Describing assembly tasks in declarative way* Extended Abstract

Jacek Malec[1] Klas Nilsson[1] Herman Bruyninckx[2]

Abstract— **Task specification in a robotized assembly context is a complex process leading from the verbalized goal of the assembly via the subsequent abstractions, each on lower level of detail, down to the robotized cell program realizing the posed goal. This process is error-prone on one hand and knowledge-intensive on the other, giving occupation to a number of specialists but precluding many small and medium enterprises from robotizing their production.**

This paper presents an approach to automatize parts of this process, offering a set of knowledge-based tools and techniques to simplify the task specification and guide the user on the way to the final robot program.

I. Introduction

There is a multitude of task description formalisms that are adapted to their intended domain of application. In case of discrete processes one usually employs some variant of state transition systems to describe the consecutive steps of the production, however, abstracting away the continuous aspects of the process. In case of purely continuously-describable phenomena some equational form, be it algebraic or differential equation systems, is usually a typical language. However, this kind of description usually abstracts away the control aspects, error handling, initialization and calibration, etc. One common way of merging the two kinds of descriptions and models are hybrid automata, allowing one to speak both about the discrete-event aspects of a process, as well as its continuous phenomena occurring between the discrete transitions among modes of operation.

Irrespectively of the modeling language used and the level of abstraction assumed, the way to the robot control program is still very long. In order to properly specify an assembly task (our domain of interest, although we expect our approach to be naturally extensible to other domains of robotized production as well) one needs to specify all involved workpieces, the geometrical environment in which the production takes place, the involved hardware and software, their interconnections and dependencies, and many other details relevant for a particular task. For each aspect of this specification either a separate language and modeling methodology needs to be used, with the unavoidable problems on the language borders, or a closed tool, usually provided by a concrete robot manufacturer and supporting only specific brand of hardware, may help in overcoming some of the barriers, but severely limiting portability and extendability.

In this paper we present an attempt to lower some of the barriers mentioned above. Our approach is based on declarative, compositional representations that extend the model-driven engineering (MDE) towards what we would like to call knowledge-driven engineering (KDE) in the rest of this text. We adopt the semantic web methodology to express knowledge about manufacturing in a machine-understandable manner. The main contribution of the reported work is a methodology combining the complete workflow, beginning with task specification and ending in an executable robot program, in a clear, declarative and compositional way.

The rest of this paper is divided as follows: first we present a number of related works focusing on task representation formalisms. Then we describe our domain of interest, i.e. the robotized assembly. The next section introduces the ontologies we developed for declarative knowledge representation about manufacturing. The following section presents the actual domain-specific languages used at various level of modeling abstraction. Finally we present the results obtained so far. We conclude with some suggestions for further work.

II. Related Work

Task representation has been an important area for the domain of robotics, in particular for autonomous robots research. The very first approaches were based on logic as a universal language for representation. A good overview of the early work can be found in [5]. The first autonomous robot, Shakey, exploited this approach to the extreme: its planning system STRIPS, its plan execution and monitoring system PLANEX and its learning component (Triangle tables) were all based on first order logic and deduction [1]. This way of thought continued, leading to such efforts as "Naive physics" by Patrick Hayes (see [5]) or "Physics for Robots" [14]. This development stopped because of the insufficient computing power available at that time, but has recently received much attention in the wider context of semantic web. The planning techniques have also advanced much and may be used nowadays for cases of substantial complexity [7], although generic automation problems are usually still beyond this limit.

Later, mixed architectures begun to emerge, with a reasoning layer on the top, reactive layer in the bottom, and some synchronisation mechanism, realized in various disguises, in the middle. This approach to building autonomous robots is

*The research leading to these results has received partial funding from the European Union's seventh framework program (FP7/2007-2013) under grant agreement No. 230902 (project ROSETTA) and No. 285380 (project PRACE).

[1]J. Malec and K. Nilsson are with Department of Computer Science, Lund University, Sweden, jacek.malec@cs.lth.se, klas@cs.lth.se.

[2]H. Bruyninckx is with Department of Mechanical Engineering, KU Leuven, Belgium, Herman.Bruyninckx@mech.kuleuven.be

prevalent nowadays [2], where researchers try to find an appropriate interface between abstract, declarative description needed for any kind of reasoning and procedural one needed for control. The problem remains open until today, only its complexity (or the complexity of solutions) grows with time and available computing power.

Task description in industrial robotics setting comes also in form of hierarchical representation and control, but the languages used are much more limited (and thus more amenable to effective implementation). There exist a number of standardized approaches, based e.g. on IEC 61131 standards [11] devised for programmable logic controllers, or proprietary solutions provided by robot manufacturers, however, to a large extent the solutions are incompatible with each other. EU projects like RoSta[1] are attempts to change this situation.

