Modularization of skill ontologies for industrial robots

Jacobsson, Ludwig; Malec, Jacek; Nilsson, Klas

Published in:
47th International Symposium on Robotics, ISR 2016

2016

Document Version:
Early version, also known as pre-print

Modularization of skill ontologies for industrial robots

Ludwig Jacobsson, Department of Computer Science, Lund University, Sweden, ludjac@gmail.com
Jacek Malec, Department of Computer Science, Lund University, Sweden, jacek.malec@cs.lth.se
Klas Nilsson, Department of Computer Science, Lund University, Sweden, klas.nilsson@cs.lth.se

Abstract

With industrial robots ready to take the next step in mastering manufacturing tasks new approaches to reduce the programming effort are needed. This is achieved by introducing skills as robot "know-how" and using them as a higher abstraction level of robot instructions during programming. The skills are reusable items providing motion control and rich declarative descriptions of complex robot capabilities. Storing the skills requires an adequate knowledge representation model that enables reuse and reasoning on skills and simplifies knowledge management. In this paper we report on development of a skill representation model and its implementation in a knowledge base. The developed model is effectively a class hierarchy of the skill concepts implemented in a modularized ontology structure. The resulting model clarifies the intrinsic concepts of a skill and presents a module structure that enables the future development and reuse of skills in general.

1 Introduction

Cognitive robotics is a fast-growing domain dealing with creating robot systems aware of their own capacities and limitations. Web-enabled intelligent robots can communicate with external knowledge bases to seek information or to exchange programs and data with each other. This is to some extent already a reality for research service robots, where projects like RoboEarth [23] (roboearth.org), KnowRob [20] (knowrob.org) or RoboHow [2] (robohow.eu) provide infrastructure for exchanging pieces of relevant knowledge among infrastructure-aware robots. In particular, in order to be able to understand each other's contributions, the robots need to have same understanding of the data and knowledge available in the common repositories. This common understanding is usually introduced via appropriate *ontologies* [6], describing the concepts in a machine-readable way.

In order to understand its own limitations, a cognitive robot must also possess the ability to reason about itself and its environment, about possible actions it can take or reactions it must offer given some circumstances. It must also understand *why* some actions are taken and how. This leads to the concept of a robot skill, encompassing physical activities, sensory interaction, purposefulness and business value, all in the context of a particular application domain. We have earlier argued for this kind of understanding of this concept [16]. Our point is that robots need not only possess skills and exchange them among themselves; they need also understanding of what a skill is and how it should be used, maintained, shared (or kept secret) and error-recovered from. They should distinguish skills ready to be run from those not ready for execution, misconfigured, failed during execution, etc.

This paper presents an ontology of skills prepared with industrial robots in mind. Its main result is actually a set of ontologies collectively describing industrial robot skills, intended to be used by cognitive robotic systems for reasoning about their activities, both physical as well as mental, like learning, teaching and interacting with other, robotic or human, agents. The particular contribution of this paper is the meta-ontology of skills, introducing concepts like configuration, coordination and orchestration of skills, while previous work [3, 7, 13, 18] has provided the necessary building blocks, i.e., the ontologies of basic robotic skills and devices (rosetta.owl), of behaviour specification (sfc.owl), of skill parameterization (params.owl). Of particular interest is the work presented in [19] where the philosophy behind the skill transfer, reparameterization and reuse by other hardware configurations is presented in detail. This paper, together with the ontology it describes, is a complement of those.

2 Robotic Skills

A *skill* of a robot is an overloaded term, used very often informally and in a slightly different meaning depending on the context. Quite often by a skill one understands a *potential* to perform a particular *action* leading to a *meaningful* outcome. Although this is a good first approximation, there are many facets of a robotic skill that need to be taken into account before we let robots think by themselves about their own activities. Therefore the description below tries to characterize the different aspects of the skill concept, and make them sufficiently concrete and semantically clear so that an automated reasoner can exploit them appropriately.

The formal skill definition presented in this section follows rather closely the verbal one introduced in [16].

Moreover, we provide here just a couple of basic definitions, while the complete formalization is available at the Lund skill ontology site `http://git.cs.lth.se/ontologies/`.
The typographical conventions in the text below are the following: $Classes$ (capitalized: concepts), `Roles` (aka relations), $individuals$ (lowercase). We use description logic for the formal definitions.

