Disciplined use of BPMN for mission modeling of Multi-Robot Systems

Khalid Bourr[1], Flavio Corradini[1], Sara Pettinari[1], Barbara Re[1], Lorenzo Rossi[1] and Francesco Tiezzi[2]

[1]School of Science and Technology, University of Camerino, Italy
[2]Dipartimento di Statistica, Informatica, Applicazioni "Giuseppe Parenti" (DISIA), University of Florence, Italy

Abstract

Nowadays, Multi-Robot Systems are an emerging research field under the umbrella of Cyber-Physical Systems. They consist of a group of robots that cooperate to accomplish a common mission. Examples of these systems are present in many application fields, e.g., agriculture, manufacture, industry, military, and health. As a consequence, there exist many frameworks facilitating the development of robotics systems. However, these tools require high skills for programming each robot's behavior and coordinating the interactions among them, which overall should produce the cooperative behavior of the Multi-Robot System needed to carry out its mission successfully. To address this problem, we propose an approach for high-level modeling the cooperative behavior of Multi-Robot Systems through disciplined use of collaboration diagrams as they are provided by the BPMN 2.0 standard. The definition of our modeling proposal has been driven by ROS2, taken as the reference framework for programming robotics systems, and its DDS implementation for intra- and inter-robot communication. We illustrate the proposed approach through a Multi-Robot System in a smart agriculture scenario.

Keywords

Multi-Robot Systems, BPMN modeling, ROS2, DDS

1. Introduction

In the Cyber-Physical Systems (CPS) domain, collaborative scenarios involving multiple robots are gaining more and more interest. The term Multi-Robot System (MRS) refers to a group of robots that work together to accomplish a common mission faster and cheaper than a single-robot system while providing scalability, reliability, flexibility and versatility [1]. In these kinds of systems, a single entity has few functionalities since the real power lies in the cooperation and the coordination of multiple robots. MRSs bring advantages in different application domains, e.g., agriculture, manufacture, industry, and health. This because they reduce the need of (possibly dangerous) human operations, e.g., the cleanup of toxic waste and rescue missions in disasters. Moreover, MRSs reduce human errors in repetitive tasks, and improves the productivity and the quality of services in human-centered applications [2].

To foster the development of MRSs, many robotic-centered frameworks provide tools by which developers can build robotics software without worrying about the underlying hardware. One of the most prominent development frameworks is Robot Operating System (ROS), which comes with ready-to-use libraries handling complex tasks. Despite the facilities provided

PoEM'21 Forum: 14th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling, November 24-26, 2021

by ROS, developing reliable MRSs is still a challenging task [3], which requires advanced programming skills to consider all at once the distributed computations made by each robot and the message exchanges needed to coordinate them. Overall, the behavior of single robots and their interactions should produce the cooperative behavior of the system that should lead to the successful accomplishment of the mission. To cope with this challenge, the literature presents several approaches belonging to Model-Driven Engineering (MDE) and Domain Specific Languages (DSL), that promise efficient and flexible methodologies for modeling and developing robotic systems [4, 5]. Nevertheless, these approaches either lack in providing a high-level abstraction of the the whole system behavior, or even discard at all scenarios involving multiple robots. To fill this gap, we propose a disciplined use of BPMN [6] for enabling the high-level modeling, and hence facilitating the development of the cooperative behavior of MRSs.

The motivation behind using BPMN, and in particular the collaboration diagrams, comes from its wide usage in MDE approaches for CPS [7]. BPMN collaborations are highly expressive and encapsulate in a unique diagram the interplay among control-flow, data-flow and communication in a flexible and understandable way, making the modeling activity easier. The resulting models are easy to read by humans and, at the same time, suitable for model-to-code solutions. Furthermore, BPMN relies on several techniques from the BPM research community supporting the development of high-quality systems, such as formalization [8], execution and animation [9], and verification [10].

