A Helping Hand: Industrial Robotics, Knowledge and User-Oriented Services

Maj Stenmark[1] and Jacek Malec[1]

Abstract— **In this paper we discuss AI in industrial robotics. In automatic control, computer vision and optimization, machine learning and data mining algorithms are widely used. However, cognition enabling mechanisms, such as high-level logic and symbolic reasoning, are still limited. This is not due to the lack of available algorithms, rather the bottleneck is knowledge representation, acquisition and transformation between different formalisms.**

In industrial robotics, cognition is not self-serving, AI technologies are rather a tool to make the user interaction, the system configuration and the task execution as cost efficient as possible. Autonomy is a mean to minimize the human workload.

In our approach, we use an online knowledge base that provides libraries with object models and task specifications, and offer services to support the user (and the robot) during programming, deployment and execution.

I. INTRODUCTION

What is considered intelligent behavior depends on the domain, for a Mars rover it is autonomy and for a toy robots it is simple user interaction. For industrial robots, it is being a good subordinate. A good subordinate is easy to communicate with, it has domain knowledge and works safely, effectively and independently, but does not take any radical decisions without approval from the supervisor.

Nowadays, computing power and storage is cheap and efficient and extensive amounts of data and processing algorithms are available. This data driven approach, together with knowledge engineering technologies such as the Semantic Web, provide a hotbed for AI research.

Within the service robotics community, the use of results from AI are commonplace, however, the field differs from industrial robotics on several key enabling factors. Service robotics is mostly research-oriented and mostly publicly funded. There is a focus on user interaction, and the solutions can be open-source and experimental. Industrial robotics on the other hand, is application-oriented, often privately funded, uses standardized or proprietary software solutions, focuses on repeatability and reliability in an often structured environment, and is regulated regarding human safety. AI has anyway entered the field through machine learning approaches used in automatic control, computer vision and optimization.

However, there is an economic factor which favors industrial robots, namely, labor cost in manufacturing industries. Still, in industries where the products have short life span, small scale production or in small businesses, the programming time and effort does not cover the savings on cheap mechanical labor. Thus, it is necessary to lower the effort of programming the robot while keeping the execution robust. This motivates the integration of AI techniques into industrial robotics systems.

We believe that building a system with domain knowledge and reasoning capabilities, we can lower the required expertise for robot instruction while keeping sufficiently robust and efficient task execution. In this paper, we present an early attempt to add AI in an industrial robotics system and discuss future research plans. We focus on the AI-related services providing user support. Our system is intended for humans cooperating with robots, therefore knowledge and the communication of knowledge is important. Since language is essential for human communication of knowledge we have taken a special interest in how natural language can be used in task instruction and user interaction.

The paper is organized as follows: first, we discuss related research, then we briefly introduce the relevant components of the system. In the end, we discuss research challenges in industrial robotics that will advance AI research.

II. RELATED WORK

In knowledge engineering, ROBOEARTH [1] is an attempt to create a possibly distributed knowledge base accessible as a "World Wide Web for robots". Although RoboEarth is robot-agnostic, and also encompasses, at least in principle, industrial robots, it does not address the issues of predominant importance for industry, i.e. real-time performance, local, closed knowledge bases and multitude of existing, developed in-house, custom methods and tools for task-level programming. A related achievement is CRAM [2], a successful attempt to enable cognition during task execution. However, it addresses mostly service tasks, putting no particular stress on real-time or accuracy and repeatability aspects of performed actions.

In industrial robotics there are a few attempts to create ontologies describing devices and task representations [3], [4], [6], these also work towards formal standardization. The ROS ecosystem has led to development of a number of informally standardized languages for semantic description of robots. The bottom one in a hierarchy and in the same time the most commonly adopted is URDF used for specifying robot hardware as consisting of links and joints. On top of it there exist a number of semantic annotation formats, like SRDF (Semantic Robot Description Format, `http://wiki.ros.org/srdf/`), which is just a set of XML-tags for specifying information not available in URDF, or SRDL (Semantic Robot Description Language) [5] which

[1]Maj Stenmark and Jacek Malec are with the Department of Computer Science, Lund University, Sweden. `maj.stenmark@cs.lth.se`, `jacek.malec@cs.lth.se`

is an OWL-based language for providing semantic information to reasoning services. SRDL is the closest previous solution to the one presented in this paper.

