Journal of Software Engineering for Robotics
7(1), July 2016, 75-99
ISSN: 2035-3928

A Survey on Domain-Specific Modeling and Languages in Robotics

Arne NORDMANN[1] Nico HOCHGESCHWENDER[2] Dennis WIGAND[1] Sebastian WREDE[1]

[1] Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University, Bielefeld, Germany
[2] Department of Computer Science, Bonn-Rhein-Sieg University, Sankt Augustin, Germany

Abstract—The development of advanced robotic systems is challenging as expertise from multiple domains needs to be integrated conceptually and technically. Model-driven engineering promises an efficient and flexible approach for developing robotics applications that copes with this challenge. Domain-specific modeling allows to describe robotics concerns with concepts and notations closer to the respective problem domain. This raises the level of abstraction and results in models that are easier to understand and validate. Furthermore, model-driven engineering allows to increase the level of automation, e.g., through code generation, and to bridge the gap between modeling and implementation. The anticipated results are improved efficiency and quality of the robotics systems engineering process. Within this contribution, we survey the available literature on domain-specific modeling and languages that target core robotics concerns. In total 137 publications were identified that comply with a set of defined criteria, which we consider essential for contributions in this field. With the presented survey, we provide an overview on the state-of-the-art of domain-specific modeling approaches in robotics. The surveyed publications are investigated from the perspective of users and developers of model-based approaches in robotics along a set of quantitative and qualitative research questions. The presented quantitative analysis clearly indicates the rising popularity of applying domain-specific modeling approaches to robotics in the academic community. Beyond this statistical analysis, we map the selected publications to a defined set of robotics subdomains and introduce an extended classification scheme to allow a fine-grained mapping of publications addressing the architecture and programming of robotics systems. We map the surveyed publications to typical development phases in robotic systems engineering. The resulting classification tree shall serve as overview and reference for potential users. Furthermore, we analyze the surveyed contributions from a language engineering viewpoint and discuss aspects such as the methods and tools used for their implementation as well as their documentation status, platform integration, typical use cases and the evaluation strategies used for validation of the proposed approaches. Finally, we conclude with recommendations for discussion in the model-driven engineering and robotics community based on the insights gained in this survey.

Index Terms—Model-Driven Engineering, Domain-Specific Modeling Languages, Code Generation, Language Engineering

1 INTRODUCTION

Model-driven engineering (MDE) and domain specific development methods are recognized to cope with the challenges of building complex heterogeneous systems in domains such as aerospace, telecommunication and automotive [1] which face similarly complex integration and modeling challenges as advanced robotics. A model can be defined as *"an abstraction of a system often used to replace the system under study"* [2] and often represents a partial and simplified view of a system or specific aspect. As such, the creation of multiple models is *"usually necessary to better represent and understand the system under study"* [2], which is particularly valid in the robotics domain due to its intrinsic interdisciplinary foundation.

Domain-specific modeling allows to describe robotics concerns with concepts and notations closer to the respective problem domain. This raises the level of abstraction and results in models that are easier to understand and validate, lowering the technical skills necessary to handle the complexity of robotics systems development. Furthermore, it focuses on increasing the level of automation, e.g., through code generation or direct model interpretation, to bridge the gap between the modeling and the implementation levels and to improve the efficiency and quality of the robotics systems engineering process.

Regular paper – *Manuscript received August 15, 2015.*

- *The research leading to these results received funding from the European Community's Horizon 2020 robotics program ICT-23-2014 under grant agreement 644727 - CogIMon and was supported by a grant of the Cluster of Excellence Cognitive Interaction Technology (CITEC) at Bielefeld University. Nico Hochgeschwender received a PhD scholarship from the Graduate Institute of the Bonn-Rhein-Sieg University which he gratefully acknowledges.*

In the last years, this approach was actively adapted to the robotics domain to handle the complexity of robotics systems development.

The purpose of this survey is to report on the state of the art in Domain-specific (Modeling) Languages (DS(M)L) in robotics, and provide an overview of subdomains relevant for programming and simulation of robotics applications that are already supported through the MDE approach. Similar surveys, yet for a wider scope, have been conducted by Biggs and MacDonald [3] as well as Van Deursen *et al.* [1]. A mapping study in the field of robotics has recently been done in the master thesis of Cattivera and Casalaro [4], yet with a narrower scope than this survey by focusing on mobile robot systems. This contribution extends on a previously published earlier version of this survey [5] with improved coverage of the available literature as well as a more in-depth discussion and classification of the surveyed publications, i.e. through the mapping of the identified publications to the phases of a typical robotics development process.

The intended addressee of this survey are potential DS(M)L users as well as system integrators who are interested in applying a domain-specific modeling approach directly for one of the anticipated MDE use cases. Furthermore, we address language developers that are interested in reuse and extension of existing approaches and who want to learn about best practices in order to foster scientific exchange and community building inside the domain. Hence, the central aim of this survey is to provide an overview on the state-of-the-art in domain-specific modeling and languages in robotics.

The remainder of this article continues with a brief outline of the quantitative and qualitative research questions that we want to investigate in this contribution, followed by an introduction to core concepts of domain-specific modeling in Section 3 and a definition of a minimal set of methodological requirements on DS(M)L approaches to be included in the survey. Subsequently, Section 4 analyses the target domain along two dimensions, which are the core robotics domains and a reference development process. Section 5 explains how the literature survey was conducted along a defined protocol, while Section 6 classifies and quantitatively assesses the covered literature, also providing an overview of several non-functional aspects along the quantitative research questions.

On that basis, Section 7 discusses the qualitative research questions along key publications and identifies best practices that are relevant both for DS(M)L users and developers. We analyze the surveyed contributions from a language engineering viewpoint and discuss aspects such as the methods and tools used for their implementation as well as their documentation status, platform integration, typical use cases and the evaluation strategies used for validation of the proposed approaches. In Section 8 we briefly discuss potential threats to the validity of the presented study. Section 9 summarizes the main findings of the survey and discusses requirements on accessibility and documentation standards for DS(M)L publications to allow more effective reuse of knowledge provided with domain-specific models in the area of robotic systems engineering. Finally, Section 10 groups the surveyed publications according to subdomain and development phase as a quick reference to the bibliography entry for the interested reader. Links at the end of each bibliography entry link back to the sections where the respective publication is mentioned.

2 Objectives

The main objective of this survey is to investigate the question: **What is the state-of-the-art of Model-Driven Engineering and Domain-Specific (Modeling) Languages in Robotics?** This main objective will be investigated along the following quantitative and qualitative research questions.

2.1 Quantitative Research Questions

The research questions addressed within this survey are an extension of our earlier analysis [5]. By answering these questions we aim to provide insights into quantitative relationships that are relevant for answering the main question introduced above. The questions can be summarized as follows:

RQ1 *Which functional aspects are typically addressed with DS(M)Ls in robotics?*
This question investigates the distribution of MDE and DS(M)L approaches over the different functional domains prevalent in robotics systems to find out which of those are particularly well supported. Our initial hypothesis is that a larger number of model-based approaches exists for well understood and mature aspects such as kinematics modeling or motion control in contrast to comparably recent research fields such as force control.

RQ2 *Which robot application development process phases are well covered by DS(M)L-based MDE approaches?*
This question investigates the distribution of MDE and DS(M)L approaches over the different phases of a robotics system engineering process. A preliminary observation is that MDE and DS(M)L approaches typically support modeling of certain capabilities and systems, but ignore runtime aspects.

RQ3 *Which tools are used to realize DS(M)Ls and apply MDE in the context of robotics?*
To investigate homogeneity and compatibility inside the domain, we analyzed which tools are used for the implementation of MDE and DS(M)L methods. Findings shall provide insights on the level of fragmentation as well as identify possible standard environments that may lead to easier (meta)model interchange and reuse of language implementations.

RQ4 *What are the publication trends?*
On a meta level we are also interested in the momentum of the MDE and DS(M)L topic in the robotics community, which we investigate along two more fine-grained questions: i) *What is the publication rate by year?* Is there

a positive trend for MDE approaches and DS(M)Ls in robotics? ii) *What are the main venues MDE and DS(M)L topics are published to?* What are the main conferences, journals or workshops where work on MDE and DS(M)L approaches is published? How is the distribution between domain (robotics) venues and those regarding the method (software and modeling venues) as well as among the publication types, i.e., workshop, conference and journal paper?

We provide answers to these questions by utilizing the statistical data gathered through the survey and by highlighting selected examples in Section 6.

2.2 Qualitative Research Questions

To further investigate the current state of the art of MDE and DS(M)Ls in robotics, we try to answer questions regarding the typical development, usage as well as the accessibility and documentation state of these approaches.

RQ5 *What is the accessibility and documentation state of DS(M)Ls and their MDE ecosystem?*
An important factor for reuse of models and DS(M)Ls, scientific exchange and community building around MDE in robotics is accessibility and documentation. This comprises several factors like technical accessibility, e.g., download of the language or models, licensing, and documentation.

RQ6 *What typical artifacts are generated by DS(M)L-based MDE approaches and how are they used?*
To assess the intended use of the MDE approaches and DS(M)Ls, we looked at the artifacts generated with their help (if any) and the context they are used in and also how this differs between the subdomains and disciplines (cf. RQ1).

RQ7 *How much platform-dependency is introduced?*
An important aspect of the generated artifacts and the model transformations is how tightly they are coupled to a certain platform. "Platform" in this context means the technical execution context, so the software framework and all additional tools or libraries necessary to use the DS(M)L or the generated artifacts.

RQ8 *How are DS(M)L approaches evaluated?*
Evaluation of a DS(M)L-based approach in its intended use-case is not only interesting from a developer's perspective, but also to validate an approach from a scientific perspective. Evaluation can also be used to demonstrate or prove whether the approach is *complete*, which implies that typical examples of the domain can be completely expressed with the modeling approach.

RQ9 *What are the development processes that lead to the surveyed MDE and DS(M)L approaches?*
The identification and formalization of domain-specific concerns in an abstract model or the automation of manual development tasks are typical motivations for the development of models and DS(M)Ls [6]. We wanted to know whether the surveyed publications report on why a DS(M)L-approach has been applied and what activities such as domain and problem assessment or expert consultation are used for domain analysis and language development.

RQ10 *How are models, metamodels and DS(M)Ls (re)used by third parties?*
While many publications introduce new MDE methods and DS(M)Ls, we are also interested whether such approaches are actually *used* by third-parties (not the developers themselves) and in how far the reported usage scenarios demonstrate the intended benefits of model-driven engineering. With this question we try to gain some insight whether model (re)usage within the domain is already happening and how the MDE approach is practically applied in a robotics context. While this question may deserve a dedicated survey, we wanted to share the preliminary observations gathered during the analysis of candidate publications as we consider these an interesting contribution.

These questions will be discussed along reference publications in Section 7. The discussion shall provide insights on best practices that are useful for language developers when publishing their DS(M)L approaches in the robotics or software engineering community. It is furthermore intended to provide hints to potential users *and* developers on which criteria are relevant when considering to use or extend a DS(M)L-based approach.

3 Domain-Specific Modeling Languages

In order to perform a systematic review on domain-specific modeling for robotics system engineering, a necessary prerequisite is to define what we consider a domain-specific (modeling) language and to briefly describe terminology, concepts as well as use cases that are relevant from the survey's point of view. While a full introduction to the topic of MDE with DS(M)Ls (cf. [2] for a recent conceptual overview) is beyond the scope of this article, the following paragraphs motivate our expectations on the surveyed publications from a software engineering perspective. These expectations partially define the inclusion and exclusion criteria of our search process (cf. Section 5) and link the presented DS(M)L concepts to the previously introduced research questions.

While some aspects such as the the basic properties of modeling languages and the different mechanisms to utilize models on a target platform are essential for language users (cf. Fig. 1), other concepts such as the formalism used at the metamodel level are mainly relevant for language developers and researchers in the DS(M)L community. Hence, the following subsections are structured according to these different perspectives.

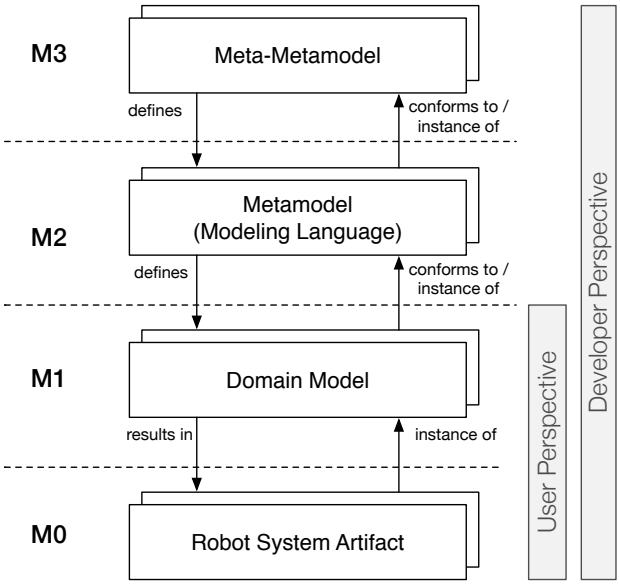

Fig. 1. Abstraction levels of model-driven engineering with DS(M)Ls informed by the standard definition of the OMG [7]. In robotics, multiple metamodels and modeling languages, models and artifacts with potentially complex interactions will be required to fully describe an executable system.

3.1 User Perspective

According to van Deursen *et al.* [1], a DSL is defined as a "*programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain*". The abstractions and notations must be "*natural/suitable for the stakeholders who specify that particular concern*" [8].

These definitions already highlight two fundamental characteristics of well-designed DSLs: their expressive power targeted at a specific domain and the definition of formal notations intuitively understandable for domain experts while being machine processable, eventually yielding executable models of robotics applications.

In contrast to *General-purpose Programming Languages* (GPL) such as C++, Java, or Python, DSLs usually contain only a restricted set of notations and abstractions specialized to one or more particular domain(s). Compared to *external* DSLs that define their own syntax and semantics, so-called *internal* DSLs are embedded in extensible general purpose languages such as Lua, Racket or Ruby. They extend the syntax and potentially the semantics of the host language with domain-specific notations and abstractions. This adds the expressive power of the DSL to the GPL.

Similarly, *Domain-Specific Modeling Languages* (DSMLs) that rely on graphical notations must be differentiated from general purpose modeling languages (GPMLs) such as UML [9] or SysML [10]. GPMLs typically provide a larger number of generic constructs and notation, which allows their application in different domains, e.g., the modeling of object-oriented software systems in UML. In contrast, DSMLs are typically comprised of a smaller set of concepts and graphical notations that are close to the respective application domain [2]. A common practice for the definition of DSMLs is the use of the UML Profile mechanism that allows to add domain-specific abstractions to UML, e.g., MARTE [11] for modeling and analyzing real-time systems. MARTE is an example of an internal DSML, which is realized as extension to an existing modeling language (UML). An external DSML typically provides a custom graphical syntax, which conforms to a custom meta-model and requires a customized framework for graphical editing.

This differentiation already allows to define the scope of the presented survey in terms of modeling languages. We consider both textual and graphical DS(M)Ls as relevant and discuss their usage in the context of robotics along **RQ3**. In contrast, we do not include publications on general purpose (modeling) languages or their use in the context of robotics system development. Hence, this scope is reflected in the search terms of our overall search process (cf. Section 5).

From a user's perspective, the execution or interpretation of robot-specific models is an important *use case* for domain-specific modeling, e.g., the embedding of a generated model-based kinematics and dynamics controller in a larger robotic system. While internal DS(M)Ls rely on (and are bound to) the execution semantics of their host language, external DS(M)Ls can be transformed to a format that directly allows execution on a target platform or interpretation, e.g., through a virtual machine. Another use case for domain-specific models in a robotics context is system analysis. Here, model checking and validation, the setup and analysis of simulations or model-based testing are typical tasks that can be addressed. Beyond execution and analysis, models are often directly suitable as documentation but can also be used to generate further visualization or documentation assets such as specific views on a system. For the survey, we did not focus on a particular use case for DS(M)Ls. Instead, we highlight in the discussion of **RQ10** representative usage scenarios.

