Journal of Computer Languages 62 (2021) 101021

Contents lists available at ScienceDirect

Journal of Computer Languages

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

Reviews

A survey of Model Driven Engineering in robotics

Edson de Araújo Silva *, Eduardo Valentin, Jose Reginaldo Hughes Carvalho, Raimundo da Silva Barreto

Institute of Computing, Federal University of Amazonas, Manaus, Amazonas, Brazil

ARTICLE INFO

Keywords:
Robotics
Model-Driven Engineering
Domain-specific language

ABSTRACT

Robots are complex to develop due to the diversity of hardware, software components and the absence of common standards. To deal with these problems, Model-Driven Engineering (MDE) is used to systematize the software development. This work aims to provide a comprehensive overview of existing model-based approaches in robotics. We present a classification of the 63 papers selected comparing past expectations with present achievements, and showing that the self-adaptation of robots has been poorly explored. With advances in machine learning, there has been an increase in the possibilities of using model-based approaches aiming at greater autonomy of robotic software.

Contents

1. Introduction

Nowadays, there are countless opportunities in which to use robots and many tasks that humans prefer to leave to them, some of which are very tedious, e.g., repetitive movements in manufacturing lines or housekeeping, and others that are very dangerous, e.g., deactivating bombs or using a pneumatic drill in an assembly industry. Additionally, some tasks are physically impossible for humans, such as cleaning inside narrow pipes or exploring volcanoes [1,2].

Despite their undeniable benefits, robotic systems consist of different hardware and software components, resulting in highly complex and variable system architectures. As a result, specialists are usually needed to assemble, configure and program such robots [3].

Reuse is one of the main problems for the construction of robotic systems. The large number of components, tools and subsystems, combined with their lack of interoperability, imply that developers do not benefit from existing systems when building new ones [4].

Although the scientific community has developed several robotic centered frameworks, some of which have many functions (e.g., ROS — Robot Operating System), such frameworks act as abstraction layers for functions and tasks for robots. System design is out of their scope

* Corresponding author.
E-mail address: edsonaraujo@ufam.edu.br (E. de Araújo Silva).

https://doi.org/10.1016/j.cola.2020.101021
Received 25 March 2020; Received in revised form 26 December 2020; Accepted 26 December 2020
Available online 8 January 2021

and important development activities are still difficult and prone to errors [5]. These robots are also expected to perform complex tasks in unstructured environments. Thus, the implementation of robotic systems, in many cases, is more an art than a systematic engineering process [P1,P2,P3,P4,P5,P6].

A systematic process is essential for replacing monolithic systems and constructing mature and reusable systems from off-the-shelf components in order to reduce costs and time-to-market and increase robustness [P6]. There is a demand for increasingly complex applications with different requirements such as reuse, flexibility and adaptability [P3].

Model Driven Engineering (MDE) is a software development methodology that focuses on creating and exploring domain models, which are conceptual models of all topics related to a problem-specific domain [6]. The models may be more understandable, platform-independent and translatable into different implementations by automatic code generation [7]. They can abstract the complexity of general-purpose programming languages and allow domain experts to contribute more appropriately to end users [8,9]. Model-oriented development methods are known to address the challenges of building complex heterogeneous systems in areas such as aerospace, telecommunications and automotive, which face similar challenges of integration and modeling as robotics [10].

The research community around the world has been developing model-based approaches, seeking to meet the demands of reuse, interoperability, reliability and cost reduction to circumvent robotic systems implementation problems and to provide developers with a systematic software engineering process.

Herein, we consider important guidance about the advancement of the field of robotics and the expectations in the Robotics Multi-Annual Roadmap 2020 (RMAR) [11], which is a document that is issued and updated by the European Commission on the public side and euRobotics AISBL on the private side. euRobotics AISBL represents the interests of the robotics community in Europe, while SPARC disseminates its intentions through the Strategic Research Agenda (SRA) and updates these documents to reflect new developments and markets.

Hence, this paper aims to highlight motives, challenges, methods and future perspectives of MDE in robotics. The results presented come from a systematic literature review (SLR) focusing on automatic generation of model-based code for robotic applications. This systematic review sought to follow clearly defined protocols, in which criteria are clearly stated before the review is conducted, and is similar to the studies in [12–14]. Therefore, it also aims to motivate SLR studies in the field of MDE, as well as apply them to other types of software systems. This is important to fairly synthesize the existing works and to understand the advances in research in this area.

This SLR is based on a study by Kitchenham and Charters [15], who sought to analyze all primary studies related to the research questions and to synthesize the main empirical evidence. This work differs from the others because it provides results that previous research [16–19] did not consider, e.g., the correlation between the phases of software development and the evolution of MDE described in RMAR [11]. Moreover, none of the related works classified the MDE approaches applied to robotics.

The main contributions of this paper are: (i) to classify model-based approaches for robotic software development; (ii) to justify why these approaches should be adopted; (iii) to outline the current model-based robotic challenges; and (iv) to compare them to expectations set in the past and indicate future directions for this field.

This paper is organized as follows: Section 2 presents prior studies about the topic, Section 3 introduces brief concepts about the Model Driven Engineering paradigm and shows how a classification can be performed, Section 4 details the methodology used in the systematic literature review, and Section 5 describes the results achieved and discusses the answers to the research questions and concludes the paper.

2. Related work

In this section, previous works are presented and compared to this paper to highlight our point of view and how important it is for the community of robotic software engineers.

The work of [16] discusses domain-specific languages (DSL) used to design, simulate and program robotics systems. This study focuses on assessing the availability and usability of DSLs to formulate their experimental or system hypotheses and generate reproducible experiments. One problem that it highlights is the fact that compatibility and reuse of different DSLs and approaches are still an issue. Furthermore, how different approaches to documentation, evaluation and platform-dependency affect the availability and usability of a DSL are discussed.

The study of [17] researched the techniques that are used to develop robotic systems and focused on component-based development approaches, service-based architecture and model-oriented software development. An increasing trend was identified regarding the application of these techniques. In 2012, when this research was carried out, only preliminary proposals were found for applying model-driven development to robotics.

The modeling structures used to verify and validate complex automation and control systems are confirmed in the study [18]. This article focuses on distributed control projects. Research has shown that in these environments, functional block approaches are still preferred. Additionally, a lack of validation and verification environments in the automation area has also been reported.

Heineck et al. [19] carried out a Systematic Literature Review (SLR) with the objective of identifying how MDE techniques have supported the robotics domain by verifying the characteristics that techniques have in supporting the software development process. They also observed other characteristics of interest, e.g., the requirements types addressed by the classification as functional and non-functional. The behavioral issue (reactive/deliberative) was also analyzed, helping to adopt a standard and techniques that address social issues to support the development of projects involving social robots and effective robotics.

The study of [20] aimed to identify current MDE approaches in robotics and analyze how these approaches provide general advantages related to modeling and how effective they are in meeting the specific requirements of a robot. In the same study, an approach to automatic adaptive robotic systems (SafeRobots) is presented, which is part of the selected primary studies. This study found that many of the robotic requirements were not addressed and only 4 studies were selected for the survey.

Unlike all the previously mentioned studies, the present study applies a formal model described in the literature to classify primary studies. We sought to correlate the stages of software development and the selected approaches, as it is important to show the progress in the MDE area for the development phases of robotic software. Furthermore, it is also possible to check the progress of selected approaches within the expectations proposed in the RMAR Roadmap [11].

3. Background in model driven engineering

This section presents background about the research presented herein. It begins with a brief description of MDE concepts that are important for understanding the classification in the Table 1. This classification is based on the conceptual model proposed by Silva [21].

MDE has emerged as a paradigm shift from code-based software development to model-based development [22]. Such an approach promotes the systematization and automation of constructing software artifacts. The models represent part of the functionality, structure and behavior of a system [23]. A metamodel includes the set of concepts needed to describe a domain at a given level of abstraction along with the relationships between them. The model is defined in terms

Table 1
Classification of model-driven approaches: concrete approaches robotics.