With a semantic reasoning layer at the top and low-level motion control at the bottom, there are several proposed architectures that attempt to bridge the gap between the high-level and the low-level representations [7]. In industrial robotics, there are several languages used for motion control. IEC 61131-3 [8] lists five standards for programmable logic controllers (PLCs), among others Sequential Function Chart (SFC) for parallel control processing. PLC languages, Statecharts [9], iTaSC specifications [10] and vendor specific robot languages are different ways to express a task. On a higher abstraction level, others express device capabilities as skills [11] that can be sequenced into a task synthesis. When building knowledge engineering systems for robotics, existing standards and formalisms should (at least to some extent) be supported to lower the integration effort.

Apart from the work in representation and standardization, the acquisition of knowledge poses another issue. Since large amounts of human knowledge is encoded in text one way to approach it is natural language parsing and knowledge extraction from online sources. Examples are Watson [12] that acquired a large amount of knowledge and won *Jeopardy!* and [13] that parse cooking recipes online and transforms them to action recipes. The latter is a prominent example of how to automatically extract skill knowledge.

Natural language is also used as a teaching and instruction of a skill. It can work as an additional input modality, e.g., when guiding the robot [14], or in high-level programming [15]. An interesting solution to verbal human-robot interaction is presented in [16], focusing on the process of context-dependent anchoring of the symbols used in the dialogue to real-world objects in the world model. However, their parser is custom-built, domain-tailored and rule-based while in the solution described in this paper we use a state-of-the-art statistical semantic parsing, offering much more generality.

III. Architecture

The distributed system consists of four main components, depicted as the rectangular boxes in Fig. 1. The knowledge base, here called Knowledge Integration Framework (KIF), is a server containing ontologies, data repositories and reasoning services. The Engineering System is a graphical user interface for high-level robot instruction that uses the data and services that KIF provides for user support. The Task Execution is built on top of the native robot controller and sensor hardware. It compiles, with the help of KIF, the symbolic task to executable files and when needed, hardware specific code, before executing it. It is implemented on a real-time enabled Linux machine, which also runs adaption and error detection algorithms.

The Engineering System and the execution environment can also communicate.

In addition to the benefit of modular exchangeable components, the rationale behind KIF as a separate entity is that the services can be black boxes. Robot vendors can offer their customers computationally expensive or data heavy cloud-based services.

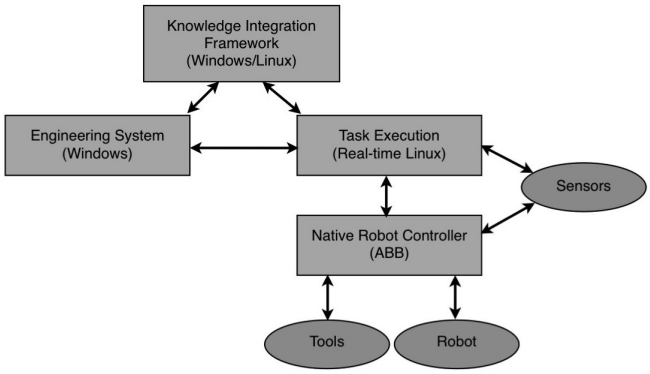

Fig. 1. The Knowledge Integration Framework provides services to the Engineering System and the Task Execution. The latter two communicate during deployment and execution of tasks. The Task Execution uses sensor input to control the robot and tools.

IV. Knowledge Integration Framework

The Knowledge Integration Framework, is a server containing a set of robotics ontologies, data repositories and service servlets. The repositories are implemented as a Sesame Triple store [17], which, together with the services are stored in an Apache Tomcat servlet container [18].