Further concerns that are highly relevant for potential DS(M)Ls users are the kind of *artifacts* such as source code, configuration files, etc. that are provided through a model-based development approach and how these artifacts are used within a *target platform*. Here, target platform refers to the specific hard- and software required to integrate and run the artifacts within a robotics system architecture. Nowadays, many approaches, e.g., Frigerio *et al.* [12], already demonstrate the parallel generation of different kinds of artifacts such as C code for real-time components as well as Matlab/Simulink code for simulation and analysis through exchangeable code generators. As part of the discussion on **RQ6**, the survey shall provide insights on the M0 artifacts (cf. Fig. 1) that

are current targets of model-driven engineering in robotics. In order to support variability with regard to the target platforms, flexible model transformation and code generation techniques allow to interface generated or non-generated artifacts, e.g., with different robot devices, software libraries or robotics middleware such as ROS or YARP. To support this mapping to different environments, models often need to be augmented with additional information that needs to be added by developers. As technical and functional variability is a major challenge in robotics system engineering, we discuss in the context of **RQ9** in how far these benefits of model-based development are already demonstrated in the reported approaches.

3.2 Developer Perspective

Language Engineering [8] extracts agreed-upon abstractions, syntax and semantics from the problem domain, e.g., by reviewing existing code examples and APIs, through the analysis of formal descriptions found in the literature or the application of further analysis patterns [6]. Based on the results of these domain analysis steps, the identified abstractions and desired notations can be realized in a DS(M)L.

In order to efficiently implement and apply a DS(M)L approach for the development of robotics systems and to fully exploit its benefits, DS(M)Ls are typically realized in *tools* tailored to model-driven development such as the Eclipse Modeling Project [13] or JetBrains MPS [14]. These so-called *language workbenches* offer extensive support for the development of DSLs. Domain-specific modeling languages are themselves often modeled using the elements and following the rules of *metamodel* languages [2] such as MOF [7] in case of UML or Ecore as part of the Eclipse Modeling Project. The alternative to the use of a potentially complex metamodel language available in a language workbench is the use of a grammar specification formalism such as (E)BNF, which can be used by parser generators. However, language workbenches provide further benefits beyond the definition of abstract and concrete syntax such as support for the development of textual and/or graphical editors with rich code completion and dynamic constraint checking at design time that improve the usability for language users. Furthermore, these environments provide extension points to plug-in required model-to-model (M2M) and model-to-text (M2T) transformations in order to generate a different textual representation from system models that integrates with the overall environment used for the development of a robotics application. Language engineering tools and formalism used in current publications on DS(M)Ls in robotics are discussed in the context of **RQ3**.

The above mentioned aspects contain fundamental facets that need to be addressed in scientific contributions presenting domain-specific modeling approaches from a language developer perspective. Hence, the DS(M)L approaches considered in this survey i) should either provide a language definition or metamodel, based on, e.g., Ecore or (E)BNF, ii) or must provide an example of their concrete syntax (notation), iii) must be textual (internal or external) or graphical languages, iv) and explain how a mapping to a target platform is achieved. We consider all of these aspects as relevant for DS(M)L developers and users, since formally described models and the identification of popular language engineering tools might lead to easier model interchange for developers, whereas users may actually want to learn about well supported modeling languages.

While many of the above mentioned criteria are formulated from a software engineering perspective, the most important criterion to judge whether a paper or article is included in the survey is whether or not it targets a relevant concern in the robotics domain. These relevant concerns are introduced in the following section that identifies a set of sub-domains considered particularly important and mature in the context of robotics systems as well as development phases that are prevalent in the engineering process of robotics applications.

4 Domain Analysis

A dedicated goal of this survey on the state of the art of domain-specific modeling and languages in robotics is to give potential DS(M)L users and developers orientation guide with regards to the domain concerns that are addressed by existing approaches. Naturally, one analysis dimension is defined by the functional aspects that are covered by a domain-specific modeling approach, i.e. which kinds of robot system aspect such as a motion control algorithm can be modeled using a surveyed approach. Since DS(M)L and MDE approaches are particularly designed to facilitate and enhance the engineering process, a second dimension is introduced that addresses the question to which phases of a typical robotics system development process the surveyed approaches contribute.

The classification schemes that we utilize within each dimension are introduced in the following.

4.1 Functional Dimension

The definition of the functional analysis dimension essentially raises the question about the mature sub-fields of engineering and research that can be identified in robotics. Finding a clear answer to this questions proves surprisingly hard. Partially, because existing publications and standard ontologies are too specific, existing taxonomies too broad, or just due to the fact that in many robotics textbooks the partitioning of the robotics field into sub-areas is still differing. Hence, we decided for the purpose of an initial classification into subdomains to utilize Part A of the Springer Handbook of Robotics [15] as a “normative” and neutral reference.

Here, Part A (*Robotics Foundations*) covers the fundamental principles and methods needed to create a robotic system. While developing such a system, various challenges have to be tackled in kinematics, dynamics, actuation, sensing, motion planning, control, programming and task planning. Thus, the

subdomains[1] chosen in this survey will correspond to these categories:

`Kinematics` refers to the motion of bodies in robotic mechanisms without taking the forces/torques causing the motion into account. Hence, it includes general representations of the position and orientation of a body, the relation among the joints as well as conventions for representing the geometry of rigid bodies connected by joints.

`Dynamics` covers the relationships between actuation and contact forces that act on robot mechanisms. Such a mechanism in this sense is described by rigid bodies connected by joints. Furthermore, it pertains to the acceleration and motion trajectories resulting from these relationships.

`Mechanisms and Actuation` focuses on the mechanical structure of a robot that creates its movable skeleton. All elements that cause a robotic mechanism to move – so called actuators – are addressed along with the mathematical model that is used to characterize the robot's performance.

`Sensing and Estimation` ranges from robot-state estimation for feedback control to task-oriented interpretation of sensor data of any kind. Apart from estimation techniques, this category also covers different kinds of information representations.

`Motion Planning` covers collision-free trajectory planning for mobile platforms as well as robot actuators.

`Motion Control` addresses the dynamical model of robotic manipulators. This includes different controller approaches, such as independent-joint, PID as well as torque control.

`Force Control` pertains to the achievement of a robust and dynamic behavior of robotic systems in compliant interaction between robot and environment. Similar to the Motion Control category, it includes different control aspects, e.g., stiffness and impedance control.

`Architectures and Programming` refers to the way a robotic system is designed on the software-level. It can be divided into architectural structure and architectural style. The structure is represented by how the system is split up into subsystems and how they interact with each other. The style however addresses the underlying computational concepts.

`Reasoning Methods` focuses on symbol-based reasoning and knowledge representation. It covers logic- as well as probability-based approaches. Furthermore, this category also addresses learning, such as inductive logic learning, neuronal networks and reinforcement learning.

Of course, many further ways of decomposition of the robotics field exist, which may all be valid. For instance, "grasping and manipulation" could very well be thought of as a subdomain in its own right. However, we consider the categories introduced above as principal components for the core robotics problems. Following this idea, a paper on grasping and manipulation will (of course this depends on the specific contribution) very likely be classified into the motion control, motion planning and probably force control subdomains.

[1]Subdomains will be marked in the **`typewriter`**-style

4.2 Development Process Dimension

One general goal of DS(M)Ls is to support and structure developer's work in development phases. Hence, we labeled the surveyed DS(M)Ls according to their intended and potential usage within a development process. To ground this analysis we utilized the Robot Application Development Process in BRICS (BRICS RAP or shortly RAP) [16]. The BRICS RAP has been developed in the EU-funded project BRICS [17] and is a holistic process model for developing robot applications both in academia and industrial settings. The process model combines ideas from traditional software engineering [18], [19], agile software development [20], model-based engineering [21], [22], and system engineering [23] and foresees in its latest revision eight different phases, each of which requires several steps to complete the task. Note, the BRICS RAP also foresees feedback and interaction among development phases, but we present for the sake of simplicity only the core phases. We decided to use the BRICS RAP in this survey for two reasons. Firstly, to the best of our knowledge the BRICS RAP is one of the very few reported process models targeting robotic applications and is therefore applicable for our survey. Secondly, the BRICS RAP aims to cover the complete life cycle of a robotic application which enables to investigate whether DS(M)Ls are used to a particular extent in certain process phases. In the following we provide a brief overview about the process phases proposed by the BRICS RAP.

- In the **scenario building** phase, environment features, constraints and characteristics are defined. Furthermore, the robot's task is defined. This includes the specification of customer acceptance tests to be performed in the specified and potentially generalized environment.
- In the **functional design** phase, hardware requirements and top-level functionalities are derived based on the scenario definition. Furthermore, top-level functionalities are decomposed and dependencies among them are identified. Also an initial functional design stating which functionalities interact with each other is developed.
- In the **platform building** phase, the robot hardware is determined. This includes the selection and potential configuration of robot's sensors and actuators meeting the requirements defined in the functional design phase.
- In the **capability building** phase, basic and composite components are constructed up to the application-level and constraints for their deployment are specified. This also includes the specification and eventually generation of additional knowledge required for component execution such as knowledge bases and training data.
- In the **system deployment** phase, top-level component(s) are packaged into a complete application system which

defines a mapping of components and composites to computational units. Furthermore, features and procedures for system launch management are developed.

- In the **system benchmarking** phase, certain test procedures targeting different quality attributes are performed such as stress testing, safety and security testing, reliability and durability testing, and performance testing.
- In the **product deployment** phase, an application is tailored to a specific robot system. This includes also the installation of maintenance instrumentation and a final target platform system testing.
- In the **product maintenance** phase, the robot application is operated and maintained. This includes eventually the analysis of log files and the tuning of system parameters.

5 Process

The selection of the publications for this survey focused on publications that developed DS(M)Ls or metamodels to conceptualize aspects of the robotics domain or support certain research or engineering aspects. Compared to our previous survey [5] we extended the process to find potential candidates for this survey. A list of potential candidates was automatically generated by a script[2] performing a keywords-based query on the widely known publication database *Google Scholar*. *Google Scholar*[3] indexes the publication databases of all major scientific publishers and allows keyword-based full-text searches while restricting certain metadata, e.g., publication year or conference. All publications resulting from the selection process (cf. Fig. 2) were then analyzed manually regarding our research questions detailed in Section 2. Furthermore, the initial analysis has been reviewed by following a four-eyes principle where one author assessed the analysis of another author.

5.1 Selection Process

The script queried the publication database for publications conforming to one of two inclusion criteria (IC):

IC1 Publication was published in the proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE International Conference on Robotics and Automation (ICRA), International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), Robotics: Science and Systems Conference (RSS), Journal of Software Engineering in Robotics (JOSER), IEEE Transactions on Robotics (TRO), Springer Autonomous Robots (AURO), Elsevier Robots and Autonomous Systems (RAS), Wiley Journal of Field Robotics (JFR)[4], Workshop on Software Development and Integration in Robotics (SDIR) or the Workshop on Domain-Specific Languages in Robotics (DSLRob) **and** full-text search matching one of the keywords *"domain-specific language(s)"*, *"domain-specific modeling language(s)"*, *"generative programming"*, *"specification language(s)"*, *"description language(s)"*, *"code generation"*, *"dsl(s)"*, *"metamodeling"*, *"metamodel(s)"*, *"metamodeling"*, *"metamodel(s)"*, *"MDE"*, *"MDSD"*.

IC2 Publication was published in the proceedings of the Code Generation Conference (CG), International Conference on Generative Programming: Concepts & Experiences (GPCE), ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS), Conference on Model-driven Engineering and Software Development (MODELSWARD) or the IEEE/ACM International Conference on Software Engineering (ICSE), **and** full-text search matching one of the keywords *"robot(s)"* and *"robotic(s)"*.

Scanning the 12 robotics conferences, workshops and journals[5] for the 20 keywords regarding model-driven or domain-

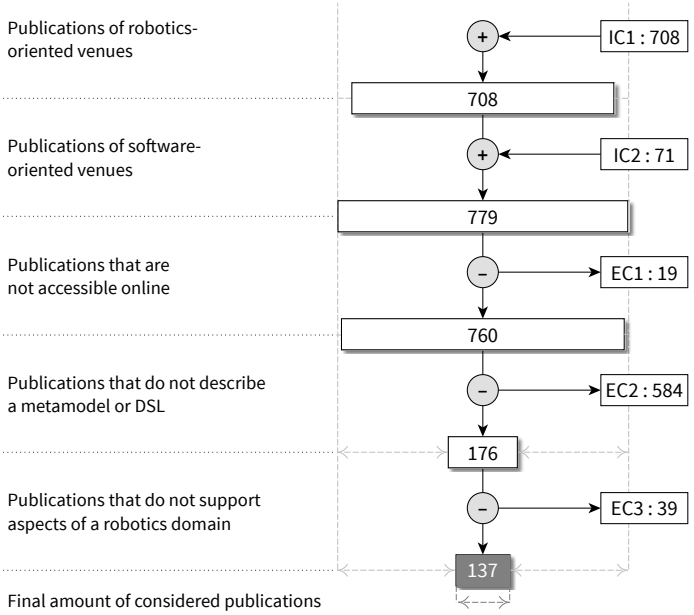

Fig. 2. Sequence diagram of the selection process. From top to bottom, an **IC** and **EC** is applied in each step respectively. While the boxes on the right represent the amount of contributions included or excluded by a specific criterion, the boxes in the middle show the actual amount after each step. Ultimately, 137 publications are considered in this survey.

[2] https://github.com/corlab/dslzoo/tree/query
[3] http://scholar.google.com/
[4] From 1984 until 2006 the journal was named *Journal of Robotic Systems*.
[5] The Workshop on Domain-Specific Languages in Robotics (DSLRob) is not available in official proceedings and can therefore not be queried via *Google Scholar*. We nevertheless included all 23 DSLRob publications of the years 2010 – 2014 manually due to its relevance to the topic.

specific methodology[6] (**IC1**) with a total of 220 single queries resulted in 708 publication candidates.

Scanning the 5 software engineering conferences, workshops and journals for the 4 keywords regarding robotics[6] (**IC2**) with a total of 20 single queries resulted in 71 additional publications, totaling in 779 publication candidates for this survey.

After this automated process, the 779 publication candidates were filtered manually by the authors of this survey by three exclusion criteria (EC):

EC1 Publication is not online accessible in article format. Presentation slides are not sufficient and books are also excluded.

EC2 Publication does not describe a metamodel or domain-specific language, or publication is not complying with our definition from Section 3, e.g., domain-specific language or metamodel not documented via grammar or example.

EC3 DSL does not model or support aspects of the introduced domain.

After applying EC1, 760 publication candidates remained. This step sorted out false positives from the Google Scholar query and publications that were only available as slides but not in proceedings.[5] 176 publication candidates remained after applying EC2, mainly filtering out publications that used our keywords somewhere in the publication, e.g., in the related work part, discussion, or bibliography, but do not introduce any domain-specific language or metamodel. Also publications that only vaguely describe a domain-specific language or metamodel, but do no provide any grammar, formal specification or even example are excluded with this criterion. Finally, 137 publications remained after applying EC3, mainly filtering out publications added through IC2 that used the term "robot", e.g., somewhere in its outlook without actually supporting a robotics use-case.

5.2 Analysis Process

The 137 publications that resulted from the selection process were manually screened by the four authors. In this process, additional metadata was attached that helps answering the research questions detailed in Section 2.