2.1 Action

One of the basic terms necessary for defining a skill is **action**. We distinguish $SimpleAction$s, $CompositeAction$s and $AtomicAction$s. The three concepts are defined as disjoint.

$$Action \doteq SimpleAction \sqcup CompositeAction \sqcup \sqcup AtomicAction.$$

$$SimpleAction \sqcap CompositeAction = \bot.$$

$$SimpleAction \sqcap AtomicAction = \bot.$$

$$CompositeAction \sqcap AtomicAction = \bot.$$

Actions have **effect**. We introduce `hasEffect` and `isEffectOf` relations for this. (This is a naming clash with OWL-S ontology, which in its turn introduces `hasEffect` to hold between a $Result$ and an $Expression$. The two ontologies are obviously distinguished by their prefixes: `http://kif.cs.lth.se/ontologies/skills.owl\#` and `http://www.daml.org/services/owl-s/1.2/Process.owl\#`, respectively.)

$$\top \sqsubseteq \forall \texttt{hasEffect}.Effect.$$

This says that the range of `hasEffect` relation is concept *Effect*.

$$\exists \texttt{hasEffect}.\top \sqsubseteq Action.$$

This says that the domain of `hasEffect` relation is the concept $Action$.

$$\texttt{hasEffect} = \texttt{isEffectOf}^{-}.$$

This says that *roles* (relations) `hasEffect` and `isEffectOf` are inverse of each other. In what follows below we will skip this kind of explanatory texts, unless they are deemed to be necessary or useful.

Motions are actions with **physical effects**. Only motions can have physical effects. A physical effect is caused by an *Actuator*. Relation pair `causes` and `isCausedBy` are used to express that.

$$Motion \doteq Action \sqcap (\forall \texttt{hasEffect}.PhysicalEffect).$$

$$\forall \texttt{hasEffect}.PhysicalEffect \sqsubseteq Motion.$$

$$\exists \texttt{causes}.\top \sqsubseteq Actuator.$$

$$\top \sqsubseteq \forall \texttt{isCausedBy}.Actuator.$$

$$\exists \texttt{isCausedBy}.\top \sqsubseteq PhysicalEffect.$$

Motions can be explicit or implicit. An *explicit* motion is somewhat equivalent to "imperative", meant to mean "executable". *Implicit* is equivalent to "declarative" and is meant to be *synthesized* later on. Implicit motions are useful for verification purposes. One possible question a motion can ask the system: "Am I executable?" An implicit motion together with an appropriate solver can yield and explicit (executable) motion. Compositionality is easier to achieve for implicit motions.
AtomicActions and SimpleActions can be expressed as OWL-S primitives. OWL-S is a formal language for characterizing composable (software) services [14]. In robotic context, we say that an action `isModeledBy` an OWL-S *Process*.
Actions have an OWL-S specification as services. This dependence is captured by the relation `isDescribedBy`.

$$\top \sqsubseteq \forall \texttt{isDescribedBy}.Service.$$

This says that the range of `isDecsribedBy` relation is *Service* class (concept).

$$\exists \texttt{isDescribedBy}.\top \sqsubseteq Action.$$

This says that the domain of `isDescribedBy` relation is the concept $Action$.

2.2 Configuration

As the skill concept is intimately related to *orchestration* (of motions), consisting in turn of *configuration* and *coordination* [16], we introduce those three concepts first.

In order to describe a *configuration* (of a system) one needs to specify the *topology* (i.e., the subsystems involved and their relations), the *connectivity* (i.e., the status of connections between the subsystems, including the physical connections as well as logical ones) and the *parameterization* (providing the actual settings for the configuration).
This dependency is captured by the following three roles (object properties):

$$\exists \texttt{involvesTopology}.\top \sqsubseteq Configuration.$$

$$\top \sqsubseteq \forall \texttt{involvesTopology}.ConfigurationTopology.$$

$$\exists \texttt{involvesConnectivity}.\top \sqsubseteq Configuration.$$

$$\top \sqsubseteq \forall \texttt{involvesConnectivity}.ConfigurationConnectivity.$$

$$\exists \texttt{involvesParameterization}.\top \sqsubseteq Configuration.$$

$$\top \sqsubseteq \forall \texttt{involvesParameterization}.ConfigurationParameterization.$$