The ontologies are described in more detail in [6]. The core ontology, `rosetta.owl`[1], contains robotics devices and skills [19], [20], [21]. The devices described in the ontology provide a set of skills, and each skill is offered by one or more devices. Skills are hierarchical and compositional, where atomic skills can be combined into compound ones. A task, which we consider being the goal specification expressed as constraints [22], is realized by a state machine of skills. Skills can have several different (platform specific) implementations.

The core ontology is extended with additional ontologies, shown in Fig. 2, that are used for different purposes. `frames.owl` is an ontology describing frames of physical objects. It contains representations of geometrical location and constraints between frames, *kinematics chains*, which are later used to generate the control program. `params.owl` contains skills, their parametrization and pre-and postconditions which is necessary for knowledge based services such as planning and consistency checking. `sfc.owl` contains representations for executable state machines and includes among others Sequential Function Charts, OpenPLC and rFSM. Finally, `injury.owl` contains injury risk data needed to calculate new speeds and evading evasive motions when humans and robots share workspace.

V. Knowledge-Based Services

The reasoning in the system is principally done on KIF by a number of services that are accessible over the internet.

[1] `http://kif.cs.lth.se/ontologies/rosetta.owl`

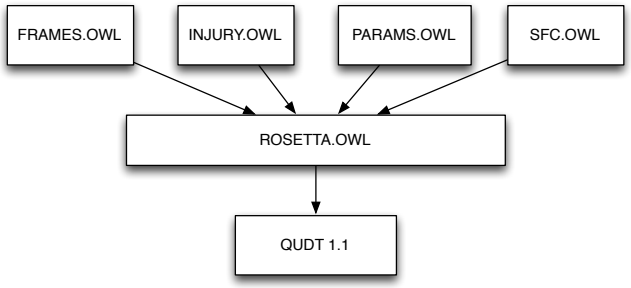

Fig. 2. The ontologies used in the Rosetta project.

This section presents the services that are currently implemented in the system. Since services are implemented as servlets, the number can easily be extended.

The most basic, but also most used, services are the object and skill libraries. The user can retrieve and reuse task and skill descriptions and upload new programs. Skills and object descriptions are represented semantically as triples or stored as text and binary files, or both, such as instances of objects with both CAD-files and semantic properties.

Other services provide user support during programming and deployment. The Engineering System uses a service for natural language programming [15]. The input to the service is English text, given directly as text or from speech-to-text in the Engineering System. In turn, the KIF service calls a generic high-performance statistical semantic parser [23] that produces predicate-argument structures for each sentence. The parser uses Propbank [24], which is a verb-centric corpus with large amounts of annotated text. The verbs have different *senses* depending on the context, *pick.01* is for example *select something from a group*, while *pick.02* refers to *bully*. Each sense has a number of arguments, *pick.01* has *arg0 - picker*, *arg1 - thing that is picked*, *arg2 - source* and *arg3 - beneficiary*. The arguments are optional, e.g., the sentence *Alice (arg0) picked out (pick.01) guitar strings (arg1) for Marley (arg3)* does not have a source argument. The parsing also extracts time, location, plural objects and manner for the actions.

In our case, the actor, given by *arg0*, can be implicitly set to the robot, while the predicate is matched to existing robot programs and the rest of the arguments are matched to objects in the robot station in the Engineering System and generates an executable sequence of steps.

For the lazy programmer, action planning and scheduling services are also available. The user can specify the assembly task by a tree of partially ordered subgoals [22] and let the planning service check the pre- and postconditions of each assembly operation and return a skill sequence. The scheduling service attempts to minimize the execution time on a system with limited resources. The current implementation is based on list-scheduling.