Some of the quantitative aspects introduced in Section 2.1, e.g., year and venue, were already annotated during the automated selection process. Further aspects like subdomains, development phases, and the formalization had to be annotated manually.

While screening the publications, the four authors of this survey also annotated if the paper was especially relevant for one of the qualitative research questions discussed in Section 2.2, e.g., being an especially good or relevant example, following best practice or being a special case in this aspect. To avoid subjective bias during the annotations of the papers, all papers were evaluated by at least two of the four authors of this survey.

[6]Due to technical restrictions, singular and plural forms of the keywords had to be queried separately, e.g., "description language" and "description languages", resulting in 20 keywords for IC1 and 4 keywords for IC2.

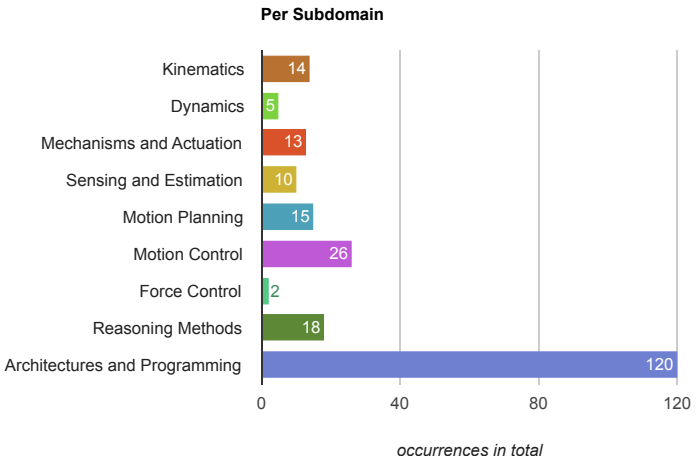

Fig. 3. Distribution of the surveyed publications over robotics subdomains as defined in [15].

6 Analysis

In the following, the surveyed publications are analyzed according to the research questions introduced in Section 2.1. The identified quantitative relationships provide a compact overview on the functional or engineering concerns that are addressed with DS(M)Ls in robotics and indicate what modeling tools are used to realize these approaches. Furthermore, we present metadata about the surveyed publications such as the temporal distribution over the last decades, which clearly indicates a rising number of DS(M)L publications. While the focus in this section generally is on quantitative analysis, we highlight selected papers if we consider these as representative for a certain kind of category or relationship.

6.1 Subdomains (RQ1)

As introduced in Section 4, the functional subdomains that serve as a basic classification ontology are motivated by the core foundations of robotics as outlined in *Springer's Handbook of Robotics* [15]. The classification result, cf. Section 10, shall serve as an annotated bibliography and reference guide for potential DS(M)L users and developers to foster discussion, reuse and extension of languages or models.

Besides this mapping of the individual publications, our initial research question is to assess the overall distribution of DS(M)L approaches with regard to the defined subdomains. Fig. 3 provides an overview for the observed relationship between robotics subdomains and DS(M)L publications as a first result with regard to this question. The underlying numbers seem to support the initial hypothesis that well

understood subdomains are covered by a larger number of contributions, whereas recent research fields are sparely or not covered at all. Domains such as **Motion Planning** and **Motion Control** (that we consider well understood), are addressed by more than 11 % of the publications. In contrast to that, so far only two contributions [24], [25] are dealing with **Force Control**. One of them [24] provides an internal DSL to specify force-velocity controlled motions following the Task-Frame formalism introduced by Mason [26]. The article is also a good example how existing robotics knowledge is reused in the form of a DS(M)L as the article is from 2011 whereas the underlying theory dates back to the 80s and 90s. We also consider **Kinematics** a mature subdomain and observe a higher number of reported DS(M)L-approaches than in the field of **Dynamics**. Rather recently, there are also contributions combining these subdomains. For instance, both **Kinematics** and **Dynamics** are covered by [27], [12], [28], [29] where for example both aspects are required to compute algebraic quantities for the sake of various control applications.

Revisiting Fig. 3, the subdomain **Architectures and Programming** attracts attention. It is addressed by over 53 % of the publications in this survey. We believe this is mainly due to the following reasons. First, while the Handbook chapter implicitly defines this subdomain to consider structural aspects of robotics architectures such as basic component models [32] and composition of components [33], [34] which is reasonable, it also includes rather computational aspects such as the *coordination* of architectural elements on different levels of abstraction, e.g., on a task [35], [36] and behavior-level [37]. As we need to apply this definition to all of the surveyed papers, a comparably large fraction of the papers, i.e. [38], [39], [40], [41], [37], are mapped to this category. A second reason that we consider relevant here is that researchers applying DS(M)L methods are likely to work on topics that are closely related to this subdomain.

The large number of publications within this category called for a deeper analysis of this subdomain. Following a similar approach as with the Handbook of Robotics, we consulted the Software Engineering Body of Knowledge [30] published by the IEEE Computer Society as a widely accepted reference to establish a more fine-grained classification taxonomy. In particular, we associate the **Programming** aspect with the key issues addressed the subsections of Section 2.2 within the SWEBOK on *Software Design* and the **Architecture** aspect with the issues outlined in the subsections of Section 2.3 on *Software Structure and Architecture*. TABLE 1 briefly summarizes each of these issues according to their original definition. We are aware that these terms are defined and introduced in the SWEBOK without a dedicated focus on robotics software development. For instance, we propose to explicitly extend the definition of `Security and Safety` to include models that address the physical safety of humans in the presence of robots. Nevertheless, the re-use of definitions from the field of Software Engineering to classify software development aspects in Robotics seems more natural and promising to us than to invent new but closely similar vocabulary.

Software Design [30, Sect. 2.2]	Sect.
`Concurrency`: Decomposition of software into processes, tasks, and threads, dealing with related issues of efficiency, atomicity, synchronization, and scheduling.	2.2.1
`Control and Handling of Events`: Organization of data and control flow as well as handling of reactive and temporal events.	2.2.2
`Data Persistence`: Handling of long-lived data.	2.2.3
`Distribution of Components`: Distribution of the software across the hardware, communication of components, and how can middleware be used to deal with heterogeneous software.	2.2.4
`Error and Exception Handling and Fault Tolerance`: Prevention, toleration, and processing of errors as well as dealing with exceptional conditions.	2.2.5
`Interaction and Presentation`: Structuring and organization of interactions with users as well as the presentation of information.	2.2.6
`Security and Safety`: Prevention of unauthorized access to and manipulation of information and other resources. Limiting of damage, continuation of service, speed-up of repair, and how to fail and recover securely. *Ensuring safety of humans in the presence of robots.*	2.2.7
Software Structure and Architecture [30, Sect. 2.3]	Sect.
`Architectural Structures and Viewpoints`: Description of architectural structures and software designs in general by independent and orthogonal views.	2.3.1
`Architectural Styles`: Descriptions and guidance for the high-level organization of software providing "a specialization of element and relation types, together with a set of constraints on how they can be used".	2.3.2
`Design Patterns`: Provide "a common solution to a common problem in a given context" [31]. Typically employed at a lower abstraction level than architectural styles.	2.3.3
`Architecture Design Decisions`: Impact of quality attributes and the trade-offs among competing quality attributes that provide the basis for design decisions.	2.3.4
`Families of Programs and Frameworks`: Software product lines or Frameworks encapsulating commonalities among elements and targeting re-use by designing customizable components that account for variability.	2.3.5

TABLE 1
Categories for decomposition of the architecture and programming subdomain. The referenced subsections point to the respective SWEBOK [30] subsections.

Hence, we conducted a deeper analysis of this subdomain by assigning each of the 120 papers to a maximum of three of the introduced categories. This annotation was done by at least two of the four authors of this survey for each paper. The outcome of this analysis is depicted in Fig. 4. First, the distribution highlights that much of the work belonging to this category is concerned with coordination aspects such

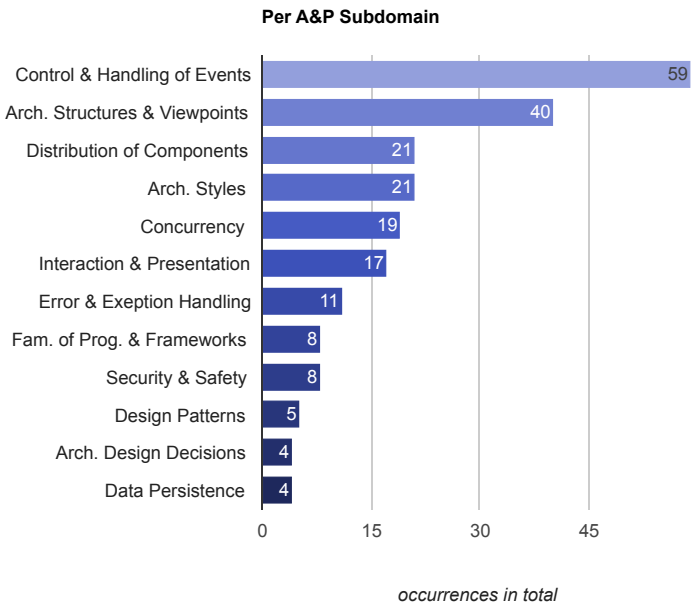

Fig. 4. Distribution of the surveyed publications within the **`Architectures and Programming`** subdomain.

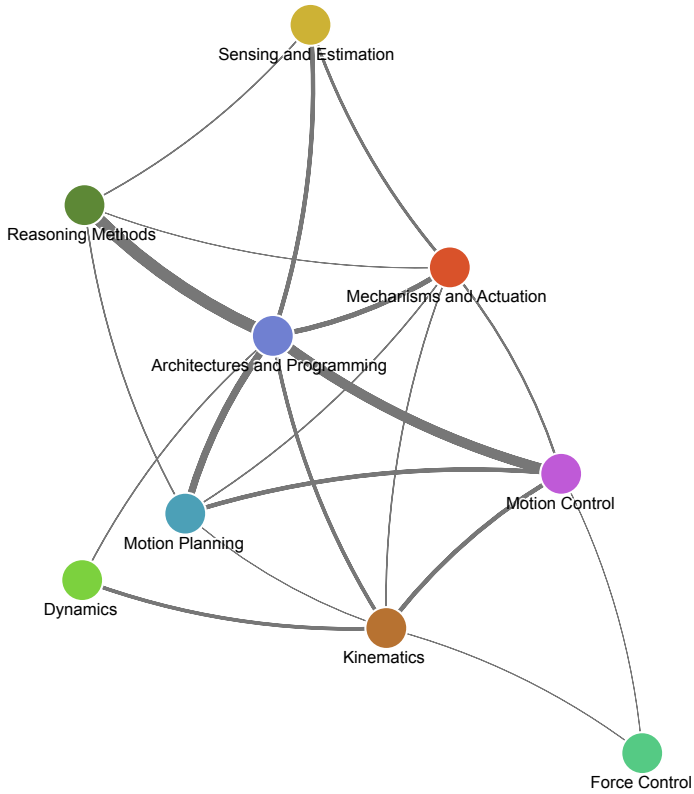

Fig. 5. The correlation between the different subdomains (colored nodes) is represented by the size of the edges. For instance, a thick edge means that a lot of publications are associated with the two subdomains.

as `Control and Handling of Events`, which is reasonable given that models for event handling are required in many applications to specify robot behavior and that we suppose that the required methods within this field are comparably well understood. Second, papers frequently consider *Architectural Structures and Viewpoints*, `Distribution of Components` and `Architectural Styles` as concerns that are targeted with domain-specific modeling languages. While principles of component-based software engineering are considered best practice in robotics software engineering, the modeling of component-based architecture and systems also represents a major topic in the general software engineering domain. Third, only a few papers report on `Security and Safety` aspects. Being a relevant concern for many advanced robotics applications and human-robot interaction, this observation is surprising. Summarizing, the resulting distribution shows that the SWEBOK-informed categories work well as a decomposition for the **`Architectures and Programming`** subdomain. Section 10.9.1 provides an overview of the respective publications within this subdomain and their individual mapping to the chosen categories.

While the SWEBOK has many more knowledge areas that are of course also relevant from the perspective of robotics systems engineering, we argue that most of these (i.e. requirements engineering) are already covered by the more domain-specific BRICS RAP that we use as orthogonal classification scheme. Please refer to the next Section 6.2 for this mapping of the surveyed contributions to development process phases.

However, the correlation among the different subdomains is as well interesting. It can be noticed in Fig. 5 that there are high correlations between **`Architectures and Programming`** and almost every other considered functional domain. This means a lot of publications that present an approach focused on e.g., **`Motion Control`** or **`Reasoning Methods`**, also considers architectural aspects. An explanation may be that these domain-specific models can typically only be validated on the real robot system if they are integrated into an overall architecture, e.g., providing realistic sensor data. The correlation between **`Motion Control`**, **`Motion Planning`** and **`Kinematics`** seen in Fig. 5, is reasonable due to the close affiliation of those domains [42], [43], [44], [45], [46], [47], [48]. Considering **`Reasoning Methods`**, it is mostly related to **`Architectures and Programming`** [49], [50], [51], [52] and **`Motion Planning`** [53], [54], whereas there is apparently no correlation to low-level **`Motion Control`**, **`Kinematics`**, **`Dynamics`** and **`Force Control`**. The strong correlation between **`Reasoning Methods`** and **`Architectures and Programming`** is meaningful as for instance in order to coordinate tasks one needs also to reason about robot capabilities and eventually varying environment and task conditions as addressed in the work of robot control architectures, e.g., [55], [56].

6.2 Development Phases (RQ2)

To answer **RQ2** we assessed the contributions from a DS(M)L user perspective, namely to which extent does a language support the development of a robotics software system within a particular development phase. In the following we summarize observations relevant for DS(M)L users and developers.

The majority of the surveyed publications, namely 128, address the **capability building** phase. This is not surprising as within this development phase not only basic components following a component model are constructed [57], but also composite components are developed [58] leading to higher-level and potentially reusable capabilities [34], [59], which are orchestrated or coordinated [60], [38], on a behavior-level [61], [41] as well as on a task-level [46], [62], [35], [63].

Much less contributions are assigned to the **platform building** phase, namely 26. However, the diversity of DS(M)Ls is impressive and ranges from means to model sensor characteristics [64], [65], [58], [52], computational hardware resources [66], kinematic and dynamic properties of manipulators [27], [67] and hands [68] to kinematic abstractions for arbitrary modular robots [69].

A similar range of publications, 35, addresses the **functional design** phase. Exemplary contributions belonging to this category present modeling approaches targeted at requirements represented in the form of crucial mission guarantees [70], constraints for machine configuration [71] or motion constraints [45] that are described in structured English.

Even though, several articles describe how they accomplish development tasks in the **deployment phase**, e.g., [72] generating ROS launch files, they do not necessarily introduce dedicated abstractions for modeling deployment activities and artifacts. In summary, 22 publications consider the system deployment phase. Exemplary contributions in this category such as [49], [66], [59] provide means to model deployment specifications, e.g., threading and platform properties, to facilitate use cases such as scheduling analysis. All these approaches (see also [73], [74]) strictly separate the deployment model from the architecture specification in order to enable deployment of the same architecture on different platforms.

Twelve articles consider DS(M)Ls to model run time aspects as required in the **system benchmarking**, **product deployment** and **product maintenance** phase. Most notably, in [75] and [76] design time models, e.g., software architecture models, are combined with adaptation models in order to express how a robot should adapt its system architecture based on varying environment, platform or task conditions.

Very few publications, namely five, have been assigned to the **scenario building** phase. Here, we have to admit that the differentiation between the scenario building phase and the functional design phase was challenging as both phases deal with high-level requirements. Nevertheless, some articles such as [77] and [78] can be clearly assigned to the scenario building phase as they deal with modeling of task and/or environment requirements.