It is worth noticing that many of the works that foster the use of BPMN for modeling IoT and CPS (see [7] for a survey on this topic) propose extensions of the notation to capture domain specific features. Differently from these works, we rely only on the elements of collaboration diagrams defined in the specification document of the BPMN 2.0 standard. In fact, we show that it is possible to successfully model the behavior of a cooperative MRS, without extending the standard. This brings two main advantages: (i) model designers are not required to learn new concepts, and (ii) existing BPMN tools can be used without the need of any customization. The definition of our modeling proposal results from a balance of opposing forces. On the one hand, it is driven by the need of providing a notation at a high-level of abstraction, which facilitates the modeling of cooperative behaviors. For example, we identified off-the-shelf building blocks, reusable in different application scenarios (e.g., the *take off* action for drones), that can be used together with custom activities to form more complex behaviors. The resulting complexity can be mastered in a standard way via modularisation of the model and the use of BPMN call activities. On the other hand, differently from other similar proposals, we were also driven by practical aspects of the robotics system programming. In this regard, we have taken as reference framework ROS, due to its large adoption by the robotics community. In particular, we have considered the second version of ROS, as it natively supports MRSs via its implementation of the Data Distribution Service (DDS) standard used for intra- and inter-robot communication. This is reflected, for example, on the usage we propose of BPMN events and related event sub-processes for modeling, respectively, the DDS communication mechanism and the event handlers associated with DDS topics via subscription.

Summing up, in this paper we present an original approach for modeling the cooperative behavior of MRSs through the use of the BPMN collaboration diagrams. Its novelty lies in the fact that we defined a disciplined use of the BPMN standard notation for dealing at once with: (i) scenarios with multiple robots, by means of collaboration diagrams; (ii) the topic-

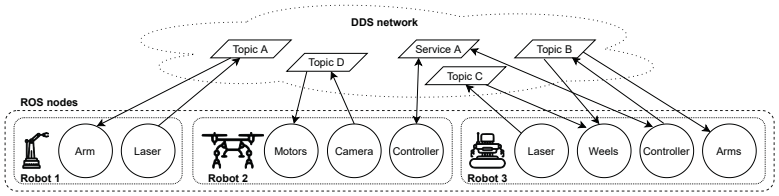

Figure 1: Communication in ROS via DDS.

based communication, by means of signal events and event sub-processes; and (iii) reuse of predefined/custom blocks by means of modularisation and use of call activities. These are distinctive features of MRSs programming that, to the best of our knowledge, are not considered altogether by any other modeling proposal.

The rest of the paper is organized as follows. Section 2 provides notions concerning the technologies at the basis of our approach and presents a running example depicting a smart agriculture scenario. Section 3 proposes our disciplined use of BPMN for describing MRSs, and illustrates the approach at work on the running example. Section 4 compares our solution with respect to the related works. Finally, Section 5 concludes the paper and touches upon directions for future works.

2. Background

This section introduces the background concepts about the development of MRSs by means of the ROS framework and its DDS implementation.

Robot Operating System. ROS (www.ros.org) is one of the most famous and used open-source frameworks for programming robots. It provides an abstraction layer on top of which developers can build robotics applications. ROS is designed as a framework of distributed nodes, which are processes able to perform computations and designed to achieve single purposes, e.g., controlling motors or the ultrasonic sensor. Nodes are embedded in a network, they can communicate with each other by sending and receiving data via topics. *Topics* exploit a publish-subscribe pattern permitting to perform topic-based communication, where a message published over a topic can be read by any number of other nodes subscribed to that topic.

The first version of ROS, i.e., ROS1, is suited for developing single-robot systems, while in recent years a second release, i.e., ROS2, has been proposed with the aim of overcoming ROS1 performance and scalability problems, thus achieving full support for MRSs. ROS2 provides real-time control and exploits a communication mechanism more suitable for inter-robot interactions. This motivates why in this paper we take ROS2 as reference release; from now on, for the sake of presentation, we will refer to ROS2 simply as ROS.

The keystone enabling the development of MRSs in ROS is the OMG standard DDS (www.omg.org/spec/DDS). It enables real-time distributed systems to operate securely as an integrated whole. It introduces a global data space where applications can share information by simply reading and writing data objects. ROS takes advantage of DDS to enable several features. Among them, the one that mostly facilitates the development of MRS is the distributed discovery system that allows ROS nodes to communicate with each other without the use of a master. The communication among ROS nodes via DDS is illustrated in Figure 1.

Before deploying the ROS code into real robots, a good practice is exploiting simulation environments. The reference simulator for ROS is Gazebo (www.gazebosim.org). It permits to rapidly design and test robotics applications into a safe, yet realistic, environment.

Running Scenario. To better reason on how to specify the behavior of a MRS, we present here a running example depicting a smart agriculture scenario.