At the theory level all the approaches combining continuous and discrete formalisms may be considered variants or extensions of hybrid systems [8], possibly hierarchical. Hybrid control architectures allow to some extent separation of concerns, where the continuous and real-time phenomena are handled in their part of the system, while the discrete aspects are treated by appropriate discrete tools. Our earlier work attempted at declaratively specifying such hybrid systems, but was limited to knowledge-based configuration [9].

Task descriptions come in different disguises, depending on the context, application domain, level of abstraction considered, tools available, etc. Usually tasks are composed out of skills, understood as capabilities of available devices [4], but the way of finding appropriate composition varies heavily, from manual sequencing in many workflows, via AI-influenced task planning [7], hybrid automata development tools [8], Statecharts [10] and Sequential Function Charts (SFCs) [11], iTaSC specifications [6], to development of monolithic programs in concrete robot programming languages, like e.g. RAPID [13].

III. Assembly Tasks

Our domain of interest is assembly, i.e. such manipulation of objects (workpieces) that creates compound objects from simpler ones. An example of such task is presented in Fig. 1 in the form of an assembly graph specifying a number of assembly operations (AOs).

Normally, a number of skills (unit task realisations) are used to perform such complex assembly, beginning with raw workpieces, proceeding via assembly operations (primitives for the above representation) into a complete product. On the topmost level we deal with discrete assembly operations. However, each one requires a complex skill (consisting of movement and possibly an associated perception action) to complete its objective. This in turn requires usually a complex program, normally expressed as set of states with properly guarded transitions, where each state corresponds to a continuous transformation of the scene (movement). For our example, the AO:4 from Fig. 1 consists of a snapping

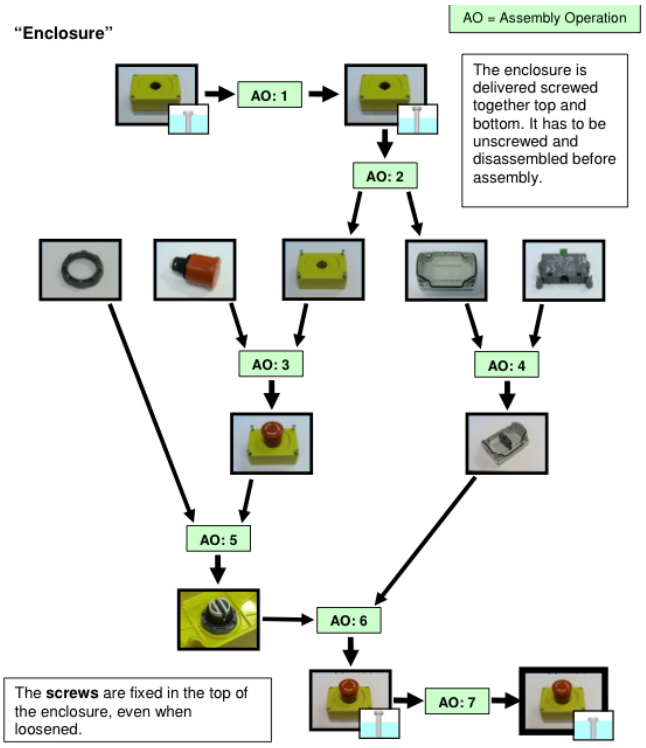

Fig. 1. Assembly graph for stop-button enclosure assembly.

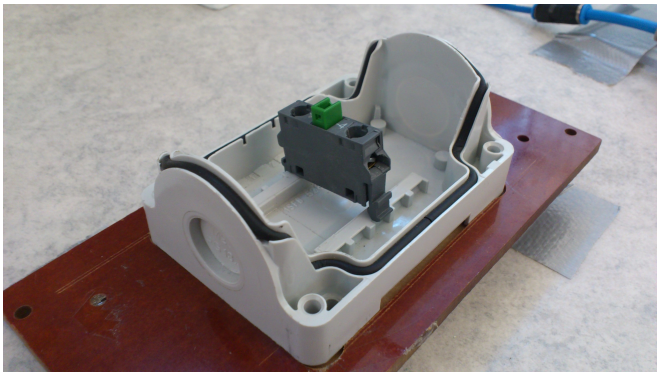

Fig. 2. Snapping the breaker into the box (AO:4).

insertion of the breaker into the plastic shell (see Fig. 2) that is modeled as a sequential function chart (SFC), see Fig. 3) which in turn is realized by a statechart running as part of the robot controller.