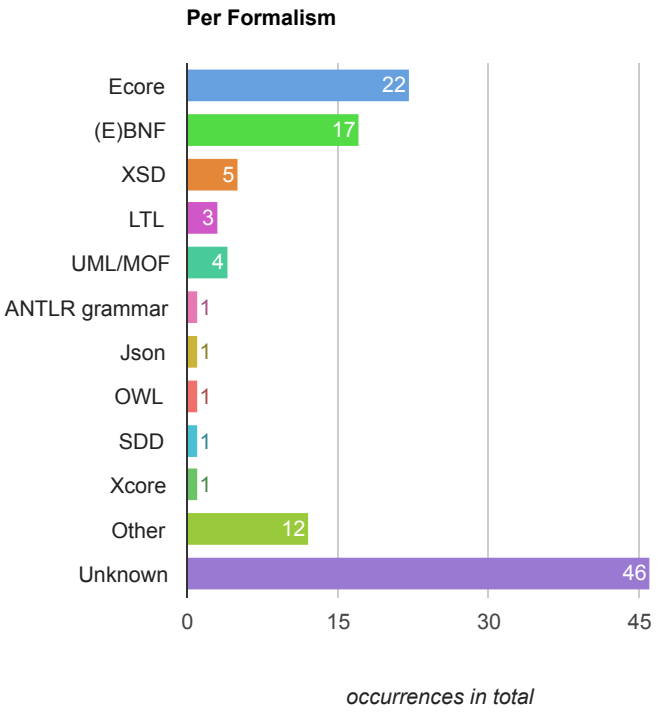

Fig. 6. Formalisms of the DS(M)Ls included in this survey.

6.3 Meta-Metamodels and Formalisms (RQ3)

This section analyzes the formalisms that were used for the development of the DS(M)Ls to shed some light on the homogeneity and compatibility inside the domain (**RQ3**). As far as this is assessable in the publications or the referenced documentation, we annotated the formalisms used for development of the metamodels and the external DS(M)Ls, as well as development of internal DS(M)Ls.

The majority of DS(M)Ls assessed with this survey is realized as an external DS(M)Ls. While these DS(M)Ls need to be based on a formalism, internal DS(M)Ls are bound to the specific syntax of their host language and are therefore not considered in Fig. 6. There are various kinds of host languages used by internal DSLs, such as F# [54], Lua [79], Prolog [80], and C++ [81] – to name a few.

In the following, the observations of this survey are presented based on Fig. 6. We gather UML Profiles [9] under the term **UML/MOF**, since the UML Profile is an extension mechanism to adapt the UML metamodel, which is based on *MetaObject Facility* [82] (MOF), to different domains. Considering Fig. 6 it can be noticed that **Ecore**[7] is one of the most used meta-metamodels. It therefore seems to be a good integration point and opportunity for DS(M)L compatibility in this domain. Since different approaches within the *Eclipse Modeling Project* (EMP) share Ecore as their representation, the possibility to use the extensive EMP tool-support as well

[7]The core of the *Eclipse Modeling Framework* [83] (EMF) includes a metamodel (Ecore) for describing models and providing runtime support.

as to reuse the Ecore model itself [84] can be considered as big advantage.

(E)BNF is also quite widely used, e.g., by [85], [86], [76], [29]. It is standardized[8] by the International Organization for Standardization[9] (ISO). Additionally, it can be converted to an Ecore model as well as the other way around. Both features foster reuse and compatibility with the EMP ecosystem.

As it can be seen in Fig. 6 there are almost as much publications using a custom tool chain, e.g., custom formalisms, parser and tools, as there are publications using (E)BNF. Those are collected under the term **Other**. A considerable amount of DS(M)Ls cannot be classified, because the used formalism is not specified in the publications and thus marked as **Unknown**. Not mentioning the underlying formalism, however, limits the possibility of reuse strongly. From the user's perspective, knowing the specific formalism is a crucial factor in order to use existing work published by third-parties.

6.4 Publication Trends (RQ4)

In order to answer **RQ4**, this section analyzes the publication trends in two fine-grained parts. The first part (cf. Section 6.4.1) covers the publication rate by year, to determine a general trend for DS(M)Ls in robotics. In the subsequent part (cf. Section 6.4.2) the main venues, DS(M)Ls are published to, are investigated. This also covers the distribution of venues regarding their domain (i.e. robotics- and software-oriented venues as well as hybrid ones).

6.4.1 Temporal distribution

Model-driven and domain-specific approaches are on the rise in robotics. We plotted the temporal distribution of the publications in this survey, as shown in Fig. 7. While from 1984 to 2009 only a few contributions (on average $\sim$2) each year were published, the amount of MDE and DSL related publications is highly increasing from 2010 on. This is equivalent to the start of the DSLRob workshop and the SIMPAR conference. The numbers, however, clearly exceed the amount of DSLRob publications per year, proving that this is more due to a general increase, rather than to the influence of single venues. Although the distribution clearly supports this overall positive trend, the anomaly in 2013 attracts attention. A reason for that might be that there was no SIMPAR in 2013 and SDIR did not publish any papers but slides only.

6.4.2 Venues

As seen in Fig. 7, the very first publication included in this survey was successfully submitted to the robotic conference IJRS (*green*) by Henderson *et al.* [58] in 1984. Since then ICRA (*blue*) is represented almost every year, although with great fluctuations regarding the number of publications. The third conference that appears in this survey is IROS (*light-blue*). From 1993 until 2010 it is represented with one to two publications per year. Starting in 2011 a continuous increase can be noticed. ICRA and IROS combined create a solid base of contributions.

Since 2009 an increasing variety of different venues is appearing. For instance: AURO, TRO and RAS, which were occasionally represented (especially from 1995-2002), are recurring again. The fact that new venues are also increasingly appearing stands out. Since 2010, the DSLRob workshop (*orange*) as well as the SIMPAR conference (*dark-yellow*) is particularly contributing to the field of DS(M)L-research.

Considering the percentage ratio of sighted (i.e. passed **IC**s) and finally accepted contributions (i.e. not excluded by **EC**s) of this survey, it can be noted that DS(M)L publications cover only a very small part of more generic robotic-conferences, such as ICRA and IROS. In case of ICRA, roughly 13 % of the sighted publications suffice the inclusion criteria of this survey. This ratio is very low, compared to workshops such as DSLRob (79 %), SDIR (23 %) and conferences, e.g., SIMPAR (35 %). Nevertheless, IROS and ICRA combined represent over 53 % of the included publications, closely followed by the DSLRob ($\sim$17 %) workshop.

The color range in Fig. 7 visualizes the difference between the representation of robotics- and software-oriented venues in this survey. Over 65 % (*blue* and *green*) of the DSL-related contributions were submitted to robotics-oriented venues, whereas hybrid venues, i.e. related to both robotics and software, are represented by over 30 % (*yellow* and *orange*). Only $\sim$4 % (*red* and *purple*) of the publications considered in this survey were submitted to purely software-related venues, such as ICSE and MODELS. This particular distribution, however, is the result of the selection and analysis process introduced in Section 5.

7 Discussion

This section discusses the qualitative research questions introduced in Section 2.2 that we think are important for i) language developers to enable language reuse, interoperability and discussing the core concepts, as well as ii) language users to allow assessing the availability and usability of the DS(M)Ls. As mentioned in Section 5.2, publications were annotated during the analysis process when being of particular interest or particular positive examples for any of the following research questions. In the following sections, the qualitative research questions are discussed along these examples.

We discuss these exemplary approaches and publications to extract best practices in terms of documentation, accessibility and evaluation of robotics DS(M)Ls to make suggestions to the community. The need for this became clear during analysis of the publications for this survey, as lots of the aspects discussed here were largely undocumented and/or were partially unaccessible.

[8] EBNF ISO/IEC 14977:1996
[9] http://www.iso.org/

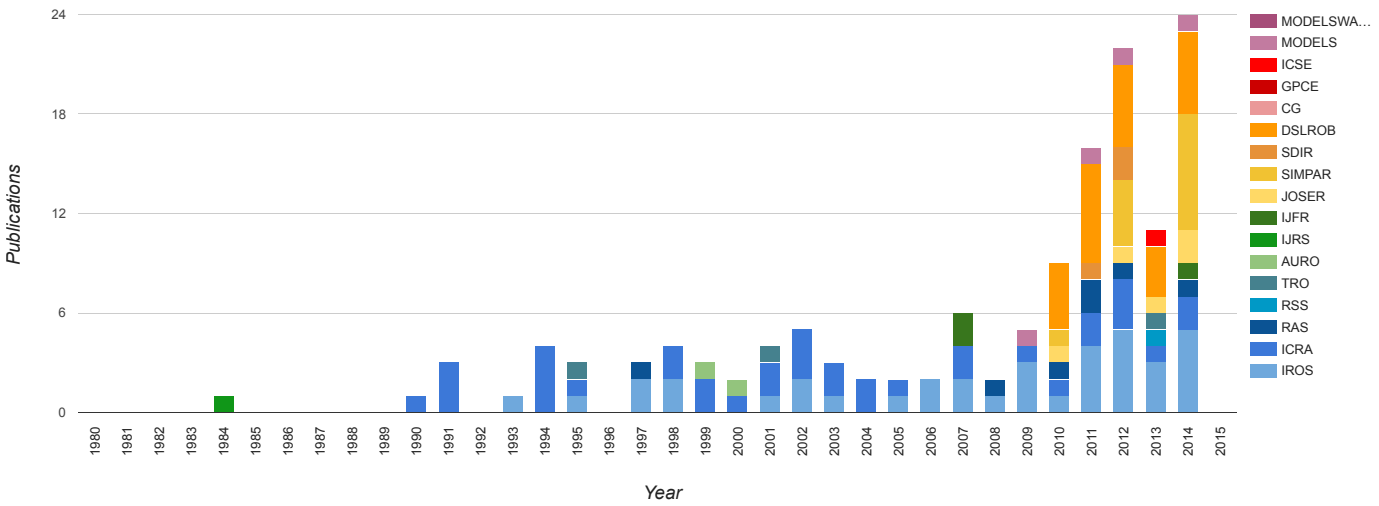

Fig. 7. Temporal distribution of the publications in this survey ranging from 1980 to 2015. Robotics- and software-related venues are distinguished by color: A color between *blue* and *green* represents robotics venues. The color space from *Yellow* to *orange* characterizes hybrid (i.e. robotics- and software-oriented) venues, while *red* and *purple* stand for pure software venues.

7.1 Accessibility and Documentation (RQ5)

An important factor for reuse of DS(M)Ls, scientific exchange and community building around DS(M)Ls in robotics is their accessibility and documentation. This comprises several factors like technical accessibility, e.g., download of the language or models, licensing, and documentation of the DS(M)L, its usage and execution context.

The majority of publications only give hints on the metamodel or show parts of it [62], [87], while some are also documenting their metamodel through exemplary models. Still that is not enough to reuse and properly interface between different approaches. Only a subset of the DS(M)Ls in this survey is documented in an exemplary manner to promote reuse as well as to enable actual usage. [88], [89], [12] allow interfacing by presenting a documentation of their metamodel. [35] generates HTML documents from Task Description Language (TDL) files with their Visual Design Tool, to facilitate the search for existing task definitions. [37], [34] provide a documentation for their approach, including install instructions. Besides that, the documentation of [27] also includes a complete description of all classes and packages of their EjsRL tool. A good way to push reuse even further is to provide tutorials and examples as done by [88], [37], [27].

To facilitate reuse, extension and integration of the developed DS(M)Ls or their metamodels, community-friendly open-source licensing schemes are required due their impact on usage, modification and redistribution [90]. Some publication already express to share this particular mindset by making their DS(M)Ls available for download as open-source software [91], [37], [12], [27], [41], [88], [34].

Making the deployment of the available software straightforward, is another crucial factor. This includes, apart from proper install instructions, a simple way to resolve dependencies (if necessary) [91]. [34] handles that matter in an exemplary manner. They provide three different methods to deploy their work: It can be installed using a Makefile[10] that automatically downloads all requirements, including the Eclipse IDE[11]. Furthermore, it can be installed from source via GitHub[12]. Finally, an eclipse update site can be used to integrate the software into an already existing instance of Eclipse.

7.2 Artifacts and Use-Case (RQ6)

To assess the intended use of the surveyed MDE and DS(M)L approaches, we looked at the artifacts generated (if any) and the context they are used in.

While model-based approaches can be used to generate visualizations of systems, e.g., of the system architecture [93] or hardware platforms [91], [69], the main use-case for DS(M)Ls is to generate executable code to perform experiments or provide supporting routines. The majority of the surveyed approaches is used for code generation, but roughly a fifth of the surveyed approaches is implemented as internal DS(M)Ls, e.g., [94], [95], [36], [79], and some of the languages use interpretation rather than code generation for their execution, e.g., [61], [96].

Among the approaches used for code generation, the ones identified within the same subdomains often cover similar use-cases and therefore are used for generation of similar

[10] A Makefile is used to specify how Make derives an, e.g., executable from source code. [92]

[11] http://www.eclipse.org/

[12] https://github.com/

artifacts. Approaches in the **Kinematics** subdomain often target simulation support, e.g., [69], [84], or are used for generation of controllers that can be embedded into motion control systems, e.g., [12], [80].

Approaches within the **Coordination** subdomain target to a large extent generation of state-chart or state machine based artifacts [60], [24], [97] since many robotics software frameworks and solution are based on them [60]. The following contributions [87], [89] generate code to realize motion tasks as state hierarchies, state transitions and import or extension of existing states.

Artifact generation from DS(M)Ls becomes especially powerful and suited for reuse if the accompanying toolchains support different parallel M2M and M2T transformations. Either to generate different artifacts like visualization, computational routines and glue code [93], or executable code for different programming languages or software platforms [29], [12], [24], [74]. [69] for example generates code for different hardware platforms and three different execution contexts: i) world files for a simulation engine, ii) C code for a virtual machine, and iii) XML configuration files for a specific compiler.

While the main target of DSL seems to be the automation of the software development, several approaches are also used for analysis and validation, e.g., [98], [49], or debugging, e.g., [99] who use their language for debugging and verifying correctness across distributed modular robots.

Robotics DS(M)L approaches, however, still use their models and languages mainly at design time [36]. Only a few of the surveyed approaches use the models to exploit the represented knowledge also at runtime, e.g., to model runtime variation points in the task at design time and bind and use them at runtime [100], [36], or to synthesize DSL programs while learning from demonstration as done by [54].

Another interesting example of using models after design time is presented by [101], [102], where DSLs are used for compact representation of programs and performing genetic programming with evolutionary optimization directly on the DSL code. Subsequent to the rearrangement of the DSL models through the genetic algorithms, GPL code is generated and tested for fitness to perform the next evolution step.

7.3 Platform (RQ7)

An important aspect of the artifacts and the model transformations discussed in the previous section is how tightly they are coupled to a certain platform and technical execution context, as discussed in Section 3.

A first differentiation to make in this case is usage of internal vs. externals DS(M)Ls.

While external DS(M)Ls are usually depending on a tool that transforms or executes their model and therefore introduce a platform-dependency, the target platform or technology can usually be chosen through their respective M2M and M2T transformations. Internal DS(M)Ls on the other hand are bound to the execution context, i.e. compiler or interpreter, of the host language. [80] discusses the impact of internal or external DSLs explicitly by implementing their model as an external DSL in Xtext and as an internal Prolog DSL. They draw the conclusion that the external DSL provides better tool support, but the internal DSL is easier from a developer's perspective in terms of implementation, and more convenient from a user's point of view, since it is executable out of the box without additional transformation steps. That said, the intended use cases of the internal and external languages are slightly different. While the primary focus of the internal DSL is on constraint checking and validation, the external DSL is also used to generate and build the corresponding infrastructure artifacts.

A second essential differentiation to make in terms of platform is between the DS(M)L being used in a interpretation or a generation manner. [61] does both and can either run their language models in an interpreter or generate C++ code from it for robots with restricted hardware resources.