	Model driven approaches	Models		Transformations		Meta modeling languages	Domain	Tool Support
		Levels	Language	Types	Languages			
1	[P7]	PSM	DSL (MAUVE)	M2T	OROCOS (C++)	–	Ground Robot	–
2	[P8]	PIM	DSL (RobotArch)	–	–	–	Ground Robot	–
3	[P9]	PSM	UML (ACORD-CS)	M2T	JAVA, shell scripts, text files	UML	Ground Robot	BOUML
4	[P10]	PSM	DSL (PYTF)	M2T	ROS (Python)	–	Simulation	–
5	[P4]	PIM	DSL (MSL)	M2T	Declarative language	ECORE	Air Robot	EMF[a] Xtext
6	[P11]	PIM, PSM	DSL (AAS)	M2T	C++	–	Ground Robot	–
7	[P12]	PIM, PSM	DSL (DSEP)	M2T	ROS (C++)	–	Simulation	–
8	[P13]	PSM	DSL	M2T	XML	ECORE	Ground Robot	EMF[a], Xtext
9	[P14]	PIM, PSM	UML (SysML, HLA)	M2M, M2T	JAVA	ECORE	Air Robot	EMF[a], QVT
10	[P15]	PSM	DSL (MAUVE)	M2T	C, C++	–	Ground Robot	–
11	[P16]	PSM	DSL (Interface Designer)	M2T	Byte Code	–	Robotic Arm	–
12	[P17]	PIM, PSM	DSL (RDIS)	M2M, M2T	JSON, ROS (Python)	AToM	Simulation	AToM3
13	[P2]	PIM, PSM	DSL (MML, RL, BL)	M2M, M2T	ROS (C++)	ECORE	Air Robot, Underwater Robots	EMF[a]
14	[P3]	PIM, PSM	UML (TARMAN)	M2M, M2T	XMI, ROS (C++)	UML	Robotic Arm	–
15	[P18]	PIM	DSL (ArchGenTool)	M2T	TEXTO (YAML, WORD)	–	Service	Python
16	[P19]	PIM, PSM	UML	M2T	ROS, OROCOS (C++)	ECORE	Robotic Arm	EMF[a]
17	[P20]	PIM, PSM	UML (MML, QBL)	M2M, M2T	SITL (C++)	ECORE	Air Robot	EMF[a], ACCELEO
18	[P21]	PSM	DSL	M2T	ROS (C++)	ECORE	Ground Robot	EMF[a]
19	[P22]	PIM, PSM	DSL	M2M, M2T	ROS (C++)	ECORE	Ground Robot	EMF[a], HAROS
20	[P23]	PIM, PSM	UML (AMoDE-RT)	M2M, M2T	JAVA	ECORE	Robotic Arm	EMF[a], GenERTiCA, MARTE, AT4U
21	[P24]	PIM, PSM	DSL (CRALA)	M2M, M2T	–	ECORE	Ground Robot	EMF[a]
22	[P25]	PIM, PSM	UML (SmartMARS)	M2M, M2T	–	–	Ground Robot	–
23	[P26]	PSM	DSL	M2T	C++	ECORE	Simulation	Papyrus
24	[P27]	PSM	DSL (DICE-R)	M2T	Hasselt	–	Robotic Arm	–
25	[P28]	PSM	DSL	M2T	–	GOPPRR	Robotic Arm	DVMEx IDE, MetaEdit+
26	[P29]	PIM	Ontological Semantics	–	–	–	Robotic Arm	–
27	[P30]	PSM	DSL (iserveU)	M2T	JAVA	ECORE	Ground Robot	EMF[a]
28	[P31]	PSM	DSL (RBDML)	M2T	XML	GOPPRR	Health Care	MetaEdit+
29	[P32]	PSM	DSL (AutomationML)	M2T	ROS (C++, Python), XML	ECORE	Robotic Arm	EMF[a]
30	[P33]	PSM	DSL (RoboticSpec)	M2T	JAVA	–	Robotic Arm	–
31	[P34]	PIM, PSM	DSL (RoboticSpec)	M2M, M2T	–	ECORE	Robotic Arm	EMF[a], ATL, Xtext
32	[P5]	PIM, PSM	Property Graphs	M2M, M2T	–	–	Air Robot	–
33	[P35]	PIM, PSM	DSL (PROMISSE)	M2M, M2T	JAVA	ECORE	Ground Robot	EMF[a], Xtext, Sirius
34	[P36]	PIM, PSM	DSL (HyperFlex)	M2M, M2T	ROS (C++), OROCOS (C++)	ECORE	Ground Robot	EMF[a]
35	[P37]	PIM, PSM	DSL (BRIDE)	M2T	ROS (C++), ROS (Python), XML	ECORE	Robotic Arm	EMF[a]

(*continued on next page*)

Table 1 (*continued*).

	Model driven approaches	Models		Transformations		Meta modeling languages	Domain	Tool Support
		Levels	Language	Types	Languages			
36	[P38]	PSM	UML (RobotML)	M2T	–	ECORE	Humanoid	EMF[a], Papyrus, Sophia, Safety Architect
37	[P39]	PSM	UML (M3L)	M2T	–	–	Modular Robot	–
38	[P40]	PIM, PSM	DSL (KSE)	M2T	NAOQi (C++)	ECORE	Humanoid	EMF[a]
39	[P41]	PSM	UML	M2T	ROS (C++)	–	Ground Robot	RTEdge, RoseRT
40	[P6]	PIM, PSM	UML (SmartMars)	M2M, M2T	ROS (C++)	ECORE	Service	EMF[a], Papyrus
41	[P42]	PIM, PSM	UML (Prometheus)	M2M, M2T	JAVA	ECORE	Ground Robot	EMF[a]
42	[P43]	PSM	DSL (CoSiMA)	M2T	OROCOS (C++)	MPS Base Language	Humanoid	JetBrains MPS
43	[P44]	PIM, PSM	UML	M2T	MontiArc Automaton	UML	Health Care	–
44	[P45]	PIM, PSM	DSL	M2M, M2T	SCXML, C++	MPS Base Language	Robotic Arm	JetBrains MPS
45	[P46]	PIM, PSM	DSL (SCP-RASQ)	M2T	XML	–	Robotic Arm	–
46	[P47]	PIM, PSM	DSLs	M2M, M2T	NAOqi (C++), SCXML6, PlantUML	MPS Base Language	Humanoid	JetBrains MPS
47	[P48]	PIM, PSM	DSL	M2T	ROS (C++), OROCOS (C++)	ECORE	Robotic Arm	EMF[a]
48	[P49]	PSM	DSL (RHEA)	M2T	ROS (C++), JAVA	–	Simulation	–
49	[P50]	PSM	UML (RoboChart)	M2T	C++	ECORE	Simulation	EMF[a], Xtext, Sirius
50	[P51]	PSM	DSL (AMSA)	M2T	ROS (C++)	ECORE	Ground Robot	EMF[a], Xtext, Sirius, Acceleo
51	[P52]	PSM	UML (RobotML)	M2T	–	ECORE	Ground Robot	EMF[a], Papyrus
52	[P53]	PIM, PSM	UML (RobotML)	M2T	OROCOS (C++)	ECORE	Ground Robot	EMF[a], Papyrus
53	[P54]	PSM	DSL (FAMILE)	M2T	C++	ECORE	Robotic Arm	Xtext
54	[P55]	PSM	Ontological Semantics (RSAW)	M2T	–	–	Ground Robot	–
55	[P56]	PIM, PSM	DSL (SSML)	M2T	ROS (C++)	ECORE	Ground Robot	EMF[a], Xtext
56	[P57]	PSM	DSL (Salty)	M2T	Python, Java e C++	–	Simulation	–
57	[P58]	PSM	DSL (HyperFlex and RPSL)	M2T	ROS (C++), OROCOS (C++)	ECORE	Simulation	EMF[a]
58	[P59]	PIM, PSM	DSL	M2M, M2T	JAVA, ROS(Python)	MontiCore	Ground Robot	MontiArc Automaton
59	[P1]	PIM, PSM	DSL	M2M, M2T	ROS (C++), OROCOS (C++)	ECORE	Ground Robot	EMF[a]
60	[P60]	PSM	DSL	M2T	ROS (C++)	ECORE	Simulation	EMF[a], Xtext
61	[P61]	PSM	DSL (ALGAE)	M2T	C, Python, GLOC	–	Underwater Robots	–
62	[P62]	PIM, PSM	DSL (V3STUDIO)	–	–	ECORE	Robotic Arm	EMF[a]
63	[P63]	PIM, PSM	DSL (vTSL)	M2M, M2T	ROS (C++)	MPS Base Language	Ground Robot	JetBrains MPS

[a]Eclipse Modeling Framework.

of formal metamodels. Model transformations, commonly described as metamodel mappings, allow the automatic transformation and evolution of models into other models, defined at any level of abstraction or any textual format, such as code [24].

This approach helps experts shift their focus from implementation space to problem space [20]. With an increase in the abstraction layer provided by the model-based paradigm, it is possible to describe applications independently from the software and hardware platform. Different models can represent system elements in different domain views and their relationships [25].

According to the level of abstraction, MDE requires a fixed domain bounded by a specific field of knowledge. This domain could be the field of robotics, which may be even more specialized in subdomains as aerial or ground robotics.

Domain-specific language (DSL) aims to do two things: (i) raise the level of abstraction beyond programming through specific problem domain; and (ii) generate code in a chosen programming language from these high-level specifications [26].

Unlike General Purpose Modeling Languages (GPML), DSL are typically composed of a set of concepts and graphical notations that are

close to the respective domain field [27]. This provides the application with principle of multi-point modeling or principle of separation of interests, which state that a complex system is best defined by multiple views, considering static and dynamic aspects [28,29].

Therefore, a modeling language provides one or more viewpoints classified according to abstraction and perspective properties. For example, in the abstraction dimension, a viewpoint can be classified using the MDA terminology [30], such as Platform Independent Model (PIM) or Platform Specific Model (PSM).

To generate software artifacts, Gerard et al. [31] consider two types of transformations in MDE to increase the abstraction level in development, using models constructed with terms similar to the domain concepts of the problem in question. The model-to-text transformations (M2T) generate or produce software artifacts: source code, XML and other text files. Model-to-model (M2M) transformations translate models into another set of models, typically closer to the solution domain or to meet the needs of different stakeholders [21].

Silva [21] proposed a conceptual model that was used to classify the approaches selected in this paper. The model defines a hierarchical structure of model driven approaches, each one at a level of abstraction with respective generalization and specialization relationships. The most abstract term (MDE) is at the top of this hierarchy, while there are concrete MD approaches at the bottom. The abstraction level understands the levels of high, middle and low (concrete). In this article, the classification is based on the lowest level: concrete.

Some aspects were considered to support the analysis and comparison of MD approaches at the concrete level, namely: abstraction level, software engineering disciplines, models, transformations, metamodeling languages, application domain and tool support. The aspects considered in the classification are detailed below:

- The “models" aspect defines the levels of abstraction (PIM or PSM) and the modeling languages implemented by each approach.
- The “Transformations" aspect defines the types (M2M or M2T) and languages used by each approach to support the transformation model.
- The “meta modeling languages" aspect defines the technologies used in the modeling languages.
- The “domain" aspect defines the application domains of each approach, which only deals with concrete approaches that are specific within the robotics field.
- Finally, the “tool support" aspect defines the type of tool supported by each approach.

4. Systematic literature review method

This section presents the investigation protocol and threats to the validity of the research.

4.1. Protocol

An SLR requires a comprehensive and unbiased research plan to ensure the integrity of the acquired data. In this work, we applied the steps proposed in Kitchenham and Charters [15] to conduct the review. The main research question that guided this work was: “What are the model-based approaches to software development in the field of robotics?" Due to the broad nature of this issue, we divided it into four other questions to better understand and organize the results.