The cooperation between unmanned ground vehicles, e.g. smart tractors or harvesting robots, and unmanned aerial vehicles, usually called drones, is a promising solution to achieve a fully autonomous and optimized farming system. The proposed application scenario consists of two, or possibly more, tractors and one drone that cooperate to identify and remove weed grass in a farmland to enhance the yields. Both the drone and the tractors are equipped with a controller, enabling computations and communications, a battery, and several sensors and actuators. At the system start-up, the drone is the only robot that can start its behavior: it receives the field's boundaries to inspect, and starts the exploration. During the overflight of the area, the drone uses the camera to recognize weed grass areas and, when found, it sends to the tractors the coordinates. This enacts the tractors, which store the weed grass coordinates and send back to the drone their distance to the weed grass area. The drone can hence elect the closest tractor and notify it. At this point, the selected tractor starts moving towards the field, avoiding possible obstacles. Once it reaches the weed grass area, it activates the blade to cut the weed, and stops its process until it receives a new position from the drone.

3. Disciplined Use of BPMN

In this section, we present our approach for modeling the collective behavior of a MRS in ROS. We first introduce a subset of BPMN elements we selected for describing the robots' mission. Then, we present a list of guidelines driving the use of the selected BPMN elements to compose a collaboration diagram that specifies actions and interactions of each robot in a MRS.

Selection of BPMN elements for MRSs. Figure 2 shows the subset of BPMN elements adopted in our approach. The considered elements are the result of several discussions we did to select, on the huge list of more than 85 elements, those that best fit with the needs of MRSs. The debate was driven, in a top-down fashion, by the modeling activity we did on different application scenarios, and, in a bottom-up fashion, by the experiments we carried out with ROS implementations via the Gazebo simulator.

Let us briefly introduce the BPMN elements resulting from our selection. A collaboration diagram consists of a collection of (possibly multi-instance) *pools*, which contain processes. In their own turn, a *process* consists of activities, gateways and events connected to each other by means of sequence flows, and data objects connected to activities and events by means of data associations. Activities represent one or more pieces of work to be performed within a process. Specifically, a *Call Activity* is a call to another process; this element enables the structuring of (possibly large and complex) models in terms of decoupled reusable processes. A *Script Task* refers to a piece of code to be executed. An *Event Sub-Process* is a sub-process that is not part of the normal flow of its parent process. *Gateways* are used to control the execution flow of a process, by managing parallel activities (AND gateway) and internal/external choices (XOR and Event-based gateways). *Events* represent something that can happen at the beginning, during,

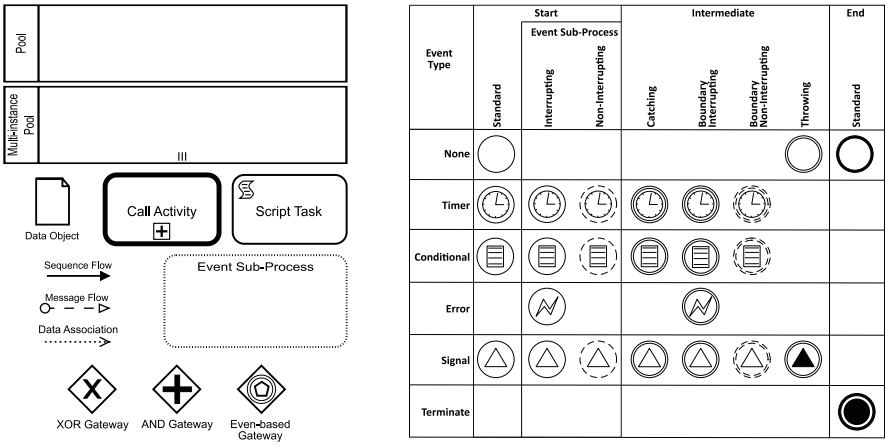

Figure 2: Selected BPMN elements.

or at the close of the process execution (*Start*, *Intermediate*, or *End* events, respectively). *Start* events are alternative entry points for enacting a process. When used in an event sub-process, they can interrupt or not the parent process. *Intermediate* events can happen during the normal flow of the process, by connecting them with sequence flows, or during an activity execution, by attaching them to the activity border. *End* events represent the possible termination of a process or a sub-process. All these events can be further characterized to describe their type and a different semantic meaning. Specifically, we consider *Timer* and *Conditional* events, to react respectively on a time delay or a condition; *Error* events, to throw or catch execution errors; *Signal* events, to describe broadcast and point-to-point communications that may carry data (see [6, p. 235]); and the *Terminate* events to kill all the processes in a pool. Finally, *Data objects* are information and material flowing in and out of activities and events. Notably, we do not consider message flows in our BPMN fragment; in fact, all communications are modeled by means of signals, which better reflect the topic-based communication mechanism based on DDS adopted by ROS for intra- and inter-robot interactions.