IV. Semantic Knowledge

In order to support task-level programming on all the levels of detail presented above, we have introduced a set of ontologies specifying our domain of discourse. First, we use a skills-and-devices ontology developed in Rosetta project [4], [3] (see Fig. 4), describing devices-skills relations. It is, in its turn, exploiting QUDT (quantities, units, dimensions, types) ontology[2] with quantities and units. On top of it we have created a parameterization ontology,

[1] www.robot-standards.org

[2] http://www.qudt.org

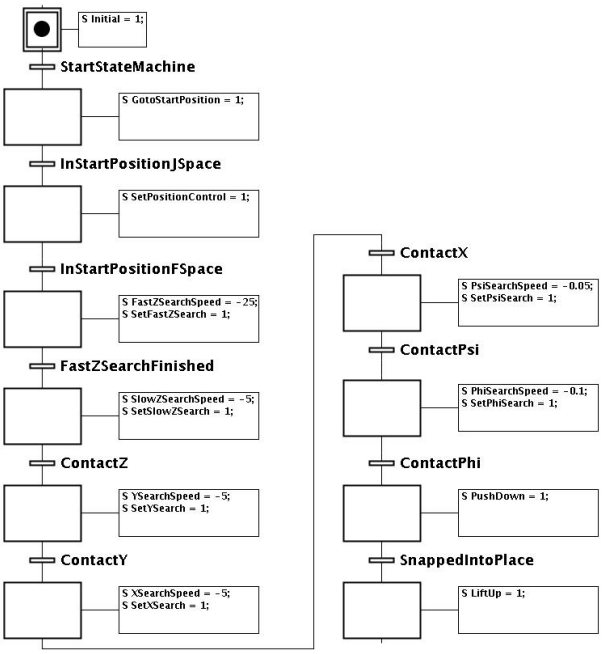

Fig. 3. Snap-fit operation (AO:4) modeled as a Sequential Function Chart.

Fig. 4. A fragment of the devices-and-skills ontology.

specific for the particular engineering tool used (ABB Robot Studio[3]). Besides, we have built a generic graph ontology and its specialization, transition systems ontology, specifying (discrete) behaviors and a number of their representations: (Sequential Function Charts of IEC61131 [11], Intermediate Modelling Language IML [4], Statecharts [12]).

The ontologies allow us to semantically represent concepts belonging to almost all involved representations, as illustrated in Fig. 5. The assembly graph, referring itself to the workpieces represented in the scene graph as objects with a given geometry, is transformed into a constraint graph (in case of assembly it usually describes various foreseen contact configurations) which in turn can be mapped into an Finite State Machine (FSM) representation of the task. Each state of the FSM is associated with an iTaSC representation, while a set of states, corresponding to a complete skill (i.e. the snap-fit assembly operation) may be treated as a parameterized program/template, with some yet unspecified error conditions to be filled in later.

This way we are prepared to translate between the representations involved if such a need arises. In particular, in case of porting task descriptions from one hardware setup to another, like when exchanging the robot used for the task, semantic representation of the involved operations proves necessary for ensuring generality of our solution.

V. System Architecture

The system we have created, shown in Fig. 6, consists of the central knowledge base, named Knowledge Integration Framework (KIF), and communicating with modern GUI-based tools used for programming robot assembly tasks (like e.g. ABB Robot Studion or KUKA WorkVisual), with the available data sources useful for learning, like log data or foreign ontologies available via WWW for robots, and with the actual system controllers for downloading and supervising the code execution.

The user creates a generic task description using the engineering tool, beginning with the assembly graph, which is stored in KIF in the RDF format, obeying the rules imposed by our ontologies. Then the assembly graph gets transformed into a constraint graph and ultimately into activity graph, possibly interacting with the user in case when some knowledge about the task decomposition into skills is missing. Then the activity graph (an SFC in one of our realizations, an rFSM in another) gets compiled into robot program and uploaded to the robot controller.

While executing the task, the declarative representation is used for monitoring and error recovery. E.g. it is possible to modify parameters of skill-implementing procedures, or to introduce additional guard steps for evaluating step pre-conditions, before resuming task execution after necessary recovery.

VI. Results

The solution has been presented as a part of the ROSETTA project demonstrator at AUTOMATICA 2012 fair. Since then we have experimented with portability of the task, successfully transferring task descriptions from one hardware setup (based on the ABB robot Frida) to another (based on