A *Configuration* is characterized by its *Status*:

$$\exists \texttt{isCharacterizedBy}.\top \sqsubseteq Configuration.$$

$$\top \sqsubseteq \forall \texttt{isCharacterizedBy}.Status.$$

$$\texttt{characterizes} = \texttt{isCharacterizedBy}^{-}.$$

A status can be expressed in several ways, e.g., using the traffic light symbol (in order to capture three, sometimes four, distinguished values of the status symbolized by red, yellow, /possibly blinking yellow/ and green, respectively), using the binary {OK, NoOK} domain or using some numeric encoding, like a digit between 0 and 9. Of course, there are many more possibilities,
We have introduced the *Configuration* concept in a generic manner, as applied to any kind of system. In the context of industrial robotic systems we will be interested in using it to characterize at least three kind of objects: *Device*, *Connection* and *Coordination*.
In each of those cases the *Status* will mean something different, but the meaning will be sufficiently clear to reason further on its base. For example, a status of a (physical) connection between two devices can be: non-connected; connected, but not configured; connected and properly configured; — mapping quite well to the traffic light visualization by red, yellow and green.

$$\exists \texttt{hasStatusValue}.\top \sqsubseteq \mathit{Status}.$$

$$\mathit{TrafficLightStatus} \sqsubseteq \mathit{Status}.$$

The *TrafficLightStatus* consists of three independent statuses, namely RedStatus, YellowStatus and GreenStatus:

$$\mathit{RedStatus} \sqsubseteq \mathit{TrafficLightStatus}.$$

$$\mathit{YellowStatus} \sqsubseteq \mathit{TrafficLightStatus}.$$

$$\mathit{GreenStatus} \sqsubseteq \mathit{TrafficLightStatus}.$$

Each of them is restricted to have just one (string) value, out of three possible ones: *On*, *Off* and *Blinking*. For this purpose there exist three data properties: `hasGreenLightState`, `hasYellowLightState`, and `hasRedLightState`, all three subproperties of `hasLightState` that has one of three possible values: "Blinking", "Off", or "On". The semantics of their combinations is dependent on the application domain.

2.3 Coordination

Coordination describes the (possibly real-time) behaviour of a correctly configured system. The means for this description normally involve some kind of *transition system*. There exist may variants of those, with different denotational and operational semantics, however, having in common the capability of specifying causal (temporal or event-based) dependencies between simpler behaviours, out of which the overall behaviour is composed. So, in order to describe coordination, one has to be able to specify simple (sometimes primitive) behaviours in some way, and then their interdependencies, like serial or parallel connections, mutual exclusion, synchronization, etc.

$$\exists \texttt{defines}.\top \sqsubseteq \mathit{Coordination}.$$

$$\top \sqsubseteq \forall \texttt{defines}.\mathit{Behaviour}.$$

We distinguish therefore *behaviours* and *behaviour descriptions*. Behaviours are normally (in robotics) associated with (robot) actions, but may also be used to denote computations, environment changes, etc. Any observable change of some physical value in time may be represented as a behaviour of an appropriate entity. Therefore the transition systems ontology lists the following concepts as subconcepts to *Behaviour*: Action, Activity, (Condition) Occurrence, Event, Execution, Process, Task. Depending on the particular domain described, each of those is somehow captured by the term behaviour. *Behaviour description* in its turn is a tool for describing a behaviour. We associate always a transition system with a behaviour. In the simplest case of a primitive action (like move), the transition system will be trivial and will consist of only one state: moving, and one outgoing transition from it, denoting completion. The continuous activity in each state will be described using a formalism appropriate for the domain in question and the tools available: differential equations, difference equations, data flow diagram, program code in some language. The resulting hybrid system is intended to capture a large part of what we intuitively describe as "behaviours" of technical systems.