For DS(M)Ls that are used in a generation manner, we differentiate between three classes of platform-dependency:

1) Proprietary robot programming languages such as *KRL* [103] and *RAPID* [104] typically target to a set of compatible platforms by a specific robot manufacturer and usually don't consider openness or platform independence[13].
2) Generation of artifacts that are tied to or dependent on a library stack, software framework or runtime environment, e.g., tied to a specific robotics framework [87], [89], [100], or targeted to be executed by a certain tool [105], [67]. While this introduces a bigger platform footprint during execution, custom tools can increase performance with respect to generic tools, e.g., in terms of parsing performance as discussed by [44].
   Some of the surveyed DS(M)Ls approaches come with exchangeable generators to explicitly allow use of the DS(M)Ls and their concepts in different frameworks or environments [74], [24]. [24] makes the platform explicit, by distinguishing between platform-independent and platform-specific models.
3) Transformation of the DS(M)L models directly to a general purpose language, e.g., Ada [88] or C++ [67], [29], [61], is the most platform-independent option. This reduced platform dependencies to a minimum, which is easier portable, even to embedded systems or system with restricted hardware capabilities [88], [61], easier to reuse and thereby eases scientific exchange. It also reduces assumptions about the platform from within the DS(M)L.

Interpretation of a DS(M)L is always being tied to a (DS(M)L-specific) interpreter [61], e.g., to directly executing

[13] Although neither KRL nor RAPID are included in the surveyed list of publications, we found them worthwhile to mention here as representatives for manufacturer-specific robot programming languages.

API calls during interpretation of the language models[96].

While platform-independence is often a motivation for the development and use of DS(M)L approaches, it may also be tied to a platform to provide tool support for this very specific platform. [106] for example bases the model on ROS nodes targeting its interactive programming, and thereby introducing platform dependency already on model level.

7.4 Evaluation (RQ8)

Evaluation of a DS(M)L based approach in its intended use-case is not only interesting from a developer's perspective, but also serves as a foundation for a decision from a user's perspective. A number of the surveyed publications evaluated the semantics or the generated artifacts. A surprising yet positive outcome of the analysis was that quite a number of the DS(M)Ls in this domain are evaluated not only in simulation, but on real hardware [89], [88], [12], [87], [107], and even on different platforms [62], [24], [107].

We can roughly differentiate two different kinds of evaluation approaches: qualitative and quantitative evaluation. Qualitative evaluation is often done by conceptual discussions based on examples and use-cases, e.g., [34], which is a suitable research methodology in software engineering research [108]. Several publications use case studies to discuss portability of their semantics to different platforms, e.g., [88], [62], [24]. [80] for example models some typical use-cases and shows how common errors can be avoided by using the proposed semantics and language. [109] discusses its approach extensively from a developer's perspective, using the language evaluation criteria from [110] including language design aspects, human factors, software engineering aspects, and application domain criteria. [80] discusses the impact of developing internal or external DS(M)Ls on the workflow and tools by implementing the model in both of them.

[111] lists four different quantitative benefits and corresponding metrics, that can be used to evaluate a model-based approach and can serve as a best practice:

1) **Efficiency**: This can be evaluated in terms of performance and memory utilization as it is done for example by [12]. They benchmarked their generated C++ code in its intended use-case, forward and inverse kinematics as well as dynamics on different numbers of degrees-of-freedom. In [112] the authors compare their approach to another framework and another library in terms of complexity to solve a targeted problem as well as quality of the generated code.
2) **Scalability** in terms of compilation time and system size.
3) **Productivity**: This can be evaluated in terms of size, effort or number of change requests, as done by [113]. They evaluate the usage of a DS(M)L from the developer's perspective against classical approaches by means of empirical software engineering. Non-functional aspects they covered comprise time spent for learning the technologies, effort for fixing bugs, component reuse and complexity of understanding reused software artifacts. [112] introduces its approach into an educational context and evaluates how much work students take to solve a problem. In [24] the authors conducted hardware experiments on a PR2 and a KUKA LWR and analyzed the necessary number of lines of code for platform-independent and robot/framework specific code.
4) **Reliability**, e.g., in terms of defects introduced in a period of time similar to what [80] does qualitatively, e.g., as done by [113], or in terms of number and duration of experiments as done by [32] who ran the system for several hours on 30 simulated robots.

Evaluation can also be used to show, if the DS(M)L is *complete* in terms of the evaluated examples. "Completeness" in this context means being able to express the domain problem, applications or typical examples of the domain entirely in the DS(M)L: *"Can typical use-cases of the DS(M)L be completely expressed within the DS(M)L?"* DS(M)Ls that for example specify controllers [12] or transformations [80] are not complete in this sense, since they need additional surrounding code or an application context.

In [114], [115] the authors state completeness of *URBI* for a set of examples, some of them listed in the paper: *"URBI, which is a command script language, is normally supposed to be used together with a client program written in C++ or Java, which will handle all the image processing and cognitive part of the robot behavior. However, it is possible to write quite complex and useful programs fully in URBI, without the use of an external client."* [88] might be complete in terms of simple applications of subsumption architectures, although it is vague to which extend the computational code of the modules can also be expressed within the DS(M)L. At least in the evaluation in their publication this part was implemented manually.

KUKA's KRL [103], [96] and ABB's RAPID [104] are complete, as they are typically used as sole language on the robot's control systems and can express entire industrial robot applications.

A slightly different completeness term is explicitly addressed by [116] who discuss how to prove completeness for their Motion Grammar in terms of the ability to cover the entire functional variability of the domain, i.e. the robot is able to respond to all situations, and how this helps proving correctness of their modeled systems.

To sum up, both, qualitative as well as quantitative evaluation can help language developers and potential users. While qualitative evaluation provides a first means to assess the general applicability to a certain problem domain, we recommend conducting also quantitative evaluation such as the ones discussed above to get a better grasp on the raised effectiveness of the development process, which is often the motivation for developing DS(M)Ls [6].

7.5 DS(M)L Development Process (RQ9)

In our previous survey [5], we already reported that very little is known about the process *how* DS(M)L developers identify and consolidate abstractions which on the one hand suit the domain best and on the other hand are the building blocks of DS(M)Ls. Unfortunately, also for the assessed articles we conclude that very little is reported about the DS(M)L development process itself. Even though, some articles report how they ground their DS(M)Ls, e.g., based on an ontology [74], [117], a formalism [80], [24], an architectural pattern [118], or a domain analysis [93], [119], [72] little is known about the involved stakeholders such as DS(M)L developers and domain experts, their requirements and interaction among each other. Ultimately, such a description could be used to define a robotics-specific DS(M)L development process model. In [120] such a process model has been identified in a reverse-engineered manner based on the insights gained during the development of the GDDL DSL [68]. However, as the model is reverse-engineered on the basis of one language and particular language objective (i.e. *task automation*) it is debatable to which extent the proposed process model can be generalized to other use cases and DS(M)L developments. Nevertheless, we would like to motivate the community to perform such reports as they provide lessons learned on *how* one could structure and perform DS(M)L development in robotics.

7.6 DS(M)L Usage (RQ10)

During the assessment of the publications we noted that several articles do not introduce a new DS(M)L, but make use of an existing approach to solve robotic-specific problems. We discuss some representative examples of usage and draw some conclusions relevant for both DS(M)L developers and users.

Within the domain of AI-based task planning and reasoning, declarative languages and formalism such as PDDL [121], ADL [122] and $C+$ [123] to name a few have been introduced. Some of these languages are used in the assessed papers to represent declarative knowledge in the context of robot plan optimization [124], to embed geometric reasoning in action descriptions [125] and to develop an integrated robot manipulation application where task planning capabilities are required [126]. We argue that the usage of these languages is not surprising as on the one hand the languages itself and underlying principles are well established and on the other hand planning frameworks usually require to specify the domain in one standard way. This at least holds true for task planner relying on the PDDL formalism.

Some articles are solely usage reports about applying general purpose model-based approaches to solve robotic problems. Most notably, the work by [127], [128] reports about the advantages and disadvantages of AADL [129] and SysML [10] to model, analyze, and validate structural and behavioral aspects of software for a robotic wheelchair. In a similar manner [130] adopt the AUTOSAR[14] methodology and conforming model-based tooling to the design of embedded robotic systems. Surprisingly we have not found any article which performs such a usage study for a core robotics DS(M)L. This might be an indication that robotic DS(M)Ls are not yet so widespread as general-purpose modeling approaches such as SysML and/or the available tooling supporting development is not mature enough as this is the case for general-purpose approaches.

Some articles mention and briefly describe usage of approaches included in this survey, e.g., [131] uses TDL [35] to model the executive of an autonomous mobile manipulator in the context of assembly tasks, [132] employs the XABSL language [37] to coordinate heterogeneous mobile robots, and the practicality of the logical sensor specification language [58] is reported by [133].

Due to the chosen keywords *specification language(s)* and *description language(s)* we found numerousness publications dealing with temporal logic languages, e.g., LTL. For instance, in [134] controllers are synthesized based on LTL formulas and in [135] LTL is applied among other formal languages to specify robot motions and actions. We report on these approaches here as DS(M)Ls and formal methods in general share several commonalities and goals such as the importance of models as a key towards systematic design, development and eventually correct-by-construction of robotics software. We argue that both communities should foster collaboration in order to make formal methods more practicable and accepted in robotics software development and to make DS(M)L approaches more well-founded in theory to foster work in the field of model validation and verification.

Quite interestingly, some approaches compose robotic DS(M)Ls and general-purpose modeling approaches. For instance, in [136] SysML [10] is used in combination with PDDL [121] for the task of manufacturing planning. Here, SysML is used to model manufacturing capabilities and process specifications whereas PDDL is used to determine acceptable plans. In [137] the BRICS Component Model (BCM) [138] and corresponding tooling is used in combination with 20sim[15] a modeling and simulation framework and graphical DSL for mechatronics systems. Here, the BCM is used to model the structural aspects of a robot motion-control architecture whereas 20sim is used to model the computational parts (aka. control algorithm). We argue that both from a DS(M)L developer and user perspective such reports are very much appreciated as they provide insights in when and how robotic DS(M)Ls can be used alone and/or in combination with general-purpose modeling languages.

[14] http://www.autosar.org/
[15] http://www.20sim.com/

8 Threats to Validity

We report on threats to internal and external validity of this survey according to the guidelines proposed by [139]. To avoid systematic errors within the survey and to enhance internal validity we formulated a well-defined selection process (see Section 5) and applied descriptive statistics to investigate the quantitative research questions (see Section 2.1). We obtained the list of potential candidates by a script performing keyword-based full-text searches on the *Google Scholar* corpus which we tested extensively and which is released[16] as open-source. We are aware that the selection of keywords impacts the number of potential candidate publications. For example, trough the addition of *specification language* to the set of keywords some of the earlier papers became part of the result set. Obviously, the set of keywords is not conclusive and may be extended by the introduction of further keywords. However, we argue that the current set of keywords represents the common vocabulary of the DS(M)L community established over the years. The temporal distribution reflects the DS(M)L publications that use the evolved vocabulary. Therefore, we excluded general terms such as *modeling* and *models*. To largely avoid systematic bias we applied the four-eyes principle during the paper analysis. To enhance external validity and generalizability of the insights we ensured to include high-ranked robotics, software engineering and model-driven engineering venues in our survey as assured trough the h5-index[17] obtained by *Google Scholar*. In summary, we are confident that threats to validity are minimal.

9 Synopsis

This contribution surveyed the available literature on domain-specific (modeling) languages, which addresses key concerns in robotics application development along a set of quantitative and qualitative research questions. The resulting mapping of publications to functional subdomains and the phases of an application development process shall serve as a reference to users and developers of DS(M)L approaches in robotics and related domains.

Our analysis yields that a major fraction of the surveyed publications addresses architectural concerns or mature subdomains such as robot motion, while only a minor number of papers target comparably recent subdomains such as force control. With regard to architectural concerns, the extended decomposition of the architecture and programming subdomain introduced within this contribution, provides further insight into existing research approaches. From the viewpoint of the robotics application development process, we showed that many DS(M)L approaches use models and model-driven engineering to generate artifacts for the capability building phase, while only a few approaches consider the application of models at runtime in robotics systems.

The presented quantitative analysis also clearly indicates that DS(M)Ls are currently an active research field given the rising number of publications at robotics conferences. That said, the robotics DS(M)L community seems to lack comparable acknowledgment at the general modeling and software engineering venues.

From a technical perspective, we also must assess that compatibility and reuse of different DSLs and approaches in a modular approach is still an issue for conceptual and technological reasons, i.e., due to the fragmentation in terms of modeling tools and formalisms. While the *Eclipse Modeling Project* may serve as an integration platform due to its wide use, further research on language modularization and reuse seems required.

We further discussed, how different approaches to accessibility and documentation as well as evaluation and platform-dependency affect the availability and usability of a DS(M)L approach. Given the current status of many DS(M)L publications with regard to their technical accessibility and thus reproducibility, we hope that this survey stimulates a discussion on how meta-models, languages and experience reports may be shared in a reproducible way within the robotics community. Along this line, we highlighted best practices that may be considered to foster improved collaboration and development within the DS(M)L community.

Future work on the basis of this survey may include a more detailed analysis of the specific concepts presented in the surveyed publications within one of the robotics subdomains. For instance, we consider this relevant for the field of architectures and programming given the number of publications addressing its concerns. To facilitate further work on the basis of this survey, we share the automated query code on GitHub. Furthermore, following the idea of the *EMF Concrete Syntax Zoo*[18] we intend to continuously maintain this overview as an online *Robotics DSL Zoo*[19] and invite the community to provide feedback and contribute.

10 Publications

To provide some kind of *map* for DS(M)L developers and users of the domain, this section provides an overview of all surveyed publications, sorted by their associated subdomains and development phases. Note, that publications appear multiple times as they have multiple subdomains and development phases annotated from the analysis process. When following the references to the bibliography at the end of this survey, links at the end of each bibliography entry link back to the sections where the respective publication is mentioned, e.g., the subdomain and development phase it is associated with, or a section in which it is discussed and used as an example.

[16] https://github.com/corlab/dslzoo/tree/query

[17] https://scholar.google.de/citations?view_op=top_venues&hl=en&vq=eng_robotics

[18] http://www.emftext.org/index.php/EMFText_Concrete_Syntax_Zoo

[19] https://corlab.github.io/dslzoo/

10.1 Kinematics

Platform Building [69], [27], [140], [67]
Functional Design [12], [141], [27], [42], [140], [142]
Capability Building [80], [28], [69], [143], [12], [27], [42], [140], [24], [142], [67], [79]

10.2 Mechanisms and Actuation

Platform Building [144], [81], [140], [65], [145], [146], [147], [68]
Functional Design [141], [140], [147]
Capability Building [148], [144], [149], [150], [140], [65], [145], [146], [151], [68]

10.3 Dynamics

Platform Building [27]
Functional Design [12], [27]
Capability Building [28], [143], [12], [27]

10.4 Motion Planning

Scenario Building [77]
Product Maintenance [48], [47]
Platform Building [144], [68]
Functional Design [54], [77], [42], [45]
Capability Building [48], [144], [47], [53], [44], [54], [77], [56], [42], [45], [107], [105], [98], [68], [43]

10.5 Sensing and Estimation

Product Maintenance [152], [52]
Platform Building [81], [52], [65], [146], [119]
Functional Design [63], [58]
Capability Building [152], [150], [63], [58], [52], [65], [146], [151], [119]

10.6 Force Control

Capability Building [25], [24]

10.7 Motion Control

Scenario Building [77]
Platform Building [46], [145], [66], [153], [67]
Functional Design [154], [77], [141], [42], [45], [142], [89]
System Deployment [96], [93], [66]
Capability Building [48], [96], [93], [25], [149], [154], [102], [44], [155], [77], [156], [42], [46], [145], [24], [45], [142], [66], [153], [67], [89], [157], [79], [43]
Product Maintenance [48]

10.8 Reasoning Methods

Scenario Building [117], [78], [51]
Platform Building [71], [49], [52], [78], [147]
Functional Design [76], [71], [54], [63], [158], [147]
System Deployment [76], [49]
Capability Building [76], [53], [99], [54], [49], [63], [52], [95], [39], [117], [78], [50], [158], [36]
Product Deployment [76]
System Benchmarking [99]
Product Maintenance [76], [99], [52]

10.9 Architectures and Programming

In accordance with the more fine-grained partitioning of the `Architectures and Programming` subdomain introduced and motivated in Section 6.1 we provide a more detailed overview of the respective publications and their subdisciplines and addressed development phases.