KIF also provides code generation for both the physical and the virtual system, e.g., one service can be used to generate JGrafChart files [25] for the sensor-based skills. Also, Maple files for kinematic calculations are stored online

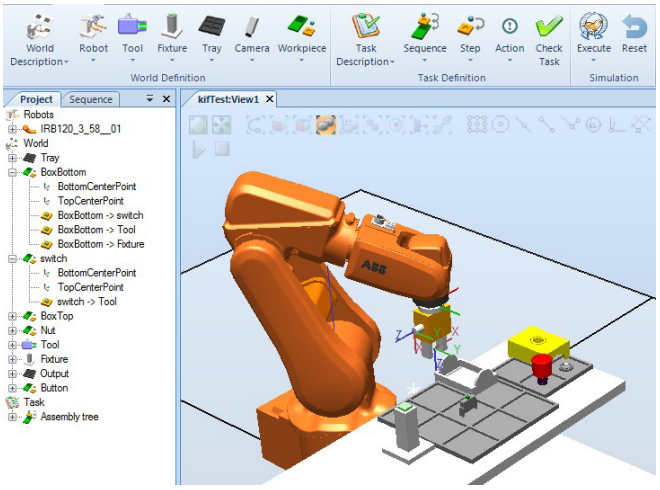

Fig. 3. The ABB RobotStudio programming environment.

from which Java-files can be generated and used to simulate robots in virtual systems.

VI. Engineering System

The Engineering System is a high-level programming interface implemented as a plugin to the programming and simulation environment ABB RobotStudio [26], see Fig. 3.

The Engineering System uses and produces KIF data for devices, workpieces, frames and skills. It is used to visualize the object geometries and properties, the frames and the task and skills.

The task specification is represented as an *assembly graph* [22], which is a partially ordered tree of assembly operations. An example of an assembly graph for an emergency stop button assembly task [27] is shown in Fig. 5. The task is transformed into a sequence of operations, graphically represented as an editable sequence. Fig. 4 displays a sequence of nested steps that combines vendor specific motions, a force-controlled assembly skill that has been downloaded from KIF and two force-controlled motions with force constraints on different axes on an object frame. Sequences can be created either 1) manually, or 2) by using the validation service on KIF, which takes the assembly graph as input and outputs a suggested sequence of actions, or 3) by the natural language programming interface described in earlier work [6], [15].

The sequence is then used to generate executable code for the target platform, which gan be both a virtual and a physical robot controller.

VII. Execution

The high-level symbolic description displayed in Fig. 4 can be executed on different platforms. One setup, shown in Fig. 6, is located at RobotLab in Lund and contains a two-armed concept robot from ABB as well as conventional ABB robot (IRB120). Our project partners at KU Leuven have a KUKA LWR. Although the low-level control in both systems are based on iTasC [10], they use different state machines descriptions, execution engines as well as robot programming languages.

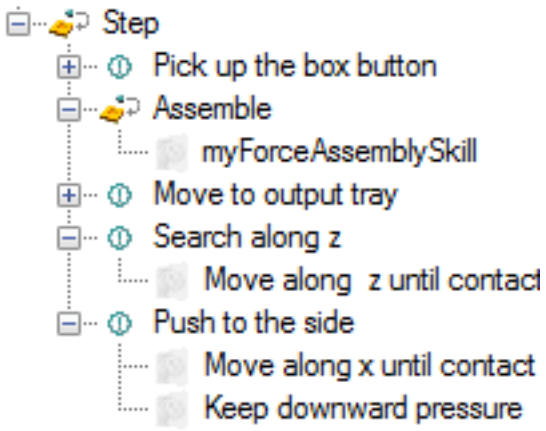

Fig. 4. An example of a sequence with hierarchical steps. First, the box button is picked, then assembled using a force-controlled skill that is downloaded from KIF. After an additional move command, there are two force-controlled motions, *Search along z* and *Push to the side* with the corresponding constraints for the motion. By right-clicking on the objects, parameters to the motions, such as force thresholds, can be edited.

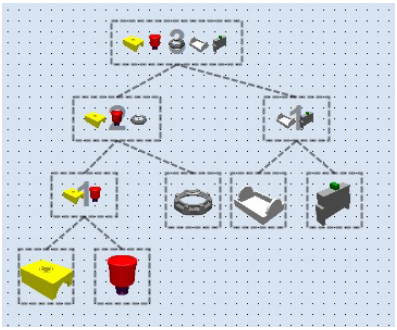

Fig. 5. An assembly graph for an assembly of an emergency stop button box. The subtrees are partially ordered, hence there are two parallel assemblies numbered 1.