$$\exists \texttt{describes}.\top \sqsubseteq \mathit{BehaviourDescription}.$$

$$\top \sqsubseteq \forall \texttt{describes}.\mathit{Behaviour}.$$

$$\exists \texttt{isDescribedBy}.\top \sqsubseteq \mathit{Behaviour}.$$

$$\top \sqsubseteq \forall \texttt{isDescribedBy}.\mathit{BehaviourDescription}.$$

$$\texttt{isDescribedBy} \equiv \texttt{describes}^{-}.$$

The ontology we have developed so far (see Sec. 3 below) names at least five such transition system formalisms: OpenPLC, Statecharts [10], Intermediate Modeling Language (IML, originating in AutomationML [5]) and Sequential Function Charts (SFC) [21], besides the generic Finite State Machines (FSM). Each is a subconcept of a *Graph*, where *Nodes* and *Arcs*, appropriately decorated, correspond to states and transitions of a transition system. Besides FSMs, graphs are also used to express (specify) assembly tasks (*Assembly Graphs*) and assembly constraints (*Constraint Graphs*).

2.4 Orchestration

Orchestration consists of Configuration and Coordination. (It is effectively a pair.)

$$\exists \texttt{hasConfiguration}.\top \sqsubseteq \mathit{Orchestration}.$$

$$\top \sqsubseteq \forall \texttt{hasConfiguration}.\mathit{Configuration}.$$

$$\exists \texttt{hasCoordination}.\top \sqsubseteq \mathit{Orchestration}.$$

$$\top \sqsubseteq \forall \texttt{hasCoordination}.\mathit{Coordination}.$$

An orchestrated skill is ready to be executed. Note that skills **MUST** be defined declaratively, otherwise they are not skills, but *incomplete skills* or *skill candidates*.

3 Skill Ontologies

The specific ontology this paper introduces on top of existing ones, `skills.owl`, is a meta-ontology in the sense that it assumes other, concrete robot skill ontologies to provide details about robotic devices, their capabilities and the interesting operations demanded by the domain. For example, in assembly there will be joining operations of various kind, like glueing, welding, riveting or screwing together, while robotic capabilities described in the bottom-level ontology will include pressing (i.e. applying a force in a given direction), force-controlled snapping, screwing a nut, etc. The meta-ontology is concerned in turn with providing the necessary semantic grounds for reasoning about a particular robotic cell setup: Is it capable of performing task X?, Will it perform the task given current configuration and parameterization? How should the coordination be performed? What behaviours will the devices involved realize?

This meta-ontology is created in order to capture, among other concepts, the 5C meta-model [22] realizing separation of concerns (Communication, Computation, Coordination, Configuration, and Composition) while describing (industrial) robot activities. The formalization closely follows the model-driven approach to software development in robotics [4], in order to guarantee that the software controlling robots is correct, verifiable, robust and modular.

The dependencies between the robotic ontologies mentioned in the introduction are illustrated in Figure 1. The `skills.owl` is the ontology introduced here, while the others are described in detail in [18]. Please note also that all the open ontologies, denoted in the figure by gray shade and license name, are available, either at the knowledge server `kif.cs.lth.se` at URIs `http://kif.cs.lth.se/ontologies/XXXX.owl`, where `XXXX` is the name of the corresponding ontology, or at the Lund skill ontology site mentioned earlier, except the two bottom ones, external to our system and available from their respective providers. We use the Creative Commons 3.0 license for the open ontologies mentioned here.

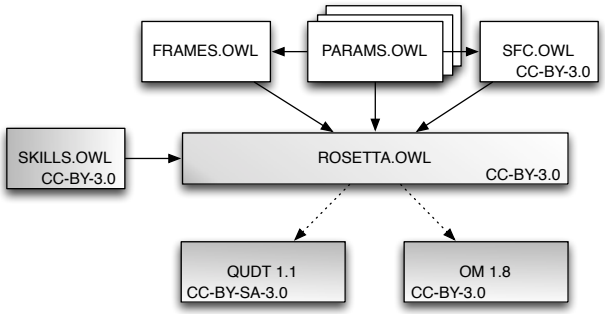

Figure 1: The original ontology structure, [18].