10.9.1 Concurrency

Platform Building [49], [140], [46], [66], [153]
Functional Design [140], [159]
System Deployment [115], [100], [49], [66], [159]
Capability Building [115], [100], [160], [41], [109], [32], [161], [49], [140], [46], [162], [107], [163], [66], [153], [159], [98], [112], [35]

10.9.2 Data Persistence

Scenario Building [117], [51]
Platform Building [84]
Capability Building [84], [117]

10.9.3 Control and Handling of Events

Scenario Building [164]
Platform Building [165], [49], [140], [46], [153], [119]
Functional Design [166], [74], [75], [63], [140], [159], [167], [89]
System Deployment [115], [168], [169], [170], [171], [165], [74], [49], [37], [86], [159]
Capability Building [115], [87], [168], [48], [148], [172], [62], [173], [28], [160], [169], [41], [109], [170], [171], [166], [174], [47], [53], [165], [61], [74], [155], [75], [150], [161], [175], [49], [63], [95], [39], [140], [46], [97], [162], [101], [176], [37], [107], [163], [86], [153], [50], [159], [94], [119], [167], [98], [112], [35], [36], [177], [89], [38], [60], [40], [178]
Product Deployment [75]
Product Maintenance [48], [172], [47], [75], [167]

10.9.4 Error and Exception Handling

Scenario Building [164]
Functional Design [154], [167]
System Deployment [168]

Capability Building [168], [48], [172], [154], [99], [150], [39], [50], [167], [36]
System Benchmarking [99]
Product Maintenance [48], [172], [99], [167]

10.9.5 Distribution of Components

Platform Building [179], [69], [49], [66]
Functional Design [74], [73], [159], [59]
System Deployment [113], [100], [169], [170], [74], [49], [73], [86], [66], [159], [59]
Capability Building [179], [113], [100], [160], [169], [170], [69], [180], [99], [74], [32], [49], [156], [73], [181], [86], [57], [66], [159], [59]
System Benchmarking [99]
Product Maintenance [99]

10.9.6 Security and Safety

Scenario Building [77], [51]
Product Maintenance [182], [167]
Functional Design [182], [183], [77], [45], [70], [167]
Capability Building [182], [183], [180], [77], [45], [70], [167]

10.9.7 Interaction and Presentation

Scenario Building [77]
Platform Building [85], [165], [27]
Functional Design [85], [54], [77], [27], [45], [70]
System Deployment [113], [88], [171], [165]
Capability Building [106], [87], [62], [113], [173], [88], [171], [85], [53], [165], [54], [77], [156], [27], [45], [70], [177]

10.9.8 Architectural Styles

Scenario Building [78]
Platform Building [78]
Functional Design [183], [184], [185]
System Deployment [118], [93], [88]
Capability Building [118], [93], [172], [88], [41], [183], [174], [184], [185], [175], [95], [39], [78], [186], [94], [98], [38], [60], [157]
Product Maintenance [172]

10.9.9 Architectural Structures and Viewpoints

Scenario Building [117], [78]
Platform Building [179], [84], [144], [64], [69], [85], [52], [27], [153], [78], [72], [119]
Functional Design [76], [64], [85], [74], [184], [34], [73], [27], [72], [59]
System Deployment [118], [76], [93], [113], [100], [88], [169], [170], [171], [74], [73], [86], [59]
Capability Building [179], [84], [118], [152], [76], [80], [148], [93], [113], [100], [88], [144], [169], [64], [25], [170], [69], [171], [85], [74], [184], [32], [34], [73], [52], [181], [27], [117], [86], [57], [153], [78], [186], [72], [119], [59], [36], [177], [157]
Product Deployment [76]
Product Maintenance [152], [76], [52]

10.9.10 Architecture Design Decisions

Scenario Building [117]
Platform Building [69]
Functional Design [184], [185]
Capability Building [69], [102], [184], [185], [117]

10.9.11 Design Patterns

Functional Design [154], [34]
Capability Building [149], [154], [34], [105]

10.9.12 Families of Programs and Frameworks

Platform Building [119]
Functional Design [187], [76], [75], [34], [73]
System Deployment [76], [96], [73]
Capability Building [187], [76], [96], [25], [75], [34], [73], [119]
Product Deployment [76], [75]
Product Maintenance [76], [75]

References

[1] A. van Deursen, P. Klint, and J. Visser, "Domain-Specific Languages: An Annotated Bibliography," *ACM Sigplan Notices*, 2000. [Online]. Available: http://www.st.ewi.tudelft.nl/~arie/papers/dslbib.pdf 1, 3.1

[2] A. Rodrigues da Silva, "Model-driven Engineering: A Survey Supported by the Unified Conceptual Model," *Computer Languages, Systems and Structures*, 2015. 1, 3, 3.1, 3.2

[3] G. Biggs and B. Macdonald, "A Survey of Robot Programming Systems," *Proceedings of the Australasian conference on robotics and automation*, pp. 1–3, 2003. [Online]. Available: http://pdf.aminer.org/000/355/292/towards_programming_tools_for_robots_that_integrate_probabilistic_computation_and.pdf 1

[4] G. Cattivera and G. L. Casalaro, "Model-Driven Engineering for Mobile Robot Systems: A Systematic Mapping Study," Master's thesis, Mälardalen University, 2015. 1

[5] A. Nordmann, N. Hochgeschwender, and S. Wrede, "A Survey on Domain-Specific Languages in Robotics," in *International Conference on Simulation, Modeling and Programming for Autonomous Robots*, Bergamo, 2014. 1, 2.1, 5, 7.5

[6] M. Mernik, J. Heering, and A. M. Sloane, "When and How to Develop Domain-Specific Languages," *ACM Computing Surveys*, vol. 37, no. 4, pp. 316–344, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1118890.1118892 2.2, 3.2, 7.4

[7] OMG, "Model Driven Architecture Guide Rev. 2.0," Tech. Rep. June, 2014. [Online]. Available: http://www.omg.org/cgi-bin/doc?omg/03-06-01 1, 3.2

[8] M. Völter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. Kats, E. Visser, and G. Wachsmuth, *DSL Engineering – Designing, Implementing and Using Domain-Specific Languages*. CreateSpace Independent Publishing Platform, 2013. [Online]. Available: http://dslbook.org 3.1, 3.2

[9] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, 2005. 3.1, 6.3

[10] L. Delligatti, *SysML Distilled: A Brief Guide to the Systems Modeling Language*, 1st ed. Addison-Wesley Professional, 2013. 3.1, 7.6

[11] S. Gerard and B. Selic, "The UML - MARTE standardized profile," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 17, Seoul, Korea, 2008, pp. 6909–6913. 3.1

[12] M. Frigerio, J. Buchli, and D. G. Caldwell, "Model based code generation for kinematics and dynamics computations in robot controllers," in *Workshop on Software Development and Integration in Robotics, St. Paul, Minnesota, USA*, 2012. 3.1, 6.1, 7.1, 7.2, 7.4, 1, 7.4, 10.1, 10.3

[13] R. C. Gronback, *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. Addison-Wesley Professional, 2009. 3.2

[14] Jetbrains.com, "Jetbrains Meta Programming System," *http://www.jetbrains.com/mps/*, 2003. [Online]. Available: http://www.jetbrains.com/mps 3.2

[15] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg, 2008. 4.1, 3, 6.1

[16] G. K. Kraetzschmar, A. Shakhimardanov, J. Paulus, N. Hochgeschwender, and M. Reckhaus, "Best practice in robotics (brics) deliverable d-2.2: Specifications of architectures, modules, modularity, and interfaces for the brocre software platform and robot control architecture workbench," 2010, project Deliverable BRICS: D2.2. 4.2

[17] R. Bischoff, T. Guhl, E. Prassler, W. Nowak, G. Kraetzschmar, H. Bruyninckx, P. Soetens, M. Haegele, A. Pott, P. Breedveld, J. Broenink, D. Brugali, and N. Tomatis, "Brics - best practice in robotics," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, June 2010, pp. 1–8. 4.2

[18] L. L. Beck and T. E. Perkins, "A survey of software engineering practice: Tools, methods, and results." *IEEE Trans. Software Eng.*, vol. 9, no. 5, pp. 541–561, 1983. 4.2

[19] I. Sommerville, *Software Engineering (7th Edition)*. Pearson Addison Wesley, 2004. 4.2

[20] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2003. 4.2

[21] D. C. Schmidt, "Guest editor's introduction: Model-driven engineering," *Computer*, vol. 39, no. 2, pp. 25–31, Feb. 2006. [Online]. Available: http://dx.doi.org/10.1109/MC.2006.58 4.2

[22] T. Stahl, M. Voelter, and K. Czarnecki, *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, 2006. 4.2

[23] A. Kossiakoff, W. N. Sweet, S. J. Seymour, and S. M. Biemer, *Systems Engineering Principles and Practice*. John Wiley & Sons, Inc., 2011. 4.2

[24] M. Klotzbucher, R. Smits, H. Bruyninckx, and J. 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. 6.1, 7.2, 2, 7.4, 3, 7.5, 10.1, 10.6, 10.7

[25] A. Angerer, A. Hoffmann, A. Schierl, M. Vistein, and W. Reif, "Robotics api: object-oriented software development for industrial robots," *Journal of Software Engineering for Robotics*, vol. 4, no. 1, pp. 1–22, 2013. 6.1, 10.6, 10.7, 10.9.9, 10.9.12

[26] M. T. Mason, "Compliance and force control for computer controlled manipulators," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 11, no. 6, pp. 418–432, June 1981. 6.1

[27] C. A. Jara, F. A. Candelas, P. Gil, F. Torres, F. Esquembre, and S. Dormido, "Ejs+ ejsrl: An interactive tool for industrial robots simulation, computer vision and remote operation," *Robotics and Autonomous systems*, vol. 59, no. 6, pp. 389–401, 2011. 6.1, 6.2, 7.1, 10.1, 10.3, 10.9.7, 10.9.9

[28] E. Aertbeliën and J. De Schutter, "etasl/etc: A constraint-based task specification language and robot controller using expression graphs," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 1540–1546. 6.1, 10.1, 10.3, 10.9.3

[29] M. Frigerio, J. Buchli, and D. Caldwell, "A Domain Specific Language for Kinematic Models and Fast Implementations of Robot Dynamics Algorithms," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2013. [Online]. Available: http://arxiv.org/abs/1301.7190 6.1, 6.3, 7.2, 3

[30] P. Bourque and R. E. Fairley, Eds., *Guide to the Software Engineering Body of Knowledge - SWEBOK v3.0*, 3rd ed. IEEE Computer Society, 2014. [Online]. Available: http://www.swebok.org 6.1, 6.1, 1

[31] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*. Reading, MA: Addison Wesley, 1995. 6.1

[32] S. Fleury, M. Herrb, and R. Chatila, "G en om: A tool for the specification and the implementation of operating modules in a distributed robot architecture," in *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2. IEEE, 1997, pp. 842–849. 6.1, 4, 10.9.1, 10.9.5, 10.9.9

[33] D. Vanthienen, M. Klotzbücher, and H. Bruyninckx, "The 5c-based architectural composition pattern: lessons learned from re-developing the itasc framework for constraint-based robot programming," *JOSER: Journal of Software Engineering for Robotics*, vol. 5, no. 1, pp. 17–35, 2014. 6.1

[34] L. Gherardi and D. Brugali, "Modeling and reusing robotic software architectures: the hyperflex toolchain," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6414–6420. 6.1, 6.2, 7.1, 7.4, 10.9.9, 10.9.11, 10.9.12

[35] R. Simmons and D. Apfelbaum, "A task description language for robot control," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1998, pp. 1931–1937. 6.1, 6.2, 7.1, 7.6, 10.9.1, 10.9.3

[36] A. Steck and C. Schlegel, "Managing execution variants in task coordination by exploiting design-time models at run-time," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2064–2069. 6.1, 7.2, 10.8, 10.9.3, 10.9.4, 10.9.9

[37] M. Loetzsch, M. Risler, and M. Jungel, "Xabsl-a pragmatic approach to behavior engineering," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 5124–5129. 6.1, 7.1, 7.6, 10.9.3

[38] S. Tousignant, E. Van Wyk, and M. Gini, "An overview of xrobots: A hierarchical state machine based language," in *ICRA-2011 Workshop on Software Development and Integration in Robotics, Shanghai, China May*, 2011, pp. 9–13. 6.1, 6.2, 10.9.3, 10.9.8

[39] S. Joyeux, F. Kirchner, and S. Lacroix, "Managing plans: Integrating deliberation and reactive execution schemes," *Robotics and Autonomous Systems*, vol. 58, no. 9, pp. 1057–1066, 2010. 6.1, 10.8, 10.9.3, 10.9.4, 10.9.8

[40] S. Wang and K. G. Shin, "Reconfigurable software for open architecture controllers," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 4090–4095. 6.1, 10.9.3

[41] V. Berenz and K. Suzuki, "Targets-drives-means: A declarative approach to dynamic behavior specification with higher usability," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 545–555, 2014. 6.1, 6.2, 7.1, 10.9.1, 10.9.3, 10.9.8

[42] Y. J. Kanayama and C. T. Wu, "It's time to make mobile robots programmable," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 329–334. 6.1, 10.1, 10.4, 10.7

[43] F. Zhang, M. Goldgeier, and P. S. Krishnaprasad, "Control of small formations using shape coordinates," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 2510–2515. 6.1, 10.4, 10.7

[44] N. Dantam, A. Hereid, A. D. Ames, and M. Stilman, "Correct software synthesis for stable speed-controlled robotic walking." in *Robotics: Science and Systems*, 2013. 6.1, 2, 10.4, 10.7

[45] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "From structured english to robot motion," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 2717–2722. 6.1, 6.2, 10.4, 10.7, 10.9.6, 10.9.7

[46] T. W. Kim and J. Yuh, "Task description language for underwater robots," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2003, pp. 565–570. 6.1, 6.2, 10.7, 10.9.1, 10.9.3

[47] N. Dantam, P. Koine, and M. Stilman, "The motion grammar for physical human-robot games," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5463–5469. 6.1, 10.4, 10.9.3

[48] J. S. Laursen, J. P. Buch, L. C. Sø rensen, D. Kraft, H. G. P. L.-P. Ellekilde, and U. P. Schultz, "Towards Error Handling in a DSL for Robot Assembly Tasks," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2014. [Online]. Available: http://arxiv.org/abs/1412.4538 6.1, 10.4, 10.7, 10.9.3, 10.9.4

[49] N. Gobillot, C. Lesire, and D. Doose, "A modeling framework for software architecture specification and validation," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 303–314. 6.1, 6.2, 7.2, 10.8, 10.9.1, 10.9.3, 10.9.5

[50] M. OBrien, R. C. Arkin, D. Harrington, D. Lyons, and S. Jiang, "Automatic verification of autonomous robot missions," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 462–473. 6.1, 10.8, 10.9.3, 10.9.4