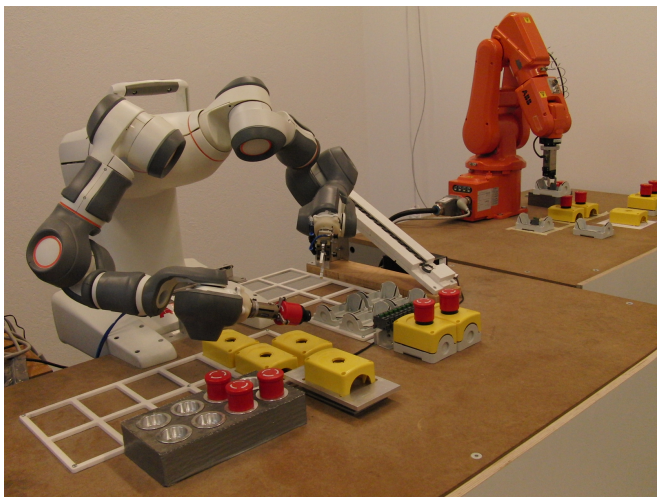

Fig. 6. The assemble workers in RobotLab at Lund University are, to the left, an ABB concept robot and, to the right, a traditional ABB industrial robot (IRB120).

In Lund, we generate vendor-specific code for simple motions and actions (such as opening and closing grippers), while sensor-based skills are implemented as state machines in JGrafChart [25]. When a skill is uploaded to the skill library, the statechart is annotated semantically. This procedure is reversed during the deployment and a new xml-file is generated. If the skill was created from scratch in the Engineering System using motion constraints, the full kinematic chain has to be extracted first.

When programming tasks using force control, it is a challenge to select suitable force-control parameters. The parameters depend on the stiffness of the involved materials. Stolt et al. [28] developed a self-tuning algorithm based on Recursive Least Squares to experimentally determine the stiffness and damping factor of an impedance controller. Other parameters can be learned from training data, such as uncertain contact positions [27].

During the execution, errors and non-nominal sensor readings are detected by creating sensor signature models using Hierarchical Dirichlet Process Hidden Markov Models [30], [29]. Error-handling procedures can be added to the task to cope with common errors, however, this is currently done by a human operator.

VIII. Experiments

We have experimented with the emergency stop button box assembly shown in Fig. 5. Initially, there are five separate parts: an emergency button, the top of the box, the bottom of the box, a switch and a nut. The complete assembly consists of four force-controlled assembly operations. The emergency button should be placed in a hole on the top of the box using a peg-in-hole skill, then rotated and fastened with the nut using a two-armed screwing operation. The switch should be snapped into place on the bottom of the box before the two subassemblies are joined together into the final product.

The programs can be generated from the assembly graph by annotating the assembly operations with skill types from the skill databases (without annotations there will be placeholder assembly skills). As an example, the switch is attached to the box bottom using a *SnapFit* skill. The *SnapFit* is stored in KIF with pre- and postconditions as well as parameters. One precondition is that the switch has to be held in the gripper and the switch is at a start position above the box bottom. When the planning service is invoked it will return a sequence of actions including intermediate picking, moving and placing of objects.

The knowledge base contains also a number of typical object descriptions, useful while defining a work cell, like feeders, trays, fixtures, tables, etc., thus simplifying the world definition phase of the programming and setting the ground for further symbol anchoring later on. All the defined skills come with parameterization and pre- and postconditions necessary for consistency checking and reasoning.

The scheduling service requires that the execution times and tools for each operation are specified, as well as the partial ordering of the actions and number of cycles. E.g., the screwing of the nut is a two-armed operation using two

specific tools, while a third tool is used for gripping the box bottom. There is a tool changer in the station, however, the time to change tools can add precious seconds to the cycle time and are therefore also included when generating a schedule for the two-armed robot.