Guidelines for MRSs modeling. We provide here a list of guidelines to be used during the modeling of a MRS through a BPMN collaboration. These guidelines define a disciplined use of the BPMN elements introduced before to represent MRS's cooperative behavior while leaving the designer enough freedom to specify almost any MRS mission. Considering the running scenario of Section 2, the BPMN collaboration in Figure 3 is a possible result of our approach; we refer to it to better present the following guidelines.

G1 Robots as pools. Robots involved in a MRS are abstracted by pools, representing the participants in the collaboration.

G1.1 Heterogeneous robots as single-instance pools. Robots that are heterogeneous, i.e., robots of a different kind or robots with different missions, are abstracted by single-instance pools. Considering Figure 3, the drone is represented as a single-instance pool.

G1.2 Homogeneous robots as multi-instance pools. Robots that are homogeneous, i.e., robots of the same kind with the same mission, are abstracted by multi-instance pools. This simplifies the resulting diagram, as a multi-instance pool represents many robots in terms of several instantiations of the same process, avoiding to repeat the same robot process into different pools. Considering Figure 3, the tractors are represented as a multi-instance pool.

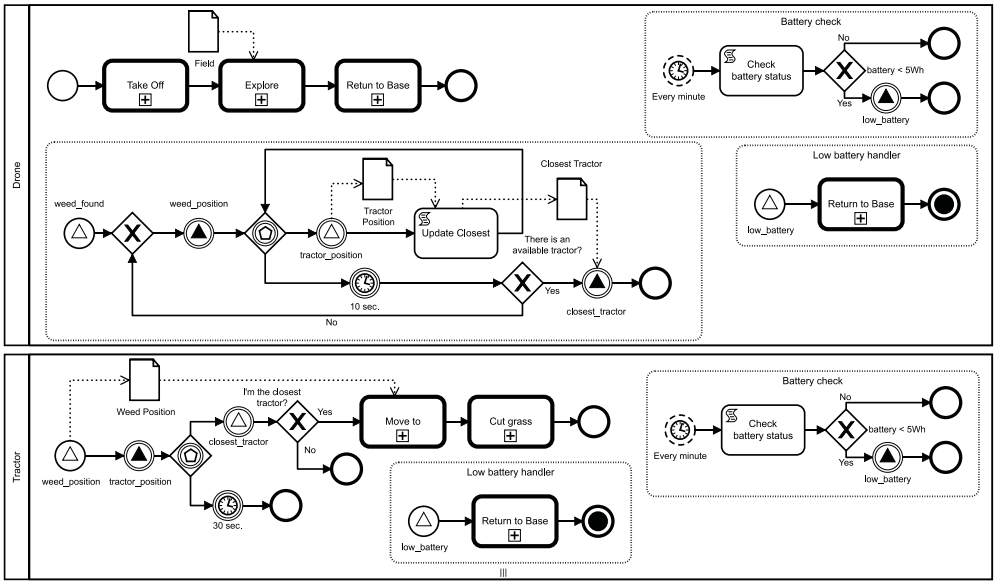

Figure 3: Collaboration diagram of the smart agriculture scenario.

G2 Mission as a process. The mission to be performed by a robot is abstracted by a process diagram within the pool of the considered robot. The process diagram expresses the control flow of the mission. For instance, the process contained into the drone pool in Figure 3 depicts the robot's behavior from the take-off until the return to the base. Activities are related to each other via sequence flows, to define their execution order, and via gateways (see guidelines G2.3-2.5), to control the execution flow.

G2.1 Actions as activities. A robot mission is mainly made by a set of actions; these actions are abstracted by activities within the mission process diagram. Based on the complexity of the action to specify, we distinguish the type of activity to use in the model between call activities and script tasks.

G2.1.1 Complex actions as call activities. Complex actions, e.g., navigation, perception and control, that can be decomposed into several steps and/or reuse already modeled procedures, are abstracted by call activities. Indeed, a call activity can be used for referencing another process diagram or other existing activities. This enables the modularisation of diagrams and lowers their dimension, speeding up the modeling of the MRS through the re-use of already modeled behaviors. Considering Figure 3, the call activity *Return to Base* appears three times in the diagram, indicating that in all these cases the called procedure is implemented in the same way.