4 Modularization

As described in Sec. refsec:skill-ontologies, the set of robotic ontologies developed in the ROSETTA and subsequent EU projects have the structure illustrated in Fig. 1. In particular, the meta-ontology `skills.owl` is separated from the ground concept ontologies, thus making them less amenable for introspection by software systems. Moreover, although the division into *frame*-related concepts (`frames.owl`, describing poses and geometrical relations), *behaviour* encoding (`sfc.owl`, formalizing transition systems of various kinds) and skill *parameters* (`params.owl`, ensuring correct execution of the underlying software) corresponds to the conceptual model we used for a very long time, their dependencies are unclear and very hard to maintain.

Therefore, in order to make the structure more modular, allowing one to extend the set of available skills with new ones, and to offer possibility of ontology plug-ins with proprietary information specific for some vendors or end-users (like e.g., proprietary descriptions of robot skills), we have rearranged the concepts defined in `skills.owl` [11] by making explicit the aspects of skill *configuration* (dependencies, parametrization, task and feature frames associated with it, etc.) and `coordination`, encompassing the system behaviour in an explicit manner. The new structure is illustrated in Fig. 2.

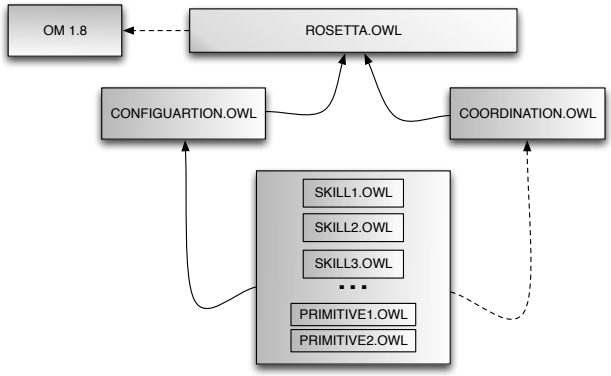

Figure 2: The new ontology structure, [11].

In this manner the important aspects of relevance to cognitive processes, i.e. coordination and configuration of skills, became explicitly available. The structure of a single robotic skill is shown in Fig. 3, while Fig. 4 presents the new structure of an example skill named Skill1,with all the necessary annotations. The reasoners exploiting this new structure are currently tested in practical settings of an SMERobotics demonstrator [8]. The ontologies have also been used in a knowledge repository for a prototype extension of the ABB RobotStudio software.

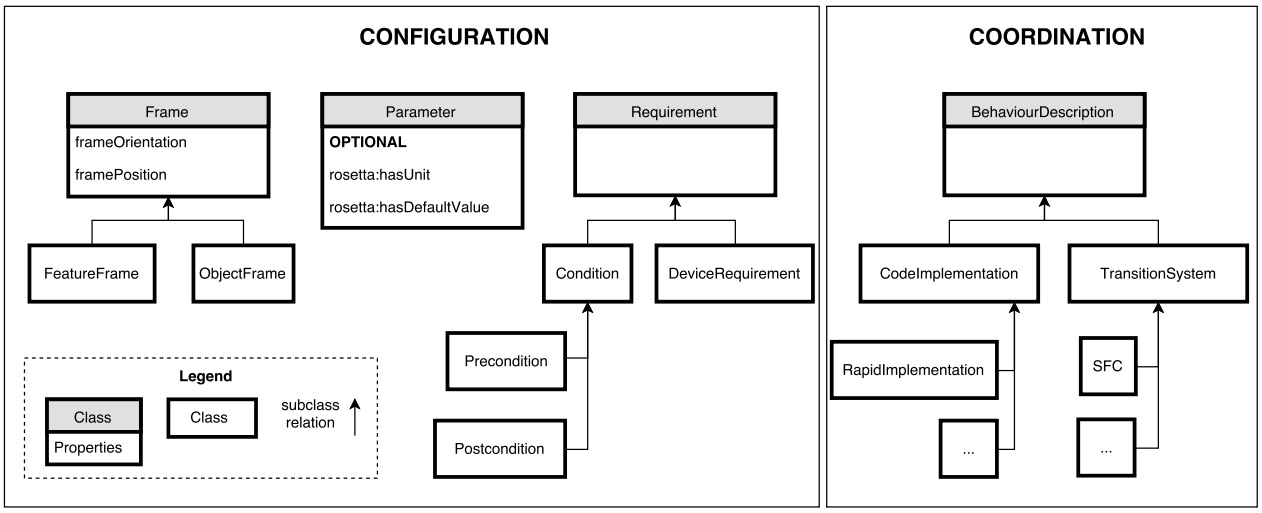

Figure 3: The new skill structure, [11].