[51] V. Raman, B. Xu, and H. Kress-Gazit, "Avoiding forgetfulness: Structured english specifications for high-level robot control with implicit memory," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1233–1238. 6.1, 10.8, 10.9.2, 10.9.6

[52] N. Hochgeschwender, S. Schneider, H. Voos, and G. K. Kraetzschmar, "Declarative specification of robot perception architectures," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 291–302. 6.1, 6.2, 10.5, 10.8, 10.9.9

[53] N. Dantam, I. Essa, and M. Stilman, "Linguistic transfer of human assembly tasks to robots," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 237–242. 6.1, 10.4, 10.8, 10.9.3, 10.9.7

[54] A. Feniello, H. Dang, and S. Birchfield, "Program synthesis by examples for object repositioning tasks," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4428–4435. 6.1, 6.3, 7.2, 10.4, 10.8, 10.9.7

[55] F. Ingrand and F. Py, "An execution control system for autonomous robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1333–1338. 6.1

[56] M. A. Hsieh, A. Cowley, J. F. Keller, L. Chaimowicz, B. Grocholsky, V. Kumar, C. J. Taylor, Y. Endo, R. C. Arkin, B. Jung *et al.*, "Adaptive teams of autonomous aerial and ground robots for situational awareness," *Journal of Field Robotics*, vol. 24, no. 11-12, pp. 991–1014, 2007. 6.1, 10.4

[57] A. Mallet, C. Pasteur, M. Herrb, S. Lemaignan, and F. Ingrand, "Genom3: Building middleware-independent robotic components," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4627–4632. 6.2, 10.9.5, 10.9.9

[58] T. Henderson and E. Shilcrat, "Logical sensor systems," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 169–193, 1984. 6.2, 6.4.2, 7.6, 10.5

[59] C. Schlegel, A. Steck, D. Brugali, and A. Knoll, "Design abstraction and processes in robotics: From code-driven to model-driven engineering," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2010, pp. 324–335. 6.2, 10.9.5, 10.9.9

[60] S. Tousignant, E. Van Wyk, and M. Gini, "Xrobots: A flexible language for programming mobile robots based on hierarchical state machines," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1773–1778. 6.2, 7.2, 10.9.3, 10.9.8

[61] T. J. de Haas, T. Laue, and T. Rofer, "A scripting-based approach to robot behavior engineering using hierarchical generators," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4736–4741. 6.2, 7.2, 7.3, 3, 7.3, 10.9.3

[62] M. Reckhaus, N. Hochgeschwender, P. G. Ploeger, and G. K. Kraetzschmar, "A Platform-Independent Programming Environment for Robot Control," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2010. [Online]. Available: http://arxiv.org/abs/1010.0886 6.2, 7.1, 7.4, 10.9.3, 10.9.7

[63] J. L. Gordillo, "L e: a high level language for specifying vision verification tasks," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1433–1439. 6.2, 10.5, 10.8, 10.9.3

[64] M. Anderson, J. Bowman, and P. Kilgo, "Rdis: Generalizing domain concepts to specify device to framework mappings," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1836–1841. 6.2, 10.9.9

[65] P. Kilgo, E. Syriani, and M. Anderson, "A visual modeling language for rdis and ros nodes using atom3," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2012, pp. 125–136. 6.2, 10.2, 10.5

[66] M. Morelli and M. Di Natale, "Control and scheduling co-design for a simulated quadcopter robot: A model-driven approach," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 49–61. 6.2, 10.7, 10.9.1, 10.9.5

[67] A. K. Ramadorai, U. Ganapathy, and F. Guida, "A generic kinematics software package," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 3331–3336. 6.2, 2, 3, 10.1, 10.7

[68] S. Schneider, N. Hochgeschwender, and G. K. Kraetzschmar, "Declarative specification of task-based grasping with constraint validation," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 919–926. 6.2, 7.5, 10.2, 10.4

[69] M. Bordignon, K. Stoy, and U. P. Schultz, "Generalized programming of modular robots through kinematic configurations," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 3659–3666. 6.2, 7.2, 10.1, 10.9.5, 10.9.9, 10.9.10

[70] D. M. Lyons, R. C. Arkin, P. Nirmal, T.-M. Liu, J. Deeb *et al.*, "Getting it right the first time: Robot mission guarantees in the presence of uncertainty," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5292–5299. 6.2, 10.9.6, 10.9.7

[71] S. Bocionek, P. Buchka, and J. Schweiger, "Generating expert systems for configuration tasks," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, 1990, pp. 896–901. 6.2, 10.8

[72] A. Ramaswamy, B. Monsuez, and A. Tapus, "Saferobots: A model-driven framework for developing robotic systems," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 1517–1524. 6.2, 7.5, 10.9.9

[73] N. Hochgeschwender, L. Gherardi, A. Shakhirmardanov, G. K. Kraetzschmar, D. Brugali, and H. Bruyninckx, "A model-based approach to software deployment in robotics," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3907–3914. 6.2, 10.9.5, 10.9.9, 10.9.12

[74] S. Dhouib, S. Kchir, S. Stinckwich, T. Ziadi, and M. Ziane, "Robotml, a domain-specific language to design, simulate and deploy robotic applications," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2012, pp. 149–160. 6.2, 7.2, 2, 7.5, 10.9.3, 10.9.5, 10.9.9

[75] F. Fleurey and A. Solberg, "A domain specific modeling language supporting specification, simulation and execution of dynamic adaptive systems," in *Model Driven Engineering Languages and Systems*. Springer, 2009, pp. 606–621. 6.2, 10.9.3, 10.9.12

[76] J. Ingles-Romero, A. Lotz, C. Vicente-Chicote, and C. Schlegel, "Dealing with Run-Time Variability in Service Robotics: Towards a DSL for Non-Functional Properties," in *Workshop on Domain-Specific Languages and models for Robotic systems*, no. iv, 2010. 6.2, 6.3, 10.8, 10.9.9, 10.9.12

[77] C. Finucane, G. Jing, and H. Kress-Gazit, "Ltlmop: Experimenting with language, temporal logic and robot control," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 1988–1993. 6.2, 10.4, 10.7, 10.9.6, 10.9.7

[78] F. R. Noreils and R. G. Chatila, "Plan execution monitoring and control architecture for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 11, no. 2, pp. 255–266, 1995. 6.2, 10.8, 10.9.8, 10.9.9

[79] D. Vanthienen, M. Klotzbuucher, J. De Schutter, T. De Laet, and H. Bruyninckx, "Rapid application development of constrained-based task modelling and execution using domain specific languages," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1860–1866. 6.3, 7.2, 10.1, 10.7

[80] T. D. Laet, W. Schaekers, J. de Greef, and H. Bruyninckx, "Domain Specific Language for Geometric Relations between Rigid Bodies targeted to Robotic Applications," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2012. [Online]. Available: http://arxiv.org/abs/1304.1346 6.3, 7.2, 7.3, 7.4, 4, 7.4, 7.5, 10.1, 10.9.9

[81] J. A. Fryer and G. T. McKee, "Resource modelling and combination in modular robotics systems," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 4. IEEE, 1998, pp. 3167–3172. 6.3, 10.2, 10.5

[82] W. Tang, "Meta object facility," in *Encyclopedia of Database Systems*. Springer, 2009, pp. 1722–1723. 6.3

[83] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, *EMF: eclipse modeling framework*. Pearson Education, 2008. 7

[84] S. Blumenthal and H. Bruyninckx, "Towards a Domain Specific Language for a Scene Graph based Robotic World Model," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2013. 6.3, 7.2, 10.9.2, 10.9.9

[85] O. Causse and J. L. Crowley, "A man machine interface for a mobile robot," in *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 1. IEEE, 1993, pp. 327–335. 6.3, 10.9.7, 10.9.9

[86] A. Mallet, S. Fleury, and H. Bruyninckx, "A specification of generic robotics software components: future evolutions of g en o m in the orocos context," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3. IEEE, 2002, pp. 2292–2297. 6.3, 10.9.3, 10.9.5, 10.9.9

[87] A. Angerer, R. Smirra, A. Hoffmann, A. Schierl, M. Vistein, and W. Reif, "A Graphical Language for Real-Time Critical Robot Commands," in *Workshop on Domain-Specific Languages and models for Robotic systems*, Tsukuba, 2012. 7.1, 7.2, 2, 7.4, 10.9.3, 10.9.7

[88] P. Trojanek, "Model-Driven Engineering Approach to Design and Implementation of Robot Control System," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2011. [Online]. Available: http://arxiv.org/abs/1302.5085 7.1, 3, 7.4, 7.4, 10.9.7, 10.9.8, 10.9.9

[89] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, "A new skill based robot programming language using uml/p statecharts," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 461–466. 7.1, 7.2, 2, 7.4, 10.7, 10.9.3

[90] A. Morin, J. Urban, and P. Sliz, "A quick guide to software licensing for the scientist-programmer," *PLoS Comput Biol*, vol. 8, no. 7, p. e1002598, 2012. 7.1

[91] W. Meeussen, J. Hsu, and R. Diankov, "Unified Robot Description Format (URDF)," 2009. [Online]. Available: http://www.ros.org/wiki/urdf 7.1, 7.2

[92] S. I. Feldman, "Make - a program for maintaining computer programs," *Software: Practice and experience*, vol. 9, no. 4, pp. 255–265, 1979. 10

[93] A. Nordmann and S. Wrede, "A Domain-Specific Language for Rich Motor Skill Architectures," in *Workshop on Domain-Specific Languages and models for Robotic systems*, Tsukuba, 2012. 7.2, 7.5, 10.7, 10.9.8, 10.9.9

[94] J. Peterson, G. D. Hager, and P. Hudak, "A language for declarative robotic programming," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1144–1151. 7.2, 10.9.3, 10.9.8

[95] I. D. Horswill, "Functional programming of behavior-based systems," *Autonomous Robots*, vol. 9, no. 1, pp. 83–93, 2000. 7.2, 10.8, 10.9.3, 10.9.8

[96] H. Mühe, A. Angerer, A. Hoffmann, and W. Reif, "On Reverse-Engineering the KUKA Robot Language," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2010. [Online]. Available: http://arxiv.org/abs/1009.5004 7.2, 7.3, 7.4, 10.7, 10.9.12

[97] M. Klotzbücher and H. Bruyninckx, "Coordinating robotic tasks and systems with rfsm statecharts," *JOSER: Journal of Software Engineering for Robotics*, vol. 3, no. 1, pp. 28–56, 2012. 7.2, 10.9.3

[98] N. Rugg-Gunn and S. Cameron, "A formal semantics for multiple vehicle task and motion planning," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 2464–2469. 7.2, 10.4, 10.9.1, 10.9.3, 10.9.8

[99] M. De Rosa, J. Campbell, P. Pillai, S. Goldstein, P. Lee, and T. Mowry, "Distributed watchpoints: Debugging large multi-robot systems," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3723–3729. 7.2, 10.8, 10.9.4, 10.9.5

[100] A. Steck and C. Schlegel, "Towards Quality of Service and Resource Aware Robotic Systems through Model-Driven Software Development," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2010, p. 6. [Online]. Available: http://arxiv.org/abs/1009.4877 7.2, 2, 10.9.1, 10.9.5, 10.9.9

[101] J. Kubica and E. Rieffel, "Creating a smarter membrane: Automatic code generation for modular self-reconfigurable robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1. IEEE, 2002, pp. 793–800. 7.2, 10.9.3

[102] R. Burbidge, J. H. Walker, and M. S. Wilson, "Grammatical evolution of a robot controller," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 357–362. 7.2, 10.7, 10.9.10

[103] KRL, "KUKA System Software 5.5 - Operating and Programming Instructions for System Integrators," KUKA Roboter GmbH, Tech. Rep., 2009. 1, 7.4

[104] RAPID, "RAPID Overview," ABB Robotics Products, Tech. Rep., 1998. 1, 7.4

[105] H. Mosemann and F. M. Wahl, "Automatic decomposition of planned assembly sequences into skill primitives," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 5, pp. 709–718, 2001. 2, 10.4, 10.9.11

[106] S. Adam and U. P. Schultz, "Towards Interactive, Incremental Programming of ROS Nodes," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2014. [Online]. Available: http://arxiv.org/abs/1412.4714 7.3, 10.9.7

[107] D. C. MacKenzie, J. M. Cameron, and R. C. Arkin, "Specification and execution of multiagent missions," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1995, pp. 51–58. 7.4, 10.4, 10.9.1, 10.9.3

[108] P. Runeson and M. Höst, "Guidelines for Conducting and Reporting Case Study Research in Software Engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009. 7.4

[109] G. Biggs and B. A. MacDonald, "Evaluating a reactive semantics for robotics," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1854–1859. 7.4, 10.9.1, 10.9.3

[110] J. Howatt, "A project-based approach to programming language evaluation," *ACM SIGPLAn Notices*, vol. 30, no. 7, pp. 37–40, 1995. 7.4

[111] T. Özgür, "Comparison of Microsoft DSL Tools and Eclipse Modeling Frameworks for Domain-Specific Modeling in the Context of the Model-Driven Development," Master, Blekinge Institute of Technology, 2007. 7.4

[112] A. Rusakov, J. Shin, and B. Meyer, "Simple concurrency for robotics with the roboscoop framework," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 1563–1569. 1, 3, 10.9.1, 10.9.3

[113] A. Romero-Garcés, L. Manso, M. Gutierrez, R. Cintas, and P. Bustos, "Improving the Lifecycle of Robotics Components using Domain-Specific Languages," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2013. [Online]. Available: http://arxiv.org/abs/1301.6022 3, 4, 10.9.5, 10.9.7, 10.9.9

[114] J. C. Baillie, "Urbi: Towards a Universal Robotic Low-level Programming Language," *International Conference on Intelligent Robots and Systems*, 2005. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1545467 7.4

[115] J.-C. Baillie, A. Demaille, Q. Hocquet, and M. Nottale, "Events! (Reactivity in urbiscript)," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2010. [Online]. Available: http://arxiv.org/abs/1010.5694 7.4, 10.9.1, 10.9.3

[116] N. Dantam and M. Stilman, "The Motion Grammar: Linguistic Perception, Planning, and Control," *Robotics: Science and Systems VII*, no. June, 2012. 7.4

[117] L. Kunze, T. Roehm, and M. Beetz, "Towards semantic robot description languages," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5589–5595. 7.5, 10.8, 10.9.2, 10.9.9, 10.9.10

[118] D. Cassou, S. Stinckwich, and P. Koch, "Using the DiaSpec Design Language and Compiler to Develop Robotics Systems," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2011. [Online]. Available: http://arxiv.org/abs/1109.2806 7.5, 10.9.8, 10.9.9

[119] A. K. Ramaswamy, B. Monsuez, and A. Tapus, "Solution space modeling for robotic systems," *Journal for Software Engineering Robotics (JOSER)*, vol. 5, no. 1, pp. 89–96, 2014. 7.5, 10.5, 10.9.3, 10.9.9, 10.9.12

[120] S. Schneider, N. Hochgeschwender, and G. Kraetzschmar, "Structured design and development of domain-specific languages in robotics," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science, D. Brugali, J. Broenink,

T. Kroeger, and B. MacDonald, Eds. Springer International Publishing, 2014, vol. 8810, pp. 231–242. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11900-7_20 7.5

[121] D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl - the planning domain definition language," Yale Center for Computational Vision and Control,, Tech. Rep. TR-98-003, 1998. 7.6

[122] E. P. D. Pednault, "ADL and the State-Transition Model of Action," *Journal of Logic and Computation*, vol. 4, pp. 467–512, 1994. 7.6