- RQ1: What are the model-driven methods, domains and tools adopted in software development in robotics?
- RQ2: What are the reasons for using the model driven engineering approach in robotics?
- RQ3: What are the challenges of using models in robotic software development?
- RQ4: What are the future perspectives for using models in robotics compared to the previous expectations of Roadmap RMAR?

We used the ACMDigital Library, DBLP, IEEE Xplorer, ScienceDirect, Scopus, Springer and Web of Science digital libraries as sources for this research, including the period from 2000 to the present. Based on the research question, we formulated the following search strings: ((“domain-specific language" OR “DSML" OR “DSL" OR “model driven engineering" OR “MDE" OR “model driven architecture" OR “MDA") AND (“*robotic*") AND (“technique" OR “method" OR “tool" OR “approach" OR “application” OR “technology”)). The term “robotic" appears with an asterisk since we aimed to include all robotic fields.

The inclusion criteria were: Papers presenting techniques, methods or approaches of Model Driven Engineering applied to robotics (CI1) and; Presenting complete results (CI2)

The exclusion criteria were: Duplicate papers (CE1); Papers not available on the web (CE2); Publications with concepts only (CE3); Editorial research, prefaces, paper summaries, interviews, news, reviews, correspondence, discussions, comments, readers’ letters and tutorial summaries, workshops, panels and poster sessions (CE4).

We prepared two filters to select the most relevant research studies. The first filter consisted of reading the titles and summaries of all papers, resulting in 199 papers according to the inclusion and exclusion criteria. For the second filter, all 199 papers from the first filter were read entirely to refine the review. As a result, 63 papers were selected as high relevance in the field of models applied to robotics. Fig. 1 shows the steps based on the selection and exclusion criteria.

4.2. Validity threats of the survey

In the literature review, there were possible internal and conclusion threats that could have resulted from how we constructed the search string, the digital libraries that we chose and the articles retrieved [32].

Regarding the construction of the research sequence, completion threats were mitigated through the sequence construction in order to include articles related to MDE and its closest concepts, e.g., MDA, DSL, DSML. In addition, the word “robotics” combined with the other words guaranteed the coverage of articles related to the research field.

Concerning the digital libraries used, conclusion threats were mitigated by choosing the major digital libraries recommended for systematic SE literature reviews [33].

In terms of the recovered articles, completion threats were mitigated by classifying the recovered articles based on their relevance to the problem domain. This was done by removing unrelated articles, checking the title and summary of each one that was retrieved and excluding articles that were out of context. Afterwards, we only considered articles from authentic sources, e.g., journals, conferences and edited scientific books. These two steps reduced threats to conclusion validity, as we wanted to avoid including any unwanted articles in our review.

We mitigated internal threats to validity in our literature review by conducting an MDE research in robotics to support our findings. The research focused on obtaining industry feedback about MDE approaches applied to robotics.

5. Results and discussion

The results presented in this section were selected from 63 primary sources.

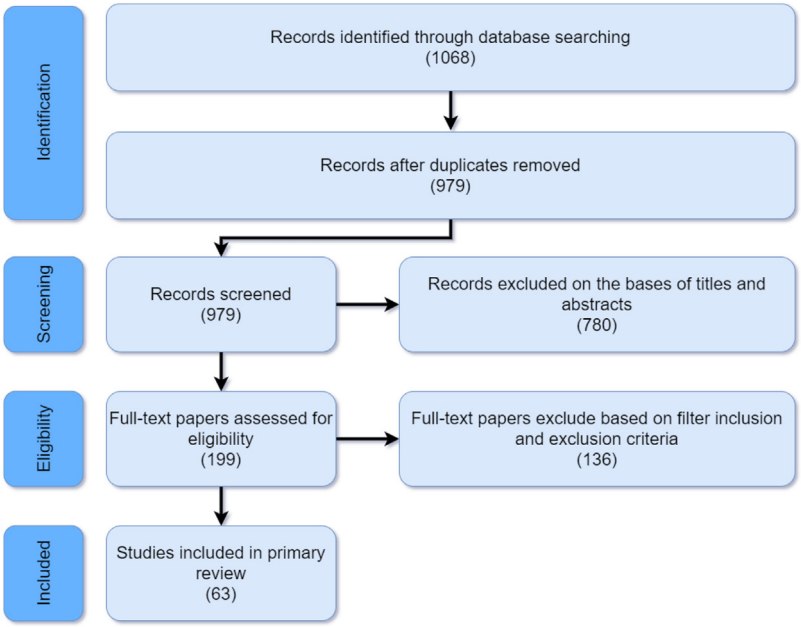

Fig. 1. Study selection.

5.1. Research Question 1

RQ1: What are the model-driven methods, domains and tools adopted in robotics software development?

Table 1 contains the model-driven methods (model, transformation, metamodeling), domain and supporting tools extracted from the papers selected in the survey. The data in such table is based on the conceptual model introduced in [21], which proposes a division between high level, medium level and concrete level approaches. High level approaches are those that model the concept of development, e.g., the MDA and MDE. Middle level approaches are metamodeling tools, or "MetaTools" in short. Concrete-level approaches are those developed for application in a particular field of activity. We restricted the present classification to the concrete level, since this paper focuses on implementing robotic systems at a low level of abstraction.

- Model-driven methods:
  - **Model levels**: can be based on the Model Driven Architecture (MDA). The MDA process can be divided into three phases: (i) build a PIM, which is an independent high-tech model; (ii) the PIM model is transformed into one or more PSMs, which is/are lower level and describe(s) the system according to a particular deployment technology; (iii) finally, the source code from each PSM is generated. PIM document the functionality and behavior of an application, thus separating the specification to the technology-specific code. In this way, they allow interoperability within and beyond platform limits [30]. The results show that most studies (about 47.62%) use the PIM model along with the PSM model in their approaches. This seeks to ensure interoperability, productivity and portability, as well as generates part or all of the code needed for the target robotic platform. About 44.44% focus on serving a specific platform only, thus using the PSM model only. The rest (7.94%) only apply the PIM model, that is, they only focus on the model and do not support code generation.
  - **Model languages**: In this context, the most common practice is using metamodeling techniques [34]. For example, in the case of OMG, modeling languages are most commonly defined through UML by creating a profile or directly using the MOF language [35], which is considered a DSL. Of the 63 approaches presented, 73.02% prioritized the use of DSL over the use of UML profiles (26.98%). This corroborates with what was discovered in [9], which states that modeling languages with UML require significant customization before the languages can be put into practice. Therefore, the most successful MDE practice is driven from the ground up.
  - **Transformation types**: MDE is fundamentally based on the model automation and automation is key to productivity [31]. This transformation can be Model to Text (M2T) or Model to Model (M2M). Code generation is the most common technique for model to text (M2T) transformations and there are several solutions and techniques, as discussed in Czarnecki et al. [36]. As Fig. 2 shows, most of the approaches selected (63.49% at the expense of 31.75% using both transformations) in this review only support this kind of transformation. This means that most approaches are interested in generating code for specific robotics platforms. Model-to-model (M2M) transformation generates a target model from a source model [31]. M2M transformations are specified by means of distinct languages, as well as by specialized model transformation languages, for different purposes and with different modeling paradigms, such as QVT, ATL, and VIATRA [21].
    QVT and ATL, for example, are based on relational approaches, where the basic idea is to state the source and target elements specified by using constraints. On the other hand, VIATRA is a graph-transformation-based approach. Therefore, these approaches operate on typed, attributed, and labeled graphs, which are specifically designed to represent UML-like models [31].
  - **Transformation languages**: Refers to the target language that is generated after the transformations. More than 50% of the approaches generate code in the C++ language, followed by Java and Python with 15.87% and 12.7%, respectively. Language C++ is the most common language for robot programming [37]. This is because robots have to deal with many sensors and their real-time data flows in order to react to their environments. C++ offers a combination of speed, since it is close to the hardware, and expressiveness that is needed for writing complex algorithms in robotics.

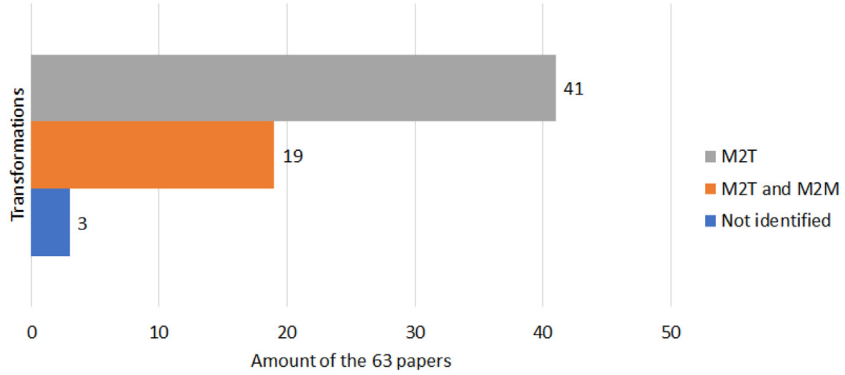

Fig. 2. Types of transformations supported.

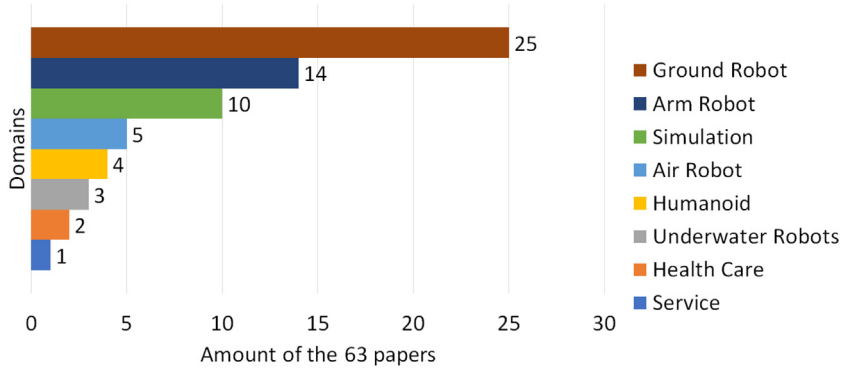

Fig. 3. Domain.