The sequence can also be generated by natural language input. The number of skill types labelled with natural language is limited, excluding synonyms it is 10. Skills that are labelled with predicates, such as *pick/take* and *place/put*, can be expressed in unconstrained language. Example sentences are ”Could you please pick the switch and insert it on the box bottom.” and ”The switch should be picked and inserted on the box bottom.” However, the system will be tricked by ”The switch is not ready for picking”, which also results in a *pick.01* and *the switch* as *arg1* because the negation is not related to the verb.

IX. FUTURE WORK

Ideally, the robot should be able to reason about knowledge it has and does not have, draw conclusions to why an execution failed and optimize the task and suggest improvements. These types of reasoning systems should be coupled with a comprehensive dialogue system, so that the human and the robot can *discuss* the task more freely. At the moment, the initiative is entirely the human's: the user employs the programming and planning support to create a task, the adaption algorithms to automatically select parameters and the error detection algorithms to learn nominal behavior. For the robot system to be perceived as intelligent, *it has to take the initiative* as well and act more autonomously and goal oriented.

We have four interest areas where we want to enhance the AI in industrial robotic systems:

Robust execution and error handling. Symbolic task descriptions make it possible for the robot to understand the meaning of its actions and plan the execution, while error detection algorithms provide a data-driven classification of the execution. During failure, the robot has very little understanding of the cause of the error, only that something failed in the current state of the skill. To improve the robustness of the execution, several issues have to be solved: How can errors be semantically labelled automatically? How can we automatically generate error handling policies from knowledge about the task and the environment (such as the object CAD data)? How can the robot explain for the human what and why the execution failed?

Developing the natural language interaction. Initially we want to develop the natural language programming interface to allow more diverse actions, such as navigating in the program (”Restart the program after the two-armed assembly”) and creating skills from scratch (which is still done best using visual programming tools). In a longer perspective, the entire robot system should be supported by a natural language interface, where the user can command actions such as ”Run the parameter adaption routine”.

Developing dialogue systems. We need to develop a mixed initiative dialogue system that governs the user interaction. The dialogue system has to consider both the user input and the current state of the system and decide what actions the robot system should take and what user input should be requested.

Knowledge acquisition. The robot system can have access to enormous amounts of data, both from online sources and from sensor input. An open research problem is to translate data to knowledge, to automatically create ontologies from reading manuals or from learned sensor data. We urgently need tools and techniques to acquire knowledge about skills in order to populate the skill libraries. In PRACE project (`http://prace-fp7.eu`), one of the goals is to create user-friendly programming tools, however, how can we generalize and extract the knowledge from the skill demonstrations that is needed to reason about the skills (action planning, error handling, etc)?

X. CONCLUSIONS

In this paper, we have presented a short overview of how we have added AI services to support the users in programming and setting up industrial robotic systems. The distributed architecture provides a way for robots to share knowledge and access remote reasoning services and computing, that rely on large amounts of data or computing power. The results presented are a first step to give industrial robots cognitive abilities and the work continues within the European FP7 projects PRACE and SMErobotics.

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for comments that have led to improvement of the paper.

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) and No. 287787 (project SMErobotics).

REFERENCES

[1] 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, Oliver Zweigle and M.J.G. (René) Van de Molengraft. RoboEarth, In *Robotics and Automation Magazine, IEEE*, vol 18, no 2, pp 69–82, June 2011.

[2] Michael Beetz, Lorenz Mösenlechner, Moritz Tenorth, *CRAM: A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments*, In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18–22, 2010, Taipei, Taiwan.

[3] Stephen Balakirsky, Zeid Kootbally, Craig Schlenoff, Thomas Kramer, and Satyandra Gupta, *An Industrial Robotic Knowledge Representation for Kit Building Applications*, Proc. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 7–12, 2012, Vilamoura, Algarve, Portugal.