G2.1.2 Simple actions as script tasks. Simple actions, which does not require any further decomposition, are abstracted by script tasks. This element refers to a piece of code to be executed by the considered robot. Considering Figure 3, to get the tractor closest to the weed, it is used the script task *Update Closest* that performs simple mathematical operations.

G2.2 Event handlers as event sub-processes. Procedures handling an event (such as the expiration of a timer, the satisfaction of a condition, the occurrence of an error, or the reception of a signal) are abstracted by event sub-processes. Indeed, this element triggers an handling process concurrent to the main process describing the robot mission. Based on the event

type, the main process can be interrupted or not. For example, checking the battery after an amount of time, as depicted in Figure 3, can be performed in parallel with the mission, and thus it is abstracted by a non-interrupting start event.

G2.3 Concurrent behaviors by means of AND gateways. Concurrent behaviors in a robot mission are rendered by means of AND split gateways.

G2.4 Internal choices as XOR gateways. Internal choices in a robot mission are rendered by XOR split gateways. In Figure 3, the decision taken on the basis of the battery charge in the *Battery check* event sub-process is rendered as a XOR gateway, with two possible outcomes.

G2.5 External choices as event-based gateways. Choices driven by events in a robot mission are abstracted by means of event-based gateways. Referring to Figure 3, a tractor waits for a *closest_tractor* signal at most for 30 seconds by means of an event-based gateway followed by two catching events.

G2.6 Missions activation as start events. The beginning of a mission is abstracted by a start event. In more detail, a none start event fires immediately the robot mission. The other event types activate the mission when a specific circumstance occurs. Referring to Figure 3, the main process of the drone contains a none start event triggered at the system start-up, while the tractors are activated only when they receive the *weed_position* signal.

G2.7 Missions shutdown as end events. The end of a mission is abstracted by an end event. The none end event stops only the current execution flow of a mission, while the terminate end event completely stops the mission.

G3 Communication via signal events. Intra- and inter-robot communications, even in presence of a payload, are abstracted by signal events. The sending of a message corresponds to a throwing signal event, and the receiving of a message is made by a catching signal event. The correlation between one or more senders and one or more receivers is made by means of the event name. Considering Figure 3, the signal event *weed_position* corresponds to the same-named topic of which the drone is the only publisher, and the tractors are the subscribers.

G4 Data as data objects. The data used along the execution of the robot's mission are abstracted by data objects. They provide storage in which activities and signal events can read or write information. We explicitly represent in the model, in terms of data objects, only the information used to drive the decision making and the data exchange in the mission execution. Of course, at implementation level, other data will be required. However, since they are confined within low-level pieces of code and do not play any role at the abstraction level of the model, they are omitted. This permits reducing the complexity of the diagram.

BPMN-driven ROS applications. To assess the suitability of our approach to support the development of MRSs, we used some BPMN collaboration models of MRS missions following the above guidelines to drive the implementation of the corresponding ROS applications. We considered the agricultural scenario, modeled in Figure 3, as well as other scenarios concerning smart warehousing and surveillance. The full description of all the scenarios, the BPMN models and the ROS code, are made available online at https://bitbucket.org/proslabteam/multirobotsimulation.

4. Related Works

A big amount of literature has been published on MDE and DSL supporting the development of robotics applications. Indeed, over the last years, researchers have attempted to define notations closer to the robotics domain in order to raise the level of abstraction and increasing the level of automation. This resulted in the proposal of many DSLs we discuss in the following.

Targeting single-robot systems, Bubeck et al. [11] present BRIDE, a platform for component-based modeling of robotic systems. Components are modeled using UML and converted to ROS meta-models, in order to generate a runnable ROS C++ code. Likewise, Brugali et al. [12] propose the HyperFlex toolchain to model software control systems for ROS. HyperFlex differs from BRIDE for the providing of two model transformations for the generation of launch files.

With a different approach, Hue et al. [13] propose a DSL based on AutomationML as modeling language with the aim of generating a runnable C++ code. Similarly, Bardaro et al. [14] use AADL as modeling language to abstract the architecture of ROS-based systems. A code generator is provided to transform models to C++ ROS code. Winiarski et al [15] propose EARL, a DSL based on SysML for designing ROS-based control system. The approach relies on a mathematical method of robot controller specification, using the concept of embodied agent.