– **Metamodeling languages**: are adopted to create metamodels. Metamodels are important because they capture a coherent set of concepts that serve as the basis for representation of abstractions of real systems through models [38]. The most used metamodeling language is Ecore, which is a language that expresses other models by leveraging their constructions [39]. Ecore is also its own metamodel, that is, Ecore is defined in its own terms. As shown in Table 1, most approaches use the Ecore metalanguage (50.79%) as the basis for their metamodels. It was also clear that the MPS Base Language (6.35%) has been growing in number of uses (5 in total).

- **Domains**: Fig. 3 shows where MDE has been applied to robotics. It is worth noting that most papers in the literature (60%) have applied their methods to ground robot and arm robotic applications. Simulation has also been applied to MDE for robotics (15%). Such technique allows the structure, characteristics and functions of a robotic system to be studied at different levels of detail. Aerial and humanoid robots add even more complexity. The application of MDE in aerial and humanoid vehicles is still low (14.06%), given the current development in approach based models.
- **Tools support**: many of these approaches required tools support that streamlined development. Fig. 4 shows the tool support used to develop the approaches presented. The Eclipse Modeling Framework (EMF), which was most widely used among these tools (30 of the total), is an Eclipse-based modeling framework and a code generation feature for building tools and other applications based on a structured data model [39]. In addition to being the first in its category, EMF has the advantages of its resources for the development of abstract syntax and concrete syntax, M2M and M2T transformations [40]. It also supports the fusion of code generated with handwritten code [41]. The Eclipse IDE is well accepted by developers around the world and familiarity with the platform Eclipse can be a relevant factor for its use with MDE. JetBrains MPS has been gaining notoriety, with 5 approaches using it to develop DSLs. It is a metaprogramming system that has been developed by JetBrains. MPS is a tool to design DSL that uses projectional editing which allows users to overcome the limits of language parsers and build DSL editors, such as those with tables and diagrams [42]. MetaEdit+ [43] is a tool for defining modeling language as a metamodel, including its concepts, rules, notations and generators, and was used in 3 approaches presented.

5.2. Research Question 2

RQ2: What are the reasons for using the model driven engineering approach in robotics?

Table 2 presents the main reasons for adopting MDE in robotics found in the systematic literature review and the papers addressing them, which highlight the reduction of complexity, reuse, variability, qualified labor and reliability. In other areas besides robotics, MDE is a mature technology [P2,P45]. One of the reasons described in RMAR [11] is: "*Generally, there are no system development processes, highlighted by the lack of general architecture models and methods. This results in the need for skill in building robotic systems instead of following established engineering processes*". Such lack of software engineering processes in robotic development implies several problems.

Complexity. The complexity of software development increases in the presence of heterogeneous hardware and by adding several computational approaches to solve classic problems such as planning, perception and control. The need for observation as a real-time constraint are examples of factors that further increase the overall complexity. In order to minimize this problem, the studies selected herein used MDE so developers and interested parties could use abstractions closer to the domain than to the concepts of knowledge. This reduces complexity and improves communication between stakeholders. In addition, a

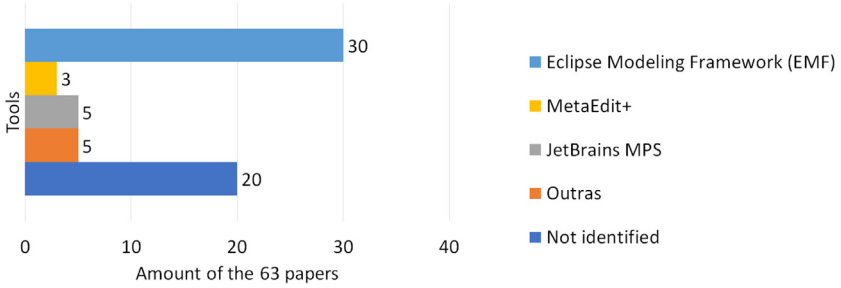

Fig. 4. Tools support.

Table 2
Main reasons for adopting MDE in robotics.

Reasons	Papers
Complexity reduction	[P1,P2,P3,P5,P6,P44,P45,P8,P13], [P14,P16,P19,P23,P25,P26,P28], [P29,P33,P34,P37,P38,P41,P42], [P46,P52,P54,P60,P61,P63]
Reusability	[P1,P3,P6,P59,P44,P11,P12,P13,P15], [P17,P18,P19,P20,P21,P22,P23,P32], [P34,P37,P40,P47,P48,P51,P53,P58,P62]
Variability	[P2,P4,P6,P12,P19,P25,P28,P30], [P33,P34,P36,P49,P50,P52,P54], [P55,P57,P58,P60,P61]
Qualified Labor	[P1,P2,P4,P20,P25,P28,P30,P32], [P35,P37,P53,P54,P60]
High Cost	[P8,P12,P25,P29,P34,P37,P43,P46], [P52,P53,P62]
Safety	[P7,P15,P20,P27,P33,P38,P41,P43,P63]
Interoperability	[P1,P3,P44,P39,P42,P49,P53]
Run-time adaptation	[P3,P4,P5,P8,P9,P10,P55]
Technology uncoupling	[P11,P31,P35,P36,P51,P60]
Dependability	[P1,P26,P43,P58,P62]
Integration	[P11,P13]

model is designed at an abstract architectural level, which can be used automatically to generate the program code with predefined models embedded with rules and standards.

Reusability. Robotic systems comprise several components and include the establishment of interconnections between components, the definition of data and control flows, and the specification of the behavioral aspects of these components [P32]. The challenge is to select, integrate, and configure a coherent set of components that provide the necessary functionality considering their mutual dependencies and architectural incompatibilities. Most robotic software that has been developed is heavily dependent on existing hardware specifications and their limitations. By increasing the level of abstraction using MDE principles development is enhanced to facilitate rapid prototyping and reuse. The main point observed in the selected studies regarding reuse is that abstraction is elevated to the level of technology independence. This ensures that, through model transformations of more specific models, components can be reused in other projects without the need for or minimal manual intervention in the code.

Variability. According to [P54], robotic systems may vary in terms of hardware, software and environment. The MDE provides a means to capture and manage the variability of a specific application domain. In this context, model-oriented product line approaches manage the variability that is introduced in robotics by different hardware and algorithms. In addition, it allows automatic configuration as well as automatic verification of existing DSL programs [P60].

Skilled Labor. Robotic software developers must master highly heterogeneous technologies to integrate them consistently and effectively. In addition, they are required to learn about mechanical properties of robots and electronics for control logic and actuators [P28]. MDE facilitates the specification, understanding and maintenance of complex robotic problems through abstraction via models and automation via model transformations. These models go from documentation only or development guidelines to executable models with their own semantics and syntax.

Dependability and High Cost. In general, the software is prone to errors during the development phase. The most common problems motivating the emergence of MDE are poor quality of the software developed [44], software that does not meet specifications [45], projects that do not follow the schedule or budget [46] and maintenance becomes expensive with project growth [47]. These problems can be caused by isolated development, monolithic software, low level abstraction languages, immature software development processes and an increasing demand for software in society [48].

Interoperability. Despite the existence of ample robotic software, they rarely interoperate due to their built-in dependencies on specific hardware or software platforms [P54,P13,P53,P18,25].

Safety. Safe behavior is essential for robots to collaborate with each other and between humans. Robotics safety can be related to aspects such as: collision prevention, adaptation of movements in the presence of demand, human interaction, fault detection and tolerance. MDE makes it possible to compose security rules and validate them both in the development stages and in the execution time. The studies [P7,P15, P27] deal with security by performing the compliance analysis with real-time restrictions in the robot's software. The other studies only deal with security in the stages of software development and testing.

Runtime adaptation. Robots that work in less structured or unstructured environments must be able to deal with situations that they are not programmed for. The knowledge representation provided by

MDE can provide reasoning for the transformation of models at run time as it is treated in the studies of [P8,P25,P55]. Neural networks can also be applied with MDE to provide the ability to learn and adapt as presented in [P10].

Technology Uncoupling. To address these issues, the literature review revealed that researchers invested significant efforts in models and tools designed to help software development in robotics [P28]. Through DSLs, they provide abstraction and notation close to the application domain and, consequently, provide better expressiveness [34,49,50]. There is a need to develop metamodels, which are meaningful representations of robotic systems,in order to reduce the close coupling between robotic concepts and implementation technologies [P31].

Integration. The complexity of software development and integration increases significantly in the presence of heterogeneous hardware and when aggregating diverse computational approaches to solve classic functional problems such as planning, perception and control [P44, P13,25]. Furthermore, non-functional requirements are usually hard to deal with. The need to observe real-time constraints is one factor that further increases the overall complexity [P1].

Software engineers need to continuously improve methods and tools to develop and maintain high-quality robotics applications [P28]. The MDE promotes this goal as it encapsulates all or part of the knowledge necessary to solve problems in a domain. In summary, this survey shows that the motives for using models in the field of robotics are the complexity, difficulty in reusing software components, variability and the need for specialized human resources. These factors account for 64% of the main problems reported in robotic systems development. It is worth noting that each cited paper may contain more than one problem.

5.3. Research Question 3

RQ3: What are the challenges of using models in the development of robotic software?

The construction of robotic software systems requires the ability to master several fields of knowledge. This alone is a huge challenge for design, development and implementation. The construction of development methodologies based on MDE is even more difficult when having to extract heterogeneous concepts abstracting from the technology and proposing concepts closer to the application domain. Among the challenges reported in the selected papers, the ones most frequently highlighted by the researchers are adaptation at runtime, complexity, the usability of the models, decoupling, standardization of concept and protocols, scalability and modularity.