[4] Joel Luis Carbonera, Sandro Rama Fiorini, Edson Prestes, Vitor A. M. Jorge, Mara Abel, Raj Madhavan, Angela Locoro, Paulo Goncalves, Tamas Haidegger Marcos E. Barreto and Craig Schlenoff, *Defining Position in a Core Ontology for Robotics*, Proc. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 2013.

[5] Lars Kunze, Tobias Roehm and Michael Beetz. Towards Semantic Robot Description Languages, In *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[6] Maj Stenmark and Jacek Malec, *Knowledge-Based Industrial Robotics*, To appear in proc. of The 12th Scandinavian AI conference, Aalborg, Denmark, November 20–22, 2013.

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

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

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

[10] 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.

[11] 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.

[12] David Ferrucci, et al. Building Watson: An Overview of the DeepQA Project, *AI Magazine*, Vol. 31, No. 3, 2010

[13] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr and Moritz Tenorth. *Robotic Roommates Making Pancakes*, In Proc. of EEE-RAS International Conference on Humanoid Robots, 2011.

[14] Matthew R. Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth Teller. *Learning Semantic Maps from Natural Language Descriptions*, Robotics Science and Systems, 2013.

[15] Maj Stenmark and Pierre Nugues, *Natural Language Programming of Industrial Robots*, To appear in Proc. International Symposium of Robotics 2013, Seoul, South Korea, 2013.

[16] Séverin Lemaignan, Raquel Ros, E. Akin Sisbot, Rachid Alami and Michael Beetz. Grounding the Interaction: Anchoring Situated Discourse in Everyday Human-Robot Interaction, In *Int. Journal Social Robotics* , vol. 4, pp. 181–199, 2012. doi: 10.1007/s12369-011-0123-x.

[17] OpenRDF Sesame, http://www.openrdf.org.

[18] Apache Tomcat, http://tomcat.apache.org.

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

[20] Andres Nilsson, Riccardo Muradore, Klas Nilsson and Paolo Fiorini, *Ontology for Robotics: a Roadmap*, Proceedings of The Int. Conf. Advanced Robotics (ICAR09) , Munich, Germany, 2009.

[21] Anders Björkelund, Jacek Malec, Klas Nilsson, Pierre Nugues and Herman Bruyninckx. *Knowledge for Intelligent Industrial Robots,* Proc. AAAI 2012 Spring Symp. On Designing Intelligent Robots, Stanford Univ., March 2012.

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

[23] Anders Björkelund, Bernd Bohnet, Love Hafdell and Pierre Nugues, *A high-performance syntactic and semantic dependency parser*, Proc. of the 23rd International Conference on Computational Linguistics: Demonstrations. Association for Computational Linguistics, pp. 33–36, 2010.

[24] Martha Palmer, Daniel Gildea and Paul Kingsbury, *The Proposition Bank: An Annotated Corpus of Semantic Roles*, Computational Linguistics, March 2005, Vol. 31, No. 1, pages 71–106. Association for Computational Linguistics, 2005.

[25] JGrafChart, http://www.control.lth.se/grafchart.

[26] ABB RobotStudio, http://new.abb.com/products/robotics/robotstudio.

[27] Andreas Stolt, Magnus Linderoth, Anders Robertsson, and Rolf Johansson. Force controlled assembly of emergency stop button. In *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[28] Andreas Stolt, Magnus Linderoth, Anders Robertsson, and Rolf Johansson. Adaptation of force control parameters in robotic assembly. In *10th International IFAC Symposium on Robot Control*, Dubrovnik, Croatia, September 2012.

[29] Enrico Di Lello, Markus Klotzbucher, Tinne De Laet, and Herman Bruyninckx. Bayesian time-series models for continuous fault detection and recognition in industrial robotic tasks. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013.

[30] Enrico Di Lello, Tinne De Laet, and Herman Bruyninckx. Hierarchical dirichlet process hidden markov models for abnormality detection in robotic assembly. In *Workshop on Bayesian Nonparametric Models (BNPM) For Reliable Planning And Decision-Making Under Uncertainty, NIPS 2012*, Lake Tahoe, USA, December 2012.