Moving on to the textual DSLs for ROS-based systems, Losvik and Rutle [16] and Sharovarskyi [17] proposed two works based on custom modeling languages. The former presents a model-driven approach for the development of MRSs made up of four components: a DSL, a task allocation module, a ROS module and a web interface. The latter proposes an approach using the modeling language Rebeca to check the safety of robots in an environment with obstacles. This work does not provide the code generator module, thus there is not a complete and automatized transition from the models to the robot code. Adam et al. [18] propose DeROS, a DSL targeting the security and safety of ROS-based systems. Another approach has been proposed by Bettini et al. [19] to leverage the powerfulness of the high-level programming language X-Klaim to deal with the distributed nature of robots' software. Although these approaches are adequate to design CPSs, they do not consider inter-robot communication aspects, so there is not a language suitable to abstract the interactions between multiple robots.

We present below the works that model the workflow of robots' activities. Bozhinoski et al. [20] describe a tool for the mission specification for a team of multicopters and the generation of a detailed flight plan, by a custom DSL that is translated into an intermediate language which represents the basic actions of an aerial vehicle. Exploiting the same tool, Ciccozzi et al. [21] define a set of DSLs to specify the civilian missions for unmanned aerial vehicles. The mission specification is done through a web-based graphical interface, that communicates with some ROS-based controllers which send commands to the copters. A use of the BPMN notation is shown by De la Croix et. al [22], which describe the TRACE tool to tailor BPMN to the robotics domain in order to model a sequence of robotic activities. The aim of this approach is to understand what will happen after an unplanned event and check if it will compromise the mission. The authors applied their proposal to a single robot equipped with ROS, which can execute its tasks and autonomously act to unexpected events. Otsu et al. [23] present an application of the TRACE tool to a multi-robot scenario. The mission is specified in a BPMN file, uploaded inside all the vehicles, and each block of the model corresponds to a robot behavior. The system uses the ROS framework with an open-source package that allows a custom message

passing among multiple master nodes. This approach could be avoided and improved by using ROS2 with DDS. Another integration of BPMN standard with an autonomous robot is presented by Rey et al. [24]. The authors automatize a warehouse process by implementing a human-robot cooperation system managed by a web interface able to control the entire process in real-time.

For what concerns the verification of MRSs, López et al. [25] provide an automatic method to verify the task-level control in autonomous robotic systems using Petri nets as input language for a model-checking tool. The Task Definition Language is used to define the workflow of a robot's task and all the constraints that must be satisfied. Each global action is then translated into a Petri net and verified in order to check system fairness and efficiency. The approach is validated in a single vehicle scenario, equipped with ROS. Figat and Zielinski [26] present a methodology for the description of a multi-robot workflow and the related application to a ROS-based system. The model uses Hierarchical Petri Net to abstract the activities of a MRS. The C++ ROS code is generated after detecting deadlocks through simulating the execution.

The analysis of the literature review shows that there is a huge gap in meeting the real-time condition, since all the works mentioned focus on the ROS1 framework that does not meet this requirement. Another drawback of using ROS1 is that the communication is managed by a single master. It creates a centralized system with a single point of failure, making it unsuitable for the development of MRSs applications. Indeed, most of the works do not consider MRSs in their approaches. All these shortcomings are taken into account in the design of our disciplined use of BPMN for MRSs.

5. Concluding Remarks

This paper proposes an approach based on a disciplined use of the BPMN notation for developing MRSs, with a specific reference to their implementation in ROS2 and DDS. We defined a selection of BPMN elements suitable for the specification of MRS missions. The subset of elements we selected has been turned out to be expressive enough for conveniently representing smart agriculture, smart warehousing and smart surveillance scenarios. Moreover, we have defined a set of guidelines for driving the modeling of MRS scenarios.

As future work, we plan to propose a model-driven framework for automatically generating ROS code from models specified via our approach. In addition, we aim at integrating our modeling approach with automated reasoning tools, in order to develop reliable MRSs.

References

[1] T. Arai, E. Pagello, L. E. Parker, Guest editorial advances in multirobot systems, IEEE Transactions on Robotics and Automation 18 (2002) 655–661.