Since the study of autonomous mobile robots has been developing, robot autonomy is increasingly required, ranging from the original needs of some manual autonomous operations to fully autonomous robots. Adaptation is the ability to adapt their behavior when the environment around the application changes. From the articles analyzed, we conclude that 39.6% of the selected studies [P4,P5, P6,P7,P8,P9,P15,P20,P23,P25,P27,P28,P30,P33,P34,P35,P36,P41,P43, P49,P51,P55,P56,P58,P61] have worked or will work on any of the aspects of adaptation at run time. Such need is evident, especially when it comes to real-time systems.

To overcome the challenge of adaptation, some studies [P9,P27, P36,P43,P49,P51,P56] used context-sensitive systems methodologies. As an example, the study of [P36] built an approach that presupposes the definition of points of variation (resources, algorithms, control strategies, coordination policies, etc.). Depending on the context, the control system dynamically chooses the appropriate variants to execute these points of variation. In the study [P51], they use parameterization and scenario modeling resources with models in order to evaluate different control settings in relation to different environmental contexts. The others used behavioral properties [P15,P28,P30,P33,P41] such as DSL Mauve [P15], which provides tools to analyze the real-time correction of the architecture and to verify the validity of some behavioral properties.

Complexity is an intrinsic factor in robotics. The main focus of the research community of software development based on models for robots is the automatic generation of software artifacts [P5], including documentation, source code and configuration files. However, to achieve this goal it is necessary to build a conceptual basis in the application domain to capture any and all important characteristics for the representation of a formal model through a concrete syntax.

The robotic domain is an interdisciplinary and knowledge-intensive task since robots can have many purposes, forms and functions [P36]. This variability, which can be applied to hardware, software and even to the robot's operating environment, increases development complexity. Additionally, discovering the shared abstract concepts that allow the structure and device to communicate becomes a great challenge. The following works [P6,P44,P45,P14,P21,P23,P25,P31,P32,P36,P41, P42,P47,P49,P51,P53,P54,P62] propose an increase in the level of abstraction and separation of concerns to manage this growing complexity.

Due to the complexity of developing robotic systems, the usability of support languages and tools must be considered when creating MDE approaches. Providing an easy and user-friendly programming environment allows different stakeholders with different educational backgrounds (for example, domain experts, software developers) to analyze, model, implement, monitor, and debug the robot's logic. Such concern was frequently found in the studies [P8,P10,P16,P3,P20,P22, P34,P35,P40,P42,P46,P48,P50,P52,P53,P60], including about 25.40% of the 63 works.

As stated by [P35], DSLs must be developed seeking to maximize their simplicity, maintain their expressiveness and enable rigorous and precise specification. Building model editors that provide support for an MDE methodology integrated with well-known development platforms or standards can help to shorten the learning curve for the tool. This is observed in studies that used the Eclipse framework (see Table 1). The MDE methodology makes programming easier because low-level details are hidden behind abstractions that are easier to manage.

Another challenge in the development of robotic systems is the modular composition. The idea is that a software system should be broken down into parts that overlap as little as possible. Since the coding of software systems began, it has been understood that the systems must be as modular as possible. However, it is necessary to follow a methodology when decomposing a system into modules and is usually done focused on software quality metrics such as coupling and cohesion. Coupling is the degree of dependence between two modules and low coupling is always desirable. On the other hand, cohesion measures how closely related the set of functions performed by a module is. Of the 63 studies, 14.28% [P10,P13,P17,P19,P30,P31,P48, P49,P62] support the decoupling between concept and technology to allow modeling in a declarative and independent structure.

Today, ad-hoc development often leads to components that are difficult to compose and reuse because they violate the basic principles of software engineering, e.g., the separation of interests [P1]. There is no structure that roboticists use decisively, nor are there standardized protocols for the development of robotic software such as protocols for communicating with robots. In order to avoid ambiguities in a model-based approach, it is necessary to come to a consensus about the available concepts, terminology and their meaning [P13]. The standardization of concepts and protocols in robotics was discussed in 14.28% [P1,P3,P11,P13,P29,P38,P48,P56,P57] of the studies. Without standardization, each particular DSL will not be reused beyond the limits of the research group that developed it, and will not lead to portability beyond the set of robots and controllers applied in that group [P48].

Reusing the experience of the application domain in the form of tasks, objectives, actors and domain models with different robots requires that the approach be scalable in a way that does not require much effort [P44]. The scalability of approaches was also mentioned in the studies [P2,P37,P49,P63].

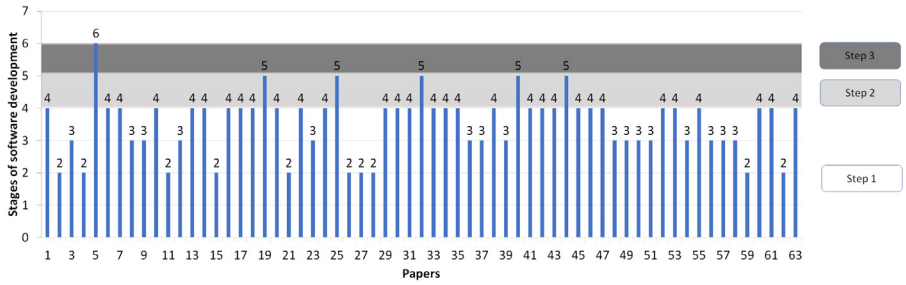

Fig. 5. Correlation between the stages of software development and the steps described in the Robotics Multi-Annual Roadmap 2020.

5.4. Research Question 4

RQ4: What are the future perspectives for using models in robotics compared to the previous expectations of Roadmap RMAR?

Advanced robots operating in complex environments have to deal with many different situations and contingencies. At run-time, the robot has to manage a large amount of different execution variants that cannot be predicted or fully programmed. Therefore, they cannot be analyzed and verified entirely during design and implementation [P6].

The next step is to provide robots with explicit representations of knowledge and let them reason about their software at run-time. This is one of the many key ingredients for autonomous and intelligent robots that are able to adapt to their tasks in dynamic environments [P5].

This run-time adaptation is set out in the RMAR [11], where it is assumed that models can be used in robotics in three steps. Step 1 assumes that the models are used to define missions by people in the design phase. Step 2 requires robots to use run-time models to interact with the environment and to explain what is happening. Step 3 means that robots adapt and improve models to redefine what they are doing based on their learning.

In this survey, a reader can compare the evolution in the research area regarding analyzed approaches with the expectations set to a few years ago.

Based on the use stages of the models defined in Roadmap 2020, it was possible to correlate them with the development phases of software defined in Sommerville et al. [51]. These phases of software development are divided into: (1) Requirements Analysis, (2) Project, (3) Implementation and Testing, (4) Integration, (5) Execution and (6) Maintenance. Thus, it is possible to verify to what extent the analyzed approaches extend into the phases of software development. This is important because it clearly demonstrates the current state of the art in the development of robotic software through models. This opens up a range of possibilities for future research in relation to the expectations of the industry.

Fig. 5 shows the correlation between the development phases defined in Sommerville et al. [51] and the 3 steps described in the RMAR [11]. Step 1 of the roadmap comprises phases 1, 2, 3 and 4. Step 2 comprises the phase (5) Execution, dealing with the adaptation at run-time. For step 3, which is equivalent to stage (6), Maintenance is understood as the approaches that manage to reach a power of self-adaptation so that the robot can modify its models from its own knowledge.

Most approaches only address the design phase (1) Requirements Analysis up to phase (4) - Integration, which represent 90.47% of all the selected studies. Fig. 5 shows that there are few articles that have dealt with the use of runtime models. Decision making based on the conditions of the environment is an example of this type of use.

Among the selected studies, only the FLYAQ platform extension approach claims to have reached Step 3. The work of Dragule et al. [P4] extended the FLYAQ platform, which is a platform designed to enable non-expert domain users to program missions for a team of multi-copters. In this study, FLYAQ was improved to support adaptive robots at the mission level, meaning that robots can change their behavior to perform missions under varying circumstances. This was achieved by including a declarative language called MSL (Mission Specification Language). Therefore, it is only necessary to specify goals that must be achieved and which restrictions should not be violated.

MSL is intended to specify properties of a mission that allow users to set time constraints declaratively. These restrictions are based on temporal patterns that can be absence (something must never occur), universality (something must always occur), existence (something must eventually occur), limited existence (something must occur at most n times), precedence (occurrence of P must be preceded by an occurrence of Q), or response (an occurrence of P must be followed by an occurrence of Q).

Step 2 was only achieved in 5 selected studies. The platform developed in Djukic et al. [P28] was able to describe activity at run time. The platform also aims to generate test scenarios and documentation during model-level debugging. The application scenario adopts the IGUS Robolink D industrial robot arm, which is divided into three DSL. The first is used to specify the topological characteristics of a hand or arm. The second DSL is created for specifying the environment in which a robot performs its actions (palette, canvas, bowl for cleaning and a paint brush). The third DSL is used to specify control logic, sensors with input signals and motors/actuators performing motions. For this task, they have used the IEC 61131–3 language aimed at specifying function block diagrams.

A motion framework, which allows the dynamic parameterization of actions and movements (parameters, dimension, rotate) based on various curves in run time and dependence on the sensor, is used to specify a robot's tasks. The arithmetic of the motion framework is similar to the electro-magnetic wave modulations, which allows the robot arm to make movements and actions in the curves resulting from the modulations. In this way, events or influences from the real environment can be integrated and manipulated, e.g., gravitation and inertia. The trajectories are calculated at run-time when the palette is moved or rotated, and also when two arms with movable root joints are used.