[123] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner, "Nonmonotonic causal theories," *Artif. Intell.*, vol. 153, no. 1-2, pp. 49–104, Mar. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.artint.2002.12.001 7.6

[124] F. Stulp and M. Beetz, "Combining declarative, procedural and predictive knowledge to generate and execute robot plans efficiently and robustly," *Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge)*, 2008. 7.6

[125] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 4575–4581. 7.6

[126] G. Havur, K. Haspalamutgil, C. Palaz, E. Erdem, and V. Patoglu, "A case study on the tower of hanoi challenge: Representation, reasoning and execution," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 4552–4559. 7.6

[127] G. Biggs, K. Fujiwara, and K. Anada, "Modelling and analysis of a redundant mobile robot architecture using aadl," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science, D. Brugali, J. Broenink, T. Kroeger, and B. MacDonald, Eds. Springer International Publishing, 2014, vol. 8810, pp. 146–157. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11900-7_13 7.6

[128] G. Biggs, T. Sakamoto, K. Fujiwara, and K. Anada, "Experiences with model-centred design methods and tools in safe robotics," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 3915–3922. 7.6

[129] P. H. Feiler and D. P. Gluch, *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language*, 1st ed. Addison-Wesley Professional, 2012. 7.6

[130] S. Wtzoldt, S. Neumann, F. Benke, and H. Giese, "Integrated software development for embedded robotic systems," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science, I. Noda, N. Ando, D. Brugali, and J. Kuffner, Eds. Springer Berlin Heidelberg, 2012, vol. 7628, pp. 335–348. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34327-8_31 7.6

[131] B. Hamner, S. C. Koterba, J. Shi, R. Simmons , and S. Singh, "An autonomous mobile manipulator for assembly tasks," *Autonomous Robots*, vol. 28, no. 1, pp. 131 – 149, January 2010. 7.6

[132] J. Kiener and O. von Stryk, "Cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, Oct 2007, pp. 959–964. 7.6

[133] M. Dekhil and T. Henderson, "Instrumented logical sensor systems-practice," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 4, May 1998, pp. 3103–3108 vol.4. 7.6

[134] E. Wolff, U. Topcu, and R. Murray, "Automaton-guided controller synthesis for nonlinear systems with temporal logic," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 4332–4339. 7.6

[135] M. Guo, K. Johansson, and D. Dimarogonas, "Motion and action planning under ltl specifications using navigation functions and action description language," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 240–245. 7.6

[136] J. Huckaby, S. Vassos, and H. Christensen, "Planning with a task modeling framework in manufacturing robotics," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 5787–5794. 7.6

[137] Y. Brodskiy, R. Wilterdink, S. Stramigioli, and J. Broenink, "Fault avoidance in development of robot motion-control software by modeling the computation," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science, D. Brugali, J. Broenink, T. Kroeger, and B. MacDonald, Eds. Springer International Publishing, 2014, vol. 8810, pp. 158–169. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11900-7_14 7.6

[138] H. Bruyninckx, M. Klotzbücher, N. Hochgeschwender, G. Kraetzschmar, L. Gherardi, and D. 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*, ser. SAC '13. New York, NY, USA: ACM, 2013, pp. 1758–1764. [Online]. Available: http://doi.acm.org/10.1145/2480362.2480693 7.6

[139] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000. 8

[140] K. C. Kang, M. Kim, J. Lee, B. Kim, Y. Hong, H. Lee, and S. Bang, "3d virtual prototyping of home service robots using asadal/obj," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 2903–2908. 10.1, 10.2, 10.9.1, 10.9.3

[141] G. S. Hornby, H. Lipson, and J. B. Pollack, "Evolution of generative design systems for modular physical robots," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 4146–4151. 10.1, 10.2, 10.7

[142] V. Manikonda, P. S. Krishnaprasad, and J. Hendler, "A motion description language and a hybrid architecture for motion planning with nonholonomic robots," in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 2. IEEE, 1995, pp. 2021–2028. 10.1, 10.7

[143] M. Frigerio, J. Buchli, and D. G. Caldwell, "Code generation of algebraic quantities for robot controllers," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2346–2351. 10.1, 10.3

[144] M. Wirkus, "Towards Robot-independent Manipulation Behavior Description," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2014. [Online]. Available: http://arxiv.org/abs/1412.3247 10.2, 10.4, 10.9.9

[145] I. Kitagishi, T. Machino, A. Nakayama, S. Iwaki, and M. Okudaira, "Development of motion data description language for robots based on extensible markup language-realization of better understanding and communication via networks," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2. IEEE, 2002, pp. 1145–1151. 10.2, 10.7

[146] O. Ljungkrantz, K. Akesson, J. Richardsson, and K. Andersson, "Implementing a control system framework for automatic generation of manufacturing cell controllers," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 674–679. 10.2, 10.5

[147] T. M. Roehr, F. Cordes, and F. Kirchner, "Reconfigurable integrated multirobot exploration system (rimres): heterogeneous modular reconfigurable robots for space exploration," *Journal of Field Robotics*, vol. 31, no. 1, pp. 3–34, 2014. 10.2, 10.8

[148] M. Moghadam, D. Christensen, D. Brandt, and U. Schultz, "Towards Python-based Domain-Specific Languages for Self-Reconfigurable Modular Robotics Research," in *Workshop on Domain-Specific Languages and models for Robotic systems*, San Francisco, USA, 2011. [Online]. Available: http://orbit.dtu.dk/services/downloadRegister/5830529/article.pdf 10.2, 10.9.3, 10.9.9

[149] K. Barth and D. Henrich, "A goto-based concept for intuitive robot programming," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2338–2345. 10.2, 10.7, 10.9.11

[150] S. Fleury, M. Herrb, and R. Chatila, "Design of a modular architecture for autonomous robot," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 3508–3513. 10.2, 10.5, 10.9.3, 10.9.4

[151] P. Nilas, P. Rani, and N. Sarkar, "An innovative high-level human-robot interaction for disabled persons," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 3. IEEE, 2004, pp. 2309–2314. 10.2, 10.5

[152] N. Hochgeschwender, S. Schneider, H. Voos, and G. K. Kraetzschmar, "Towards a Robot Perception Specification Language," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2013, pp. 3–6. 10.5, 10.9.9

[153] H. Nishiyama, H. Ohwada, and F. Mizoguchi, "Logic specifications for multiple robots based on a current programming language," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 1. IEEE, 1998, pp. 286–291. 10.7, 10.9.1, 10.9.3, 10.9.9

[154] J. P. Buch, J. S. Laursen, L. C. Sørensen, L.-P. Ellekilde, D. Kraft, U. P. Schultz, and H. G. Petersen, "Applying simulation and a domain-specific language for an adaptive action library," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 86–97. 10.7, 10.9.4, 10.9.11

[155] J. A. Fayman, E. Rivlin, and H. I. Christensen, "Av-shell, an environment for autonomous robotic applications using active vision," *Autonomous Robots*, vol. 6, no. 1, pp. 21–38, 1999. 10.7, 10.9.3

[156] A. R. Graves and C. Czarnecki, "Distributed generic control for multiple types of telerobot," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 3. IEEE, 1999, pp. 2209–2214. 10.7, 10.9.5, 10.9.7

[157] H. Utz, G. Kraetzschmar, G. Mayer, and G. Palm, "Hierarchical behavior organization," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 2598–2605. 10.7, 10.9.8, 10.9.9

[158] V. Raman and H. Kress-Gazit, "Explaining impossible high-level robot behaviors," *Robotics, IEEE Transactions on*, vol. 29, no. 1, pp. 94–104, 2013. 10.8

[159] F. J. Ortiz, D. Alonso, F. Rosique, F. Sánchez-Ledesma, and J. A. Pastor, "A component-based meta-model and framework in the model driven toolchain c-forge," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 340–351. 10.9.1, 10.9.3, 10.9.5

[160] S. Aggarwal, S. Mitra, and S. S. Jagdale, "Specification and automated implementation of coordination protocols in distributed controls for flexible manufacturing cells," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 2877–2882. 10.9.1, 10.9.3, 10.9.5

[161] E. Freund, M. Schluse, and J. Rossmann, "State oriented modeling as enabling technology for projective virtual reality," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2001, pp. 1842–1847. 10.9.1, 10.9.3

[162] J. Košecká, H. I. Christensen, and R. Bajcsy, "Experiments in behavior composition," *Robotics and Autonomous systems*, vol. 19, no. 3, pp. 287–298, 1997. 10.9.1, 10.9.3

[163] T. Maenpaa, A. Tikanmaki, J. Riekki, and J. Roning, "A distributed architecture for executing complex tasks with multiple robots," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 3449–3455. 10.9.1, 10.9.3

[164] S. Knoop, M. Pardowitz, and R. Dillmann, "Automatic robot programming from learned abstract task knowledge," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 1651–1657. 10.9.3, 10.9.4

[165] C. Datta, C. Jayawardena, I. H. Kuo, and B. A. MacDonald, "Robostudio: A visual programming environment for rapid authoring and customization of complex services on a personal service robot," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2352–2357. 10.9.3, 10.9.7

[166] E. Coste-Maniere and N. Turro, "The maestro language and its environment: Specification, validation and control of robotic missions," in *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2. IEEE, 1997, pp. 836–841. 10.9.3

[167] A. J. Ramirez and B. H. Cheng, "Automatic derivation of utility functions for monitoring software requirements," in *Model Driven Engineering Languages and Systems*. Springer, 2011, pp. 501–516. 10.9.3, 10.9.4, 10.9.6

[168] M. Klotzbücher, G. Biggs, and H. Bruyninckx, "Pure Coordination using the Coordinator-Configurator Pattern," in *Workshop on Domain-Specific Languages and models for Robotic systems*, vol. 231940, 2012. 10.9.3, 10.9.4

[169] D. Alonso, C. Vicente-Chicote, F. Ortiz, J. Pastor, and B. Alvarez, "V3cmm: A 3-view component meta-model for model-driven robotic software development," *Journal of Software Engineering for Robotics*, vol. 1, no. 1, pp. 3–17, 2010. 10.9.3, 10.9.5, 10.9.9

[170] M. Bordignon, K. Stoy, and U. P. Schultz, "A virtual machine-based approach for fast and flexible reprogramming of modular robots," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 4273–4280. 10.9.3, 10.9.5, 10.9.9

[171] B. Bouzouia, F. Guerroumi, and A. Boukhezar, "A three-layer workcell control architecture design," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 1185–1191. 10.9.3, 10.9.7, 10.9.9

[172] A. Paikan, G. Metta, and L. Natale, "A Representation of Robotic Behaviors using Component Port Arbitration," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2014. [Online]. Available: http://arxiv.org/abs/1412.4847 10.9.3, 10.9.4, 10.9.8

[173] B. Schwartz, L. Nägele, A. Angerer, and B. A. MacDonald, "Towards a Graphical Language for Quadrotor Missions," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2014. [Online]. Available: http://arxiv.org/abs/1412.1961 10.9.3, 10.9.7

[174] X. Dai, G. Hager, and J. Peterson, "Specifying behavior in c++," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1. IEEE, 2002, pp. 153–160. 10.9.3, 10.9.8

[175] E. Gat, "Alfa: A language for programming reactive robotic control systems," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1116–1121. 10.9.3, 10.9.8

[176] K. Kułakowski and T. Szmuc, "Modeling robot behavior with ccl," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2012, pp. 40–51. 10.9.3

[177] Y. Sun, J. Gray, K. Bulheller, and N. von Baillou, *A model-driven approach to support engineering changes in industrial robotics software*. Springer, 2012. 10.9.3, 10.9.7, 10.9.9

[178] H. C. Woithe and U. Kremer, "A programming architecture for smart autonomous underwater vehicles," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 4433–4438. 10.9.3

[179] M. Anderson, C. Crawford, and M. Stanforth, "Enabling Robot Device Discovery Through Robot Device Descriptions," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2011. 10.9.5, 10.9.9

[180] S. Chaki and J. Edmondson, "Model-driven verifying compilation of synchronous distributed applications," in *Model-Driven Engineering Languages and Systems*. Springer, 2014, pp. 201–217. 10.9.5, 10.9.6

[181] W. Hongxing, D. Xinming, L. Shiyi, T. Guofeng, and W. Tianmiao, "A component based design framework for robot software architecture," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 3429–3434. 10.9.5, 10.9.9

[182] S. Adam, M. Larsen, K. Jensen, and U. P. Schultz, "Towards rule-based dynamic safety monitoring for mobile robots," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 207–218. 10.9.6

[183] V. Braberman, N. D'Ippolito, N. Piterman, D. Sykes, and S. Uchitel, "Controller synthesis: From modelling to enactment," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 1347–1350. 10.9.6, 10.9.8

[184] B. Dittes and C. Goerick, "Intelligent system architectures-comparison by translation," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 1015–1021. 10.9.8, 10.9.9, 10.9.10

[185] ——, "A language for formal design of embedded intelligence research systems," *Robotics and Autonomous Systems*, vol. 59, no. 3, pp. 181–193, 2011. 10.9.8, 10.9.10

[186] I. Pembeci and G. Hager, "Functional reactive programming as a hybrid system framework," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 727–734. 10.9.8, 10.9.9

[187] T. Buchmann, J. Baumgartl, D. Henrich, and B. Westfechtel, "Towards A Domain-Specific Language For Pick-And-Place Applications," in *Workshop on Domain-Specific Languages and models for Robotic systems*, 2014. [Online]. Available: http://arxiv.org/abs/1401.1376 10.9.12

Arne Nordmann Arne Nordmann received his diploma in electrical engineering at TU Dortmund University, Germany, in 2009. Afterwards he joined the *Cognitive Systems Engineering* group at the Bielefeld Institute for Cognition and Robotics (CoR-Lab) as Ph.D. student. In 2015 he joined the corporate research department at Robert Bosch GmbH in Stuttgart as research engineer. His focus of research resides on model-driven engineering methods and domain-specific languages in the context of robotics systems and advanced driver assistance.

Nico Hochgeschwender received his diploma in computer science at University of Applied Sciences Ravensburg-Weingarten, Germany, in 2007. Afterwards he joined ESG GmbH, Munich, Germany as a system engineer developing avionics software for Unmanned Aerial Vehicles. In 2009 he joined Bonn-Rhein-Sieg University, Sankt Augustin, Germany as a research scientist in the EU-funded project BRICS (Best Practice in Robotics), researching model-driven engineering methods for robots. Since 2013 he has also been affiliated with the University of Luxembourg, Luxembourg as a Ph.D. student investigating methods, models and tools that enable the design and development of adaptive robot perception architectures.

Dennis Leroy Wigand received his M. Sc. degrees in computer science from Bielefeld University, Germany, in 2015. Afterwards he joined the *Cognitive Systems Engineering* group at the Bielefeld Institute for Cognition and Robotics (CoR-Lab) as Ph.D. student. Dennis Wigand's particular research interest lies in domain-specific system engineering with respect to code generation. Due to his participation in the EU project *CogIMon (Horizon 2020)*, which aims at a step-change in compliant human-robot interaction, he is particularly focused on the force domain.

Dr. Sebastian Wrede received his PhD (Dr.-Ing.) in Computer Science from Bielefeld University in 2008. Since 2009 he heads the Cognitive Systems Engineering group (at CoR-Lab) and is responsible investigator in the Excellence Cluster on Cognitive Interaction Technology (CITEC) at Bielefeld University. Furthermore, he is coordinator of the innovation project *FlexiMon* within the it's OWL leading-edge cluster on reconfigurable robotics systems in manufacturing and responsible investigator in the EU project *CogIMon (Horizon 2020)*. Sebastian Wrede's focus of research resides on model-driven engineering methods, domain-specific languages and software architectures for interactive robotics applications. He is a member of GI and IEEE RAS TC-SOFT.