[2] C. W. de Silva, Some issues and applications of multi-robot cooperation, in: Computer Supported Cooperative Work in Design, IEEE, 2016, pp. 2–2.

[3] C. Crick, G. Jay, S. Osentoski, B. Pitzer, O. C. Jenkins, Rosbridge: ROS for Non-ROS Users, volume 100 of *STAR*, Springer, 2017, pp. 493–504.

[4] A. Nordmann, N. Hochgeschwender, S. Wrede, A survey on domain-specific languages in robotics, in: SIMPAR, LNCS 8810, Springer, 2014, pp. 195–206.

[5] E. de Araújo Silva, E. Valentin, J. R. H. Carvalho, R. da Silva Barreto, A survey of Model Driven Engineering in robotics, Computer Languages 62 (2021) 101021.
[6] OMG, Business Process Model and Notation (BPMN) v. 2.0, 2011.
[7] Compagnucci et al., Modelling notations for IoT-aware business processes: A systematic literature review, in: BP-Meet-IoT, LNCS 397, Springer, 2020, pp. 108–121.
[8] F. Corradini, C. Muzi, B. Re, L. Rossi, F. Tiezzi, Formalising and animating multiple instances in BPMN collaborations, Information Systems 101459 (2019).
[9] F. Corradini, C. Muzi, B. Re, F. Tiezzi, L. Rossi, MIDA: multiple instances and data animator, in: BPM Demos, volume 2196, CEUR-WS.org, 2018, pp. 86–90.
[10] Corradini et al., A formal approach to modeling and verification of business process collaborations, SCP 166 (2018) 35–70.
[11] Bubeck et al., BRIDE - A toolchain for framework-independent development of industrial service robot applications, in: ISR, VDE, 2014, pp. 137–142.
[12] D. Brugali, L. Gherardi, Hyperflex: A model driven toolchain for designing and configuring software control systems for autonomous robots, in: ROS, Springer, 2016, pp. 509–534.
[13] Y. Hua, S. Zander, M. Bordignon, B. Hein, From AutomationML to ROS: A model-driven approach for software engineering of industrial robotics using ontological reasoning, in: ETFA, IEEE, 2016, pp. 1–8.
[14] Bardaro et al., From Models to Software Through Automatic Transformations: An AADL to ROS End-to-End Toolchain, in: Robotic Comp., IEEE, 2019, pp. 580–585.
[15] Winiarski et al., EARLl—embodied agent-based robot control systems modelling language, Electronics 9 (2020) 379.
[16] D. S. Losvik, A. Rutle, A domain-specific language for the development of heterogeneous multi-robot systems, in: MODELS-C, IEEE, 2019, pp. 549–558.
[17] K. Sharovarskyi, Software for safe mobile robots with ROS 2 and REBECA, Master's thesis, Mälardalen University, 2020.
[18] Sorin et al., Towards rule-based dynamic safety monitoring for mobile robots, in: SIMPAR, LNCS 8810, Springer, 2014, pp. 207–218.
[19] L. Bettini, K. Bourr, R. Pugliese, F. Tiezzi, Writing Robotics Applications with X-Klaim, in: ISoLA, volume 12477 of *LNCS*, Springer, 2020, pp. 361–379.
[20] Bozhinoski et al., FLYAQ: Enabling non-expert users to specify and generate missions of autonomous multicopters, in: ASE, 2015, pp. 801–806.
[21] Ciccozzi et al., Adopting mde for specifying and executing civilian missions of mobile multi-robot systems, IEEE Access 4 (2016) 6451–6466.
[22] J.-P. de la Croix, G. Lim, Event-driven modeling and execution of robotic activities and contingencies in the Europa lander mission concept using BPMN, in: i-SAIRAS, ESA, 2020.
[23] Otsu et al., Supervised Autonomy for Communication-degraded Subterranean Exploration by a Robot Team, in: AeroConf, IEEE, 2020, pp. 1–9.
[24] R. Rey, M. Corzetto, J. A. Cobano, L. Merino, F. Caballero, Human-robot co-working system for warehouse automation, in: ETFA, IEEE, 2019, pp. 578–585.
[25] J. López, P. Sánchez-Vilariño, R. Sanz, E. Paz, Implementing autonomous driving behaviors using a message driven petri-net framework, Sensors 20 (2020) 449.
[26] M. Figat, C. Zielinski, Robotic system specification methodology based on hierarchical petri nets, IEEE Access 8 (2020) 71617–71627.