Nordmann et al. [P45] introduce a conceptual framework and a metamodel for motion control architectures based on motion primitives. The metamodel is comprised of two DSLs that model two aspects: (i) structural aspects of motion architectures that focus on so-called Adaptive Modules that model motion primitives as dynamic systems with machine learning capabilities; and (ii) behavioral aspects for the coordination of motion control systems. The flexibility to adapt motion primitives to different situations and environments is accomplished through machine learning techniques. A qualitative validation was built of n Tracking Controllers, including recurrent neural networks as Adaptive Modules and a scaling Transformation. Superposition of outputs from the Tracking Controller resulted in the intended mixture of controllers.

SMARTMARS [P6] is a UML Profile based on Papyrus UML. Several models are created at design-time by the developer and used at run-time by the robot. Thus, both the system engineer and the robot reason using the parameters, constraints, properties and other information explained in the same models. Such access is through different representations. Knowledge about how to transform models between the

different representations with partial information has to be considered, in addition to which representation is required. The results that can be re-incorporated are encoded in the tools (design-time) and in the action-plots (run-time). The models comprise variation points that are bound during the development phase. These variation points are linked to the pre-programmed components that are executed according to previously configured factors.

Another important observation is that so far only 1 work has dealt with Stage 3 proposed in RMAR [11], which states that robots will be able to adapt and improve models based on their own learning. This demonstrates a research area that has not yet been explored, and is the next step for the evolution of robotic software development through models.

In light of the above, it is worth noting that there has been a significant advance within the perspectives of RMAR [11] in Step 1, given the increased use of this approach since 2012. Regarding Step 2 and 3, there have been few publications (9.5%), however, the results of such studies are promising.

6. Conclusion

MDE offers the possibility of systematically and simultaneously concentrating different levels of abstraction in which developers can work to improve the quality of systems in terms of security, dependability and reusability, reduce variability and complexity, and promote the reuse of software and hardware components.

This survey shows that software engineering applications in robotics are complex endeavors that require multiple domain solutions, as well as their successful integration. Current programming has a fairly low level of parameterization of actions and activities, causing major failures in obtaining robotic applications.

Failures introduced into the system during development and unexpected or unobserved problems while interacting with the environment during operation adversely affect dependability. The use of formal methodologies such as MDE can leverage the development of software for robots and provide several benefits as increased productivity, maximized compatibility between systems (via reuse of standardized models), communication between individuals and teams working on the system (through standardized terminology and good practices), and simplification of the design process (through models of recurring design patterns in the application domain).

Due to the increasing interest in the use of MDE applied to the field of robotics, this paper provides an analysis of the benefits, such as reduced complexity and allowed reusability. It presents the challenges found in the development through this paradigm, such as the adaptation in runtime, usability of the models and decoupling between technologies. Present applications of the selected approaches demonstrate the direction of research with 60% focusing on terrestrial robots and the robotic arm. This paper also provides a systematic overview of evidence regarding model-based approaches, detailing the methodologies used (see Section 5.1) and the problems (see Section 5.2) faced when using this paradigm.

The evolution of the MDE supports an increase in its application within the development phases of robotic software that extends from the requirements analysis phase to the execution of the software phase, where it is possible for the robot to adapt itself according to its interactions with the environment. At runtime, this adaptation means that the robot has a control logic programmed for certain situations, previously mapped by software engineers, to follow alternative codes according to certain parameters. This ability to adapt was verified in only 6 of the 63 papers presented herein.

Based on the analysis presented, robotic evolution involves giving robots knowledge about their representation models and letting them reason about such models at run-time, increasing their perception about their actions through machine learning techniques. This is important because in most missions performed by robots there are unforeseeable and emergent situations during execution, therefore, they must be resilient to such situations to guarantee execution and mission fulfillment. This self-adaptation is fundamental for autonomous and intelligent robots to face adversities in their tasks and dynamic environments.

Overall, the state of the art has advanced in relation to the first two stages of RMAR [11]. However, the contributions in Steps 2 and 3 are below expectations and need further studies to support them, while Step 3 represents an opportunity for greater autonomy in robotics.

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.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nìvel Superior - Brasil (CAPES) - Finance Code 001. According to Article 48 of Decree n° 6.008/2006, this study was funded by Samsung Electronics of Amazonia Ltda, Brazil, under the terms of Federal Law n° 8.387/1991, through agreement n° 003, signed with ICOMP/UFAM, by the Amazonas State Research Support Foundation (FAPEAM), Brazil through projects 122/2018 (UNIVERSAL) and 003/2018 (PROINT).

References

[1] Yujun Ma, Yixue Hao, Yongfeng Qian, Min Chen, Cloud-assisted humanoid robotics for affective interaction, in: Control, Automation and Robotics (ICCAR), 2016 2nd International Conference on, IEEE, 2016, pp. 15–19.

[2] Zhihui Du, Weiqiang Yang, Yinong Chen, Xin Sun, Xiaoying Wang, Chen Xu, Design of a robot cloud center, in: Autonomous Decentralized Systems (ISADS), 2011 10th International Symposium on, IEEE, 2011, pp. 269–275.

[3] Claudia Pons, Gabriela Pérez, Roxana Giandini, Gabriel Baum, Applying MDA and OMG robotic specification for developing robotic systems, in: International Conference on System Analysis and Modeling, Springer, 2016, pp. 51–67.

[4] Víctor Juan Expósito Jiménez, Herwig Zeiner, A domain specific language for robot programming in the wood industry-A practical example, in: ICINCO (2), 2017, pp. 549–555.

[5] Azamat Shakhimardanov, Nico Hochgeschwender, Gerhard K. Kraetzschmar, Component models in robotics software, in: Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop, 2010, pp. 82–87.

[6] Douglas C. Schmidt, Model-driven engineering, Comput.-IEEE Comput. Soc. 39 (2) (2006) 25.

[7] Benoît Lelandais, Marie-Pierre Oudot, Benoit Combemale, Applying model-driven engineering to high-performance computing: Experience report, lessons learned, and remaining challenges, J. Comput. Lang. 55 (2019) 100919.

[8] Geylani Kardas, Model-driven development of multi-agent systems: a survey and evaluation, Knowl. Eng. Rev. (2013) 1.

[9] Jon Whittle, John Hutchinson, Mark Rouncefield, The state of practice in model-driven engineering, IEEE Softw. 31 (3) (2013) 79–85.

[10] Deniz Akdur, Vahid Garousi, Onur Demirörs, A survey on modeling and model-driven engineering practices in the embedded software industry, J. Syst. Archit. 91 (2018) 62–82.

[11] SPARC Robotics, Robotics 2020 multi-annual roadmap for robotics in Europe, in: SPARC Robotics, EU-Robotics AISBL, The Hauge, The Netherlands, accessed Feb, vol. 5, 2018, 2016.

[12] Muhammad Azeem Akbar, Jun Sang, Arif Ali Khan, Sajjad Mahmood, Syed Furqan Qadri, Haibo Hu, Hong Xiang, et al., Success factors influencing requirements change management process in global software development, J. Comput. Lang. 51 (2019) 112–130.

[13] Mustafa Abshir Mohamed, Moharram Challenger, Geylani Kardas, Applications of model-driven engineering in cyber-physical systems: A systematic mapping, J. Comput. Lang. (2020) 100972.

[14] Abbas Javan Jafari, Abbas Rasoolzadegan, Security patterns: A systematic mapping study, J. Comput. Lang. 56 (2020) 100938.

[15] Barbara Kitchenham, Stuart M. Charters, Guidelines for performing systematic literature reviews in software engineering, 2007.

[16] Arne Nordmann, Nico Hochgeschwender, Sebastian Wrede, A survey on domain-specific languages in robotics, in: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2014, pp. 195–206.

