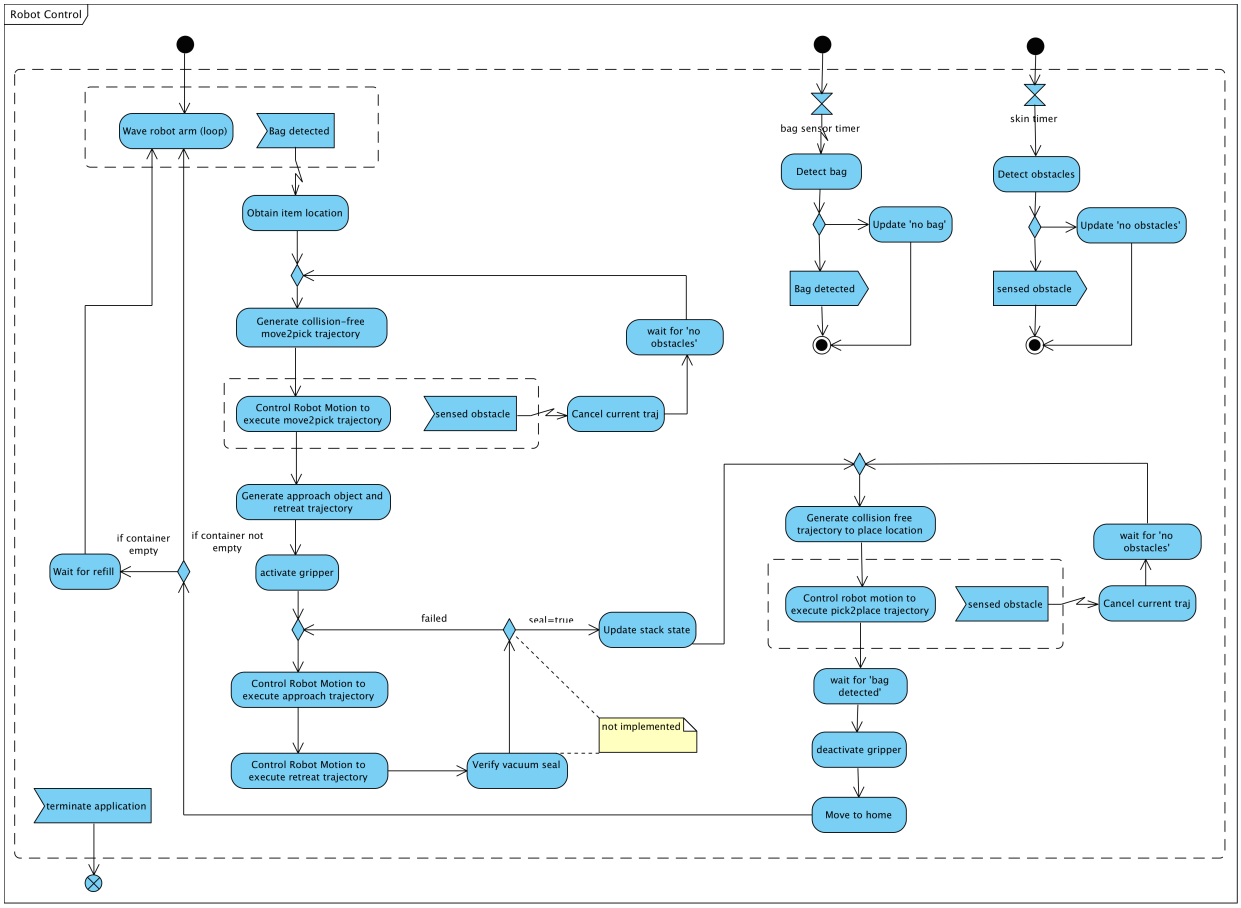

Fig. 5. Activity diagram of the robot control.

N2 Chart of the Collaborative Robot Application Functional Interfaces						
	contents of the container	- environment 3D model - drop pose - start application - skin distance signal - current robot status	- start application event - skin pressure signal	- positions of items in the container - current robot joint configuration	bag sensor signal	
F1 Coordinate application	- stack contents request - update stack	- request plan - request motion execution		- request plan - request motion execution - gripper OFF - gripper ON		
container empty	**F2 Manage stack of products**	index of current item		index of current item		updated contents of the container
- joint trajectory to input pose - robot at desired location		**F3 Move Robot Safely**				- robot joint path - stop robot & cancel current joint path
- start request - refilled update	refilled update		**F4 Operator interface**			
- joint trajectory to input pose - robot at desired		current robot joint configuration		**F5 Handle product**		- vacuum in the gripper - robot joint path
bag detected					**F.6 Detect bag**	

TABLE II

from several sources and based on experience have been col- lected by the authors. For the robot application, the heuristics

applied are related to safety, efficiency and general systems engineering concerns such as functional clustering. Safety is a main concern in a collaborative robotic system. Safety concerns are the identification and management of hazards and hazards are caused by failures. Physical devices can produce random failures but software does not fail randomly; software failures are due to design faults. Safety is achieved by avoiding or protecting against system failures. One heuristic we applied is the avoidance of non-deterministic behavior. This heuristic is aimed to enforce a specific sequence of actions, by handling random events (for example dynamic obstacle detection) and controlling all access to shared resources. This heuristic is related to the efficiency concerns as well. This heuristic can be used when correct sequencing of actions must be ensured, especially when commission failures are safety concerns. It can be implemented by either hardware or software. The robot behavior models and the identification of interruptible regions in the activity diagrams of the robotic system in Fig. 5 contribute to implement this heuristic.