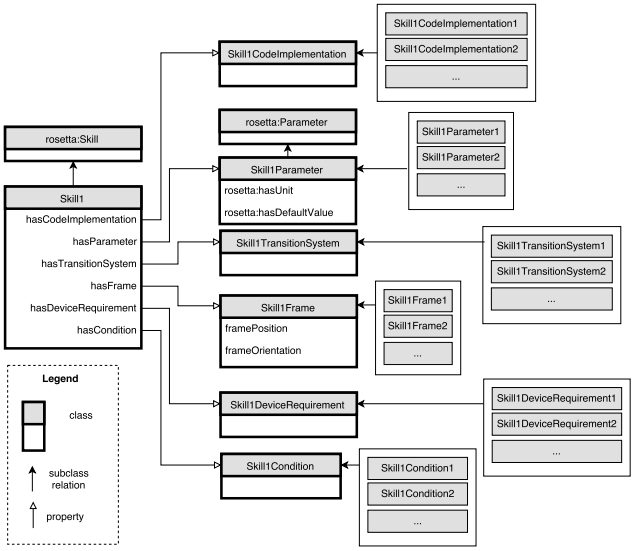

Figure 4: An example Skill1 represented according to the ontology, [11].

5 Related work

Cognitive robotics demands explicit reasoning about robot capabilities. Ontologies ease this task by making this knowledge explicit and transferable among agents. There is a number of projects developing this kind of robotic knowledge bases, like the mentioned already KnowRob [20], RoboEarth [23] or Open Robotics Ontology ORO [12], but they usually refer to the service robotics domain and involve mobility as the basic robot activity. A good recent overview of this research may be found in [9]. In case of industrial robotics the amount of work is less extensive, but there exist initiatives to semantically characterize basic concepts, like the Core Ontology for Robotics and Automation, CORA, [17], based in its turn on the Suggested Upper Merged Ontology SUMO. however, the concepts introduced there are rather basic, and do not reach the complexity necessary to characterize industrial robotic skills. There are some interesting and valuable attempts to remedy this problem, but they are yet limited in their scope, like [1], [15] or [24].

6 Conclusions

The main purpose of this paper is to announce the existence of a set of modular ontologies intended to characterize industrial robotic skills, focusing on robotized assembly in particular. The ontologies are under continuous development and may still miss some important aspects, but are already used in a meaningful way in our research projects (including the ones mentioned below in acknowledgments). We particularly stress the skill portability as one of the necessary aspects and most important properties. We expect the ontologies to be useful also outside the context of our projects and do hope to get feedback from prospective adopters.

7 Acknowledgment

The research leading to these results has received partial funding from the European Union's seventh framework program (FP7/2007-2013) under grant agreements No. 230902 (project ROSETTA), No. 285380 (project PRACE), No. 287787 (project SMErobotics), and from the H2020 program under grant agreement No. 644938 (project SARAFun). The authors are grateful to Maj Stenmark for relentless experimentation with continuously varying ontologies.

References

[1] S. Balakirsky, Z. Kootbally, C. Schlenoff, T. Kramer, and S. Gupta. An industrial robotic knowledge representation for kit building applications. In *Proceedings IEEE/RSJ International*

Conference on Intelligent Robots and Systems (IROS), 2012.

[2] Michael Beetz, Moritz Tenorth, and Jan Winkler. Open-EASE – a knowledge processing service for robots and robotics/ai researchers. In *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, USA, 2015.

[3] Anders Björkelund, Lisett Edström, Mathias Haage, Jacek Malec, Klas Nilsson, Pierre Nugues, Sven Gestegård Robertz, Denis Störkle, Anders Blomdell, Rolf Johansson, Magnus Linderoth, Anders Nilsson, Anders Robertsson, Andreas Stolt, and Herman Bruyninckx. On the integration of skilled robot motions for productivity in manufacturing. In *Proc. IEEE International Symposium on Assembly and Manufacturing*, Tampere, Finland, 2011.

[4] Davide Brugali. Model-driven software engineering in robotics. *IEEE Robotics and Automation Magazine*, 22(3):155–166, September 2015.