[17] Claudia Pons, Roxana Giandini, Gabriela Arévalo, A systematic review of applying modern software engineering techniques to developing robotic systems, Ing. Invest. 32 (1) (2012) 58–63.
[18] Chia-Han Yang, Valeriy Vyatkin, Cheng Pang, Model-driven development of control software for distributed automation: a survey and an approach, IEEE Trans. Syst. Man Cybern. Syst. 44 (3) (2013) 292–305.
[19] Tiago Heineck, Enyo Gonçalves, Aêda Sousa, Marcos Oliveira, Jaelson Castro, Model-driven development in robotics domain: a systematic literature review, in: 2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), IEEE, 2016, pp. 151–160.
[20] Arunkumar Ramaswamy, Bruno Monsuez, Adriana Tapus, Model-driven software development approaches in robotics research, in: Proceedings of the 6th International Workshop on Modeling in Software Engineering, ACM, 2014, pp. 43–48.
[21] Alberto Rodrigues da Silva, Model-driven engineering: A survey supported by the unified conceptual model, Comput. Lang. Syst. Struct. 43 (2015) 139–155.
[22] Thomas Stahl, Markus Voelter, Krzysztof Czarnecki, Model-Driven Software Development: Technology, Engineering, Management, John Wiley & Sons, Inc., 2006.
[23] Sybren Deelstra, Marco Sinnema, Jilles Van Gurp, Jan Bosch, Model driven architecture as approach to manage variability in software product families, in: Proc. of Model Driven Architecture: Foundations and Applications, 2003, pp. 109–114.
[24] Tom Mens, Pieter Van Gorp, A taxonomy of model transformation, Electron. Notes Theor. Comput. Sci. 152 (2006) 125–142.
[25] E. Estévez, A. Sánchez García, J. Gámez García, J. Gómez Ortega, A novel model-based approach to support development cycle of robotic arm applications, IFAC Proc. Vol. 47 (3) (2014) 3465–3470.
[26] Steven Kelly, Juha-Pekka Tolvanen, Domain-Specific Modeling: Enabling Full Code Generation, John Wiley & Sons, 2008.
[27] Nordmann Arne, Hochgeschwender Nico, Wigand Dennis, Wrede Sebastian, A survey on domain-specific modeling and languages in robotics, J. Softw. Eng. Robot. 7 (1) (2016) 75–99.
[28] Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language User Guide (Addison-Wesley Object Technology Series), vol. 1, Addison-Wesley Professional, 1999.
[29] Marco Brambilla, Jordi Cabot, Manuel Wimmer, Model-driven software engineering in practice, Synth. Lect. Softw. Eng. 1 (1) (2012) 1–182.
[30] MDA OMG, Guide version 1.0.1, in: Object Management Group, 62, 2003, p. 34.
[31] Sebastien Gerard, Jean-Philippe Babau, Joel Champeau, Model Driven Engineering for Distributed Real-Time Embedded Systems, Wiley-IEEE Press, 2010.
[32] Nasser Mustafa, Yvan Labiche, Dave Towey, Mitigating threats to validity in empirical software engineering: A traceability case study, in: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), vol. 2, IEEE, 2019, pp. 324–329.
[33] Sunil Nair, Jose Luis De La Vara, Sagar Sen, A review of traceability research at the requirements engineering conference re@ 21, in: 2013 21st IEEE International Requirements Engineering Conference (RE), IEEE, 2013, pp. 222–229.
[34] Marjan Mernik, Jan Heering, Anthony M. Sloane, When and how to develop domain-specific languages, ACM Comput. Surv. (CSUR) 37 (4) (2005) 316–344.
[35] Bran Selic, A systematic approach to domain-specific language design using UML, in: Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC'07. 10th IEEE International Symposium on, IEEE, 2007, pp. 2–9.
[36] Krzysztof Czarnecki, Ulrich W. Eisenecker, Krysztof Czarnecki, Generative Programming: Methods, Tools, and Applications, vol. 16, Addison Wesley Reading, 2000.
[37] Lentin Joseph, Robot Operating System (ROS) for Absolute Beginners, Springer, 2018.
[38] Federico Ciccozzi, Davide Di Ruscio, Ivano Malavolta, Patrizio Pelliccione, Jana Tumova, Engineering the software of robotic systems, in: Software Engineering Companion (ICSE-C), 2017 IEEE/ACM 39th International Conference on, IEEE, 2017, pp. 507–508.
[39] Dave Steinberg, Frank Budinsky, Ed Merks, Marcelo Paternostro, EMF: Eclipse Modeling Framework, Pearson Education, 2008.
[40] Richard C. Gronback, Eclipse Modeling Project: a Domain-Specific Language (DSL) Toolkit, Pearson Education, 2009.
[41] E. Merks, Eclipse modeling framework–Interview with ed merks, 2010, jaxenter.com.
[42] JetBrains MPS, MPS: Meta programing system the domain-specific language creator by JetBrains", 2019, URL: https://www.jetbrains.com/mps/.
[43] Juha-Pekka Tolvanen, Matti Rossi, MetaEdit+: defining and using domain-specific modeling languages and code generators, in: Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, ACM, 2003, pp. 92–93.
[44] Stephen H. Kan, Metrics and Models in Software Quality Engineering, Addison-Wesley Longman Publishing Co., Inc., 2002.
[45] Cliff Jones, Peter O'Hearn, Jim Woodcock, Verified software: A grand challenge, Computer 39 (4) (2006) 93–95.
[46] Lawrence H. Putnam, Ware Myers, Measures for Excellence: Reliable Software on Time, Within Budget, Prentice Hall Professional Technical Reference, 1991.
[47] Rajiv D. Banker, Srikant M. Datar, Chris F. Kemerer, Dani Zweig, Software complexity and maintenance costs, Commun. ACM 36 (11) (1993) 81–95.
[48] Jack Greenfield, Keith Short, Software factories: assembling applications with patterns, models, frameworks and tools, in: Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, ACM, 2003, pp. 16–27.
[49] Thomas Degueule, Benoit Combemale, Arnaud Blouin, Olivier Barais, Jean-Marc Jézéquel, Melange: A meta-language for modular and reusable development of dsls, in: Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering, ACM, 2015, pp. 25–36.
[50] Christian Schäfer, Thomas Kuhn, Mario Trapp, A pattern-based approach to DSL development, in: Proceedings of the Compilation of the Co-Located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11, ACM, 2011, pp. 39–46.
[51] Ian Sommerville, Software Engineering, internat. ed., Pearson, Boston, 2011.

Primary studies

[P1] Herman Bruyninckx, Markus Klotzbücher, Nico Hochgeschwender, Gerhard Kraetzschmar, Luca Gherardi, Davide Brugali, The BRICS component model: a model-based development paradigm for complex robotics software systems, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, ACM, 2013, pp. 1758–1764.
[P2] Federico Ciccozzi, Davide Di Ruscio, Ivano Malavolta, Patrizio Pelliccione, Adopting MDE for specifying and executing civilian missions of mobile multi-robot systems, IEEE Access 4 (2016) 6451–6466.
[P3] Elisabet Estevez, Alejandro Sanchez Garcia, Javier Gamez Garcia, Juan Gomez Ortega, An UML based approach for designing and coding automatically robotic arm platforms, Rev. Iber. Automa. Inf. Ind. 14 (1) (2017) 82–93.
[P4] Swaib Dragule, Bart Meyers, Patrizio Pelliccione, A generated property specification language for resilient multirobot missions, in: International Workshop on Software Engineering for Resilient Systems, Springer, 2017, pp. 45–61.
[P5] Nico Hochgeschwender, Sven Schneider, Holger Voos, Herman Bruyninckx, Gerhard K. Kraetzschmar, Graph-based software knowledge: Storage and semantic querying of domain models for run-time adaptation, in: Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), IEEE International Conference on, IEEE, 2016, pp. 83–90.
[P6] Andreas Steck, Alex Lotz, Christian Schlegel, Model-driven engineering and run-time model-usage in service robotics, in: ACM SIGPLAN Notices, vol. 47, 2011, pp. 73–82.
[P7] Nicolas Gobillot, Charles Lesire, David Doose, A design and analysis methodology for component-based real-time architectures of autonomous systems, J. Intell. Robot. Syst. 96 (1) (2019) 123–138.
[P8] Lei Zhang, Huaxi Zhang, Zheng Fang, René Zapata, Marianne Huchard, A domain specific architecture description language for autonomous mobile robots, in: 2012 IEEE International Conference on Information and Automation, IEEE, 2012, pp. 283–288.
[P9] Rossano P. Pinto, Eleri Cardozo, Paulo R.S.L. Coelho, Eliane G. Guimarães, A domain-independent middleware framework for context-aware applications, in: Proceedings of the 6th International Workshop on Adaptive and Reflective Middleware: Held at the ACM/IFIP/USENIX International Middleware Conference, 2007, pp. 1–6.
[P10] Georg Hinkel, Henning Groenda, Sebastian Krach, Lorenzo Vannucci, Oliver Denninger, Nino Cauli, Stefan Ulbrich, Arne Roennau, Egidio Falotico, Marc-Oliver Gewaltig, et al., A framework for coupled simulations of robots and spiking neuronal networks, J. Intell. Robot. Syst. 85 (1) (2017) 71–91.
[P11] Philipp A. Baer, Roland Reichle, Michael Zapf, Thomas Weise, Kurt Geihs, A generative approach to the development of autonomous robot software, in: Engineering of Autonomic and Autonomous Systems, 2007. EASe'07. Fourth IEEE International Workshop on, IEEE, 2007, pp. 43–52.
[P12] Subhrojyoti Roy Chaudhuri, Amar Banerjee, N. Swaminathan, Venkatesh Choppella, Arpan Pal, P. Balamurali, A knowledge centric approach to conceptualizing robotic solutions, in: Proceedings of the 12th Innovations on Software Engineering Conference (Formerly Known As India Software Engineering Conference), 2019, pp. 1–11.
[P13] Nico Hochgeschwender, Luca Gherardi, Azamat Shakhirmardanov, Gerhard K. Kraetzschmar, Davide Brugali, Herman Bruyninckx, A model-based approach to software deployment in robotics, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 3907–3914.
[P14] Paolo Bocciarelli, Andrea D'Ambrogio, Alberto Falcone, Alfredo Garro, Andrea Giglio, A model-driven approach to enable the simulation of complex systems on distributed architectures, Simulation 95 (12) (2019) 1185–1211.
[P15] Nicolas Gobillot, Charles Lesire, David Doose, A modeling framework for software architecture specification and validation, in: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2014, pp. 303–314.

[P16] Michael Moser, Michael Pfeiffer, Josef Pichler, A novel domain-specific language for the robot welding automation domain, in: Emerging Technology and Factory Automation (ETFA), 2014 IEEE, IEEE, 2014, pp. 1–6.

[P17] Paul Kilgo, Eugene Syriani, Monica Anderson, A visual modeling language for RDIS and ROS nodes using atom 3, in: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2012, pp. 125–136.

[P18] Emanuele Ruffaldi, Ioannis Kostavelis, Dimitrios Giakoumis, Dimitrios Tzovaras, ArchGenTool: A system-independent collaborative tool for robotic architecture design, in: Methods and Models in Automation and Robotics (MMAR), 2016 21st International Conference on, IEEE, 2016, pp. 7–12.

[P19] E. Estévez, Alejandro Sánchez García, Javier Gámez García, Juan Gómez Ortega, ART(2)ool: a model-driven framework to generate target code for robot handling tasks, Int. J. Adv. Manuf. Technol. 97 (1–4) (2018) 1195–1207.

[P20] Davide Di Ruscio, Ivano Malavolta, Patrizio Pelliccione, Massimo Tivoli, Automatic generation of detailed flight plans from high-level mission descriptions, in: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, 2016, pp. 45–55.

[P21] Nadia Hammoudeh Garcia, Mathias Lüdtke, Sitar Kortik, Björn Kahl, Mirko Bordignon, Bootstrapping mde development from ros manual code-part 1: Metamodeling, in: 2019 Third IEEE International Conference on Robotic Computing (IRC), IEEE, 2019, pp. 329–336.