[3] http://www.abb.com/product/seitp327/-78fb236cae7e605dc1256f1e002a892c.aspx

[4] http://www.automationml.org

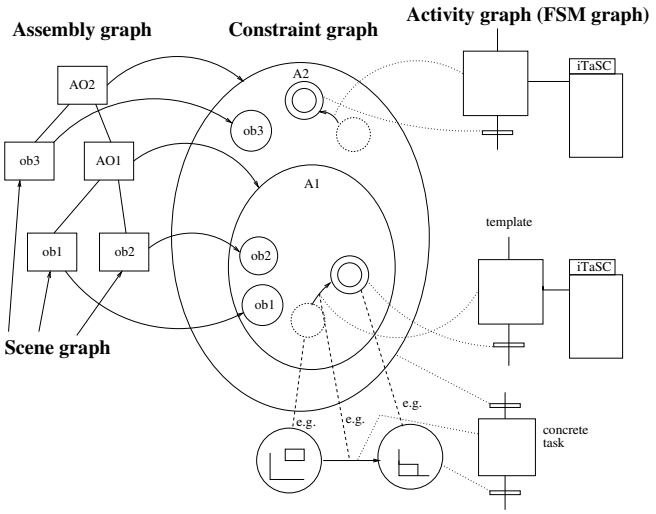

Fig. 5. Semantic characterization of the representations involved.

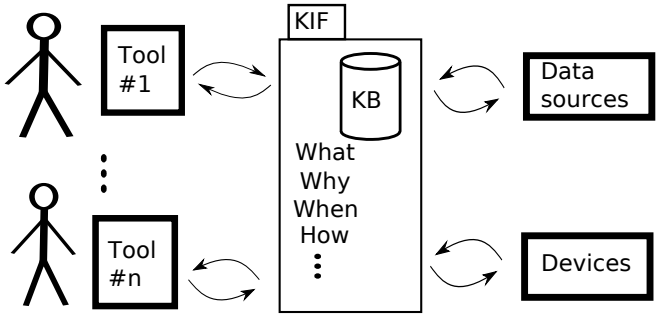

Fig. 6. The semantic solution for task representation.

ABB robot iRB-120), after necessary reparameterization. It should be noted that declarativeness of the task specification lets an unexperienced user manipulate the robot skills with relative ease.

Recently we have also experimented with transferring task specifications from ABB-hardware-based stations to KUKA-robot-based ones. Of course, this requires the hardware and vendor-specific knowledge to be available in the system, but the task-level description remains the same. This illustrates the power of declarative skill representation.

VII. Conclusions

Although we are very satisfied by the results we have achieved so far, namely portable task specifications based on semantically rich declarative representations, a lot remains to be done. One major task is to fill in the knowledge base with actual assembly knowledge, spanning the whole spectrum of possible applications. Although this task will never be finished, the acceptance for our approach among end users depends on availability of somewhat complete skill library, useful in practice. One possible way towards this end will be a world wide web of robot-knowledge, created by robots, for robots and understandable by robots. This is why semantics is important, otherwise this goal would never be realizable.

Acknowledgments

We would like to thank the numerous participants of the Rosetta and PRACE project workshops for the fruitful discussions leading to concretization of the ideas presented in this paper.

References

[1] James Allen, James Hendler, and Austin Tate, editors. *Readings in Planning*. Morgan Kaufmann, 1990.

[2] George A. Bekey. *Autonomous Robots*. MIT Press, 2005.

[3] Anders Björkelund, Herman Bruyninckx, Jacek Malec, Klas Nilsson, and Pierre Nugues. Knowledge for intelligent industrial robots. In *Proc. AAAI Spring Symposium on Designing Intelligent Robots*. AAAI Press, 2012.

[4] 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. doi: 10.1109/ISAM.2011.5942366.

[5] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, 1985.

[6] Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decré, Ruben Smits, Erwin Aertbeliën, Kasper Claes, and Herman Bruyninckx. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research*, 26(5):433–455, 2007.

[7] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning, Theory and Practice*. Morgan-Kaufman, 2004.

[8] Rafal Goebel, Ricardo G. Sanfelice, and Andrew R. Teel. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[9] Mathias Haage, Jacek Malec, Anders Nilsson, Klas Nilsson, and Sławomir Nowaczyk. Declarative-knowledge-based reconfiguration of automation systems using a blackboard architecture. In Anders Kofod-Petersen, Fredrik Heintz, and Helge Langseth, editors, *Proc. 11th Scandinavian Conference on Artificial Intelligence*, pages 163–172. IOS Press, 2011. doi: 10.3233/978-1-60750-754-3-163.

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

[11] IEC. IEC 61131-3: Programmable controllers – part 3: Programming languages. Technical report, International Electrotechnical Commission, 2003.

[12] Markus Klotzbücher and Herman Bruyninckx. Coordinating robotic tasks and systems with rFSM statecharts. *Journal of Software Engineering for Robotics*, 1(3):28–56, 2012.

[13] ABB Robotics. Robot studio 5.15 overview. http://www.abb.com/product/seitp327/30450ba8a4430bcfc125727d004987be.aspx, 2013.

[14] James G. Schmolze. Physics for robots. In *Proc. AAAI-86*, pages 44–50, 1986.