Finally, consistent modeling of the requirements and robot system capabilities has allowed to detect elements missing in the initial design, for example implementation of more automated and simple routines for calibration and configuration, system start and shutdown.

V. Conclusions

This paper presents the application of model-based systems engineering and architecture centric design for advanced industrial robot applications. The ISE&PPOOA methodology is applied to obtain the functional architecture of a collaborative product handling robot application. The benefits identified include:

- Modelling the interfaces of the robot functions allows to modularize the system, reducing the time to implement changes, for example replacing certain components, to address new requirements.
- Identification of design heuristics and patterns allows to capture design solutions for robot capabilities independently or their physical implementation, promoting their reuse.

In addition, challenges for the application of MBSE to robotic systems have been encountered, such as: modelling robot behaviours that implement multiple functions, and reuse of the modelling artifacts. The research presented is work in progress. Next steps include the integrated modelling of the physical architecture and the functional architecture in SysML, the detailed specification and traceability to the application's quality attributes and associated performance and other non-functional requirements, with a special focus on safety.

N2 Chart of 'F3: Move Robot Arm Safely' Functional Interfaces			
- environment 3D model (URDF) - request plan - drop pose - index of current item	- current robot status - request motion execution	- start application - skin distance signal	
F3.1 Plan collision-free trajectory	- joint trajectory to input pose		- joint trajectory to input pose
- current robot joint configuration	**F3.2 Execute and Monitor interruptible trajectory**		- robot joint path - robot at desired location - stop robot & cancel current joint path
	- obstacle detected - no obstacle	**F3.3 Sense obstacle**	

TABLE III

VI. Acknowledgements

The work leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 609206, and Horizon 2020 research and innovation programme under grant agreement no. 732287. The authors thank their colleagues at the Institute of Cognitive Systems in TU Munich, Materialise, LAAS-CNRS, Siemens and the TU Delft Robotics Institute for the development the robotic system.

References

[1] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, vol. PP, no. 99, pp. 1–17, 2017.

[2] M. Wisse, "Factory in a Day Project (FiaD)," http://www.factory-in-a-day.eu/, Oct. 2013.

[3] C. Hernandez, M. Bharatheesha, W. Ko, H. Gaiser, J. Tan, K. van Deurzen, M. de Vries, B. V. Mil, J. van Egmond, R. Burger, M. Morariu, J. Ju, X. Gerrmann, R. Ensing, J. van Frankenhuyzen, and M. Wisse, "Team delft's robot winner of the amazon picking challenge 2016," *CoRR*, vol. abs/1610.05514, 2016. [Online]. Available: http://arxiv.org/abs/1610.05514

[4] J. L. Fernández-Sánchez, "ISE & Process Pipelines in OO Architectures (ISE&PPOOA)." [Online]. Available: http://www.omgwiki.org/MBSE/doku.php?id=mbse:ppooa

[5] J. L. Fernandez, J. Lopez, and J. P. Gomez, "Feature article: Reengineering the avionics of an unmanned aerial vehicle," *IEEE Aerospace and Electronic Systems Magazine*, vol. 31, no. 4, pp. 6–13, April 2016.

[6] M. Bharatheesha, C. Hernandez, M. Wisse, N. Giftsun, and G. Dumonteil, "Dynamic Obstacle Avoidance for Collaborative Robot Applications," in *Workshop on IC3 - Industry of the future: Collaborative, Connected, Cognitive. Novel approaches stemming from Factory of the Future and Industry 4.0 initiatives, IEEE International Conference on Robotics and Automation (ICRA), Singapore, accepted May 2017.*, 2017.

[7] P. Mittendorfer, E. Yoshida, and G. Cheng, "Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot." *Advanced Robotics*, vol. 29, no. 1, pp. 51–67, 2015.

[8] E. Dean-Leon, B. Pierce, P. Mittendorfer, F. Bergner, K. Ramirez-Amaro, W. Burger, and G. Cheng, "TOMM: Tactile Omnidirectional Mobile Manipulator," in *2017 IEEE International Conference on Robotics and Automation (ICRA), Accepted*, May 2017.

[9] I. A. Sucan and S. Chitta. "MoveIt!", Online available: http://moveit.ros.org.

[10] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.