[P22] Nadia Hammoudeh Garcia, Ludovic Deval, Mathias Lüdtke, Andre Santos, Björn Kahl, Mirko Bordignon, Bootstrapping MDE development from ROS manual code-part 2: Model generation, in: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS), IEEE, 2019, pp. 95–105.

[P23] Marco Aurélio Wehrmeister, Edison Pignaton de Freitas, Alécio Pedro Delazari Binotto, Carlos Eduardo Pereira, Combining aspects and object-orientation in model-driven engineering for distributed industrial mechatronics systems, Mechatronics 24 (7) (2014) 844–865.

[P24] Huaxi Zhang, Lei Zhang, Zheng Fang, Harold Trannois, Marianne Huchard, René Zapata, CRALA: Towards a domain specific language of architecture-centric cloud robotics, in: Information and Automation, 2015 IEEE International Conference on, IEEE, 2015, pp. 456–461.

[P25] Christian Schlegel, Andreas Steck, Davide Brugali, Alois Knoll, Design abstraction and processes in robotics: From code-driven to model-driven engineering, in: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2010, pp. 324–335.

[P26] Garazi Juez Uriagereka, Estibaliz Amparan, Cristina Martinez Martinez, Jabier Martinez, Aurelien Ibanez, Matteo Morelli, Ansgar Radermacher, Huascar Espinoza, Design-time safety assessment of robotic systems using fault injection simulation in a model-driven approach, in: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), IEEE, 2019, pp. 577–586.

[P27] Jan Van den Bergh, Kris Luyten, DICE-R: defining human-robot interaction with composite events, in: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ACM, 2017, pp. 117–122.

[P28] Verislav Djukić, Aleksandar Popović, Juha-Pekka Tolvanen, Domain-specific modeling for robotics: from language construction to ready-made controllers and end-user applications, in: Proceedings of the 3rd Workshop on Model-Driven Robot Software Engineering, ACM, 2016, pp. 47–54.

[P29] Stefan Zander, Nadia Ahmed, Yingbing Hua, Empowering the model-driven engineering of robotic applications using ontological semantics and reasoning, in: Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, SCITEPRESS-Science and Technology Publications, Lda, 2016, pp. 192–198.

[P30] Kai Adam, Arvid Butting, Oliver Kautz, Bernhard Rumpe, Andreas Wortmann, Executing robot task models in dynamic environments, in: MODELS (Satellite Events), 2017, pp. 95–101.

[P31] Chandan Datta, Elizabeth Broadbent, Bruce A. MacDonald, Formalizing the specifications of a domain-specific language for authoring behaviour of personal service robots, in: Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), IEEE International Conference on, IEEE, 2016, pp. 98–103.

[P32] Yingbing Hua, Stefan Zander, Mirko Bordignon, Björn Hein, From automationml to ROS: A model-driven approach for software engineering of industrial robotics using ontological reasoning, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2016, pp. 1–8.

[P33] Rui Wang, Yingxia Wei, Houbing Song, Yu Jiang, Yong Guan, Xiaoyu Song, Xiaojuan Li, From offline towards real-time verification for robot systems, IEEE Trans. Ind. Inf. 14 (4) (2018) 1712–1721.

[P34] Pedro Sánchez, Diego Alonso, José Miguel Morales, Pedro Javier Navarro, From Teleo-Reactive specifications to architectural components: A model-driven approach, J. Syst. Softw. 85 (11) (2012) 2504–2518.

[P35] Sergio García, Patrizio Pelliccione, Claudio Menghi, Thorsten Berger, Tomas Bures, High-level mission specification for multiple robots, in: Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering, 2019, pp. 127–140.

[P36] Davide Brugali, Luca Gherardi, Hyperflex: A model driven toolchain for designing and configuring software control systems for autonomous robots, in: Robot Operating System (ROS), Springer, 2016, pp. 509–534.

[P37] Alexander Bubeck, Benjamin Maidel, Felipe Garcia Lopez, Model driven engineering for the implementation of user roles in industrial service robot applications, Proc. Technol. 15 (2014) 605–612.

[P38] Nataliya Yakymets, M. Sango, S. Dhouib, R. Gelin, Model-based engineering, safety analysis and risk assessment for personal care robots, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 6136–6141.

[P39] Mirko Bordignon, Ulrik Pagh Schultz, Kasper Stoy, Model-based kinematics generation for modular mechatronic toolkits, ACM SIGPLAN Not. 46 (2) (2010) 157–166.

[P40] Alexandros Paraschos, Nikolaos I. Spanoudakis, Michail G. Lagoudakis, Model-driven behavior specification for robotic teams, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 171–178.

[P41] A. Beaulieu, S.N. Givigi, D. Ouellet, J.T. Turner, Model-driven development architectures to solve complex autonomous robotics problems, IEEE Syst. J. 12 (2) (2018) 1404–1413.

[P42] José M. Gascueña, Elena Navarro, Antonio Fernández-Caballero, Model-driven engineering techniques for the development of multi-agent systems, Eng. Appl. Artif. Intell. 25 (1) (2012) 159–173.

[P43] Dennis Leroy Wigand, Sebastian Wrede, Model-driven scheduling of real-time tasks for robotics systems, in: 2019 Third IEEE International Conference on Robotic Computing (IRC), IEEE, 2019, pp. 46–53.

[P44] Kai Adam, Arvid Butting, Robert Heim, Oliver Kautz, Bernhard Rumpe, Andreas Wortmann, Model-driven separation of concerns for service robotics, in: Proceedings of the International Workshop on Domain-Specific Modeling, ACM, 2016, pp. 22–27.

[P45] Arne Nordmann, Sebastian Wrede, Jochen Steil, Modeling of movement control architectures based on motion primitives using domain-specific languages, in: Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE, 2015, pp. 5032–5039.

[P46] Johan Sund Laursen, Lars-Peter Ellekilde, Ulrik Pagh Schultz, Modelling reversible execution of robotic assembly, Robotica 36 (5) (2018) 625–654.

[P47] D.L. Wigand, A. Nordmann, M. Goerlich, S. Wrede, Modularization of domain-specific languages for extensible component-based robotic systems, in: 2017 First IEEE International Conference on Robotic Computing (IRC), 2017, pp. 164–171, http://dx.doi.org/10.1109/IRC.2017.34.

[P48] Markus Klotzbücher, Ruben Smits, Herman Bruyninckx, Joris De Schutter, Reusable hybrid force-velocity controlled motion specifications with executable domain specific languages, in: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, 2011, pp. 4684–4689.

[P49] Orestis Melkonian, Angelos Charalambidis, RHEA: a reactive, heterogeneous, extensible, and abstract framework for dataflow programming, in: Proceedings of the 5th ACM SIGPLAN International Workshop on Reactive and Event-Based Languages and Systems, 2018, pp. 11–20.

[P50] Alvaro Miyazawa, Pedro Ribeiro, Wei Li, Ana Cavalcanti, Jon Timmis, Jim Woodcock, Robochart: modelling and verification of the functional behaviour of robotic applications, Softw. Syst. Model. 18 (5) (2019) 3097–3149.

[P51] Hamza El Baccouri, Goulven Guillou, Jean-Philippe Babau, Robotic system testing with AMSA framework, in: MODELS Workshops, 2018, pp. 316–325.

[P52] Selma Kchir, Saadia Dhouib, Jérémie Tatibouet, Baptiste Gradoussoff, Max Da Silva Simoes, RobotML for industrial robots: Design and simulation of manipulation scenarios, in: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2016, pp. 1–8.

[P53] Saadia Dhouib, Selma Kchir, Serge Stinckwich, Tewfik Ziadi, Mikal Ziane, Robotml, a domain-specific language to design, simulate and deploy robotic applications, in: International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2012, pp. 149–160.

[P54] Thomas Buchmann, Johannes Baumgartl, Dominik Henrich, Bernhard Westfechtel, Robots and their variability–A societal challenge and a potential solution, in: Product Line Approaches in Software Engineering (PLEASE): 2015 IEEE/ACM 5th International Workshop on, IEEE, 2015, pp. 27–30.

[P55] Mohamed Walid Ben Ghezala, Amel Bouzeghoub, Christophe Leroux, RSAW: A situation awareness system for autonomous robots, in: 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV), IEEE, 2014, pp. 450–455.

[P56] Arunkumar Ramaswamy, Bruno Monsuez, Adriana Tapus, Saferobots: A model-driven approach for designing robotic software architectures, in: 2014 International Conference on Collaboration Technologies and Systems (CTS), IEEE, 2014, pp. 131–134.

[P57] Trevor Elliott, Mohammed Alshiekh, Laura R Humphrey, Lee Pike, Ufuk Topcu, Salty-A domain specific language for GR (1) specifications and designs, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 4545–4551.

[P58] Davide Brugali, Nico Hochgeschwender, Software product line engineering for robotic perception systems, Int. J. Semant. Comput. 12 (01) (2018) 89–107.

[P59] Jan O. Ringert, Bernhard Rumpe, Andreas Wortmann, Tailoring the montiarcautomaton component & connector ADL for generative development, in: Proceedings of the 2015 Joint MORSE/VAO Workshop on Model-Driven Robot Software Engineering and View-Based Software-Engineering, ACM, 2015, pp. 41–47.

[P60] Johannes Baumgartl, Thomas Buchmann, Dominik Henrich, Bernhard Westfechtel, Towards easy robot programming-using DSLs, code generators and software product lines, in: ICSOFT, 2013, pp. 548–554.

[P61] Hans Christian Woithe, Ulrich Kremer, TrilobiteG: A programming architecture for autonomous underwater vehicles, in: Proceedings of the 16th ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems 2015 CD-ROM, 2015, pp. 1–10.

[P62] Diego Alonso, Cristina Vicente-Chicote, Olivier Barais, V3Studio: A component-based architecture modeling language, in: Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on the, IEEE, 2008, pp. 346–355.

[P63] Christian Heinzemann, Ralph Lange, vTSL-A formally verifiable DSL for specifying robot tasks, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 8308–8314.