[5] Rainer Drath, editor. *Datenaustausch in der Anlagenplanung mit AutomationML. Integration von CAEX, PLCopen, XML und COLLADA*. Springer, 2010.

[6] Tom R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43:907–928, 1995.

[7] M. Haage, J. Malec, A. Nilsson, K. Nilsson, and S. Nowaczyk. Declarative-knowledge-based reconfiguration of automation systems using a blackboard architecture. In A. Kofod-Petersen, F. Heintz, and H. Langseth, editors, *Proc. 11th Scandinavian Conference on Artificial Intelligence*, pages 163–172. IOS Press, 2011.

[8] Mathias Haage, Stefan Profanter, Ingmar Kessler, Alexander Perzylo, Nikhil Somani, Olof Sörnmo, Martin Karlsson, Sven Gestegård Robertz, Klas Nilsson, Ludovic Resch, and Michael Marti. On cognitive robot woodworking in SMErobotics. In *Proc. ISR 2016*, 2016.

[9] Tamas Haidegger et al. Applied ontologies and standards for service robots. *Robotics and Autonomous Systems*, 61:1215–1223, 2013.

[10] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[11] Ludwig Jacobsson. A module-based skill ontology for industrial robots. Master's thesis, Department of Computer Science, Faculty of Engineering LTH, Lund University, 2015.

[12] S. Lemaignan, R. Ros, L. Moösenlechner, R. Alami, and M. Beetz. ORO, a knowledge management platform for cognitive architectures in robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 3548–3553, 2010.

[13] Jacek Malec, Klas Nilsson, and Herman Bruyninckx. Describing assembly tasks in declarative way. In *Proc. IEEE ICRA 2013 Workshop on Semantics, Identification and Control of Robot-Human-Environment Interaction*, 2013.

[14] David Martin et al. OWL-S 1.2 release. http://www.ai.sri.com/daml/services/owl-s/1.2/, accessed 30.06.2014, 2004.

[15] A. Nilsson, R.o Muradore, K. Nilsson, and P. Fiorini. Ontology for robotics: a roadmap. In *Proceedings of Int. Conf. Advanced Robotics (ICAR09)*, Munich, Germany, 2009.

[16] Klas Nilsson, Elin Anna Topp, Jacek Malec, and Il-Hong Suh. Enabling reuse of robot tasks and capabilities by business-related skills grounded in natural language. In *Proc of ICAS2013*, Lisbon, Portugal, March 2013.

[17] Edson Prestes et al. Towards a core ontology for robotics and automation. *Robotics and Autonomous Systems*, 61:1193–1204, 2013.

[18] Maj Stenmark and Jacek Malec. Knowledge-based instruction of manipulation tasks for industrial robotics. *Robotics and Computer Integrated Manufacturing*, 2014.

[19] Maj Stenmark, Jacek Malec, Klas Nilsson, and Anders Robertsson. On distributed knowledge bases for robotized small-batch assembly. *IEEE Trans. Automation Science and Engineering*, 2015.

[20] M. Tenorth and M. Beetz. Knowrob - a knowledge processing infrastructure for cognition-enabled robots. part 1: The knowrob system. *International Journal of Robotics Research*, 32(5):566–590, 2013.

[21] Alfred Theorin. *A Sequential Control Language for Industrial Automation*. PhD thesis, Department of Automatic Control, Lund University, Sweden, November 2014.

[22] Dominick Vanthienen, Markus Klotzbuecher, and Herman Bruyninckx. The 5C-based architectural composition pattern. *Journal of Software Engineering in Robotics*, 5(1):17–35, 2014.

[23] Markus Waibel, Michael Beetz, Javier Civera, Raffaello d'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Häussermann, Rob Janssen, J.M.M. Montiel, Alexander Perzylo, Bjoern Schiessle, Moritz Tenorth, Olivier Zweigle, and M.J.G. (René) Van de Molengraft. Roboearth. *IEEE Robotics and Automation Magazine*, 18(2):69–82, June 2011.

[24] Cezary Zieliński and Tomasz Kornuta. An object-oriented robot ontology. In *Proc. IEEE Intelligent Systems*, volume 323 of *AISC*. Springer, 2014.