On the Integration of Skilled Robot Motions for Productivity in Manufacturing

Björkelund, Anders; Edström, Lisett; Haage, Mathias; Malec, Jacek; Nilsson, Klas; Nugues, Pierre; Robertz, Sven; Störkle, Denis; Blomdell, Anders; Johansson, Rolf; Linderoth, Magnus; Nilsson, Anders; Robertsson, Anders; Stolt, Andreas; Bruyninckx, Herman

Published in:
IEEE International Symposium on Assembly and Manufacturing (ISAM), 2011

DOI:
10.1109/ISAM.2011.5942366

2011

On the integration of skilled robot motions for productivity in manufacturing

Anders Björkelund, Lisett Edström
Mathias Haage, Jacek Malec
Klas Nilsson, Pierre Nugues
Sven Gestegård Robertz, Denis Störkle
Dept. of Computer Science
Lund University, Sweden
Email: klas.nilsson@cs.lth.se

Anders Blomdell, Rolf Johansson
Magnus Linderoth, Anders Nilsson
Anders Robertsson, Andreas Stolt
Dept of Automatic Control
Lund University, Sweden

Herman Bruyninckx
Dept. of Mechanical Engineering
K.U.Leuven, Belgium

***Abstract*—Robots used in manufacturing today are tailored to their tasks by system integration based on expert knowledge concerning both production and machine control. For upcoming new generations of even more flexible robot solutions, in applications such as dexterous assembly, the robot setup and programming gets even more challenging. Reuse of solutions in terms of parameters, controls, process tuning, and of software modules in general then gets increasingly important.**

There has been valuable progress within reuse of automation solutions when machines comply with standards and behave according to nominal models. However, more flexible robots with sensor-based manipulation skills and cognitive functions for human interaction are far too complex to manage, and solutions are rarely reusable since knowledge is either implicit in imperative software or not captured in machine readable form.

We propose techniques that build on existing knowledge by converting structured data into an RDF-based knowledge base. By enhancements of industrial control systems and available engineering tools, such knowledge can be gradually extended as part of the interaction during the definition of the robot task.

I. Introduction

Productivity in manufacturing is the basis for competitiveness, and thereby the basis for developments and businesses within and around manufacturing systems. Manufacturing, being the transformation of resources into products that meet market needs, is the key to our wealth, now and in the future when those resources are increasingly recycled and scarce. In other words, on an overall level, to live well we have to manufacture well.

The need for productivity clearly implies the need for performance. In long-batch large-volume production, high-performance motions and processes can be achieved by extensive engineering, leading to for instance fixed automation or special purpose machines. Today, however, the other cornerstone of productivity is flexibility, which is to meet variations in products (customizations, change of suppliers, etc.), processes (environmental changes, etc.), and resources (continual improvements of equipment, lowering machine costs, etc.). For the combination of flexibility and performance, in manufacturing just as in software and control, a model-based approach is appropriate.

A. Model-based system integration

With the aim of facilitating system integration and to make system integration solutions reusable, modeling for automation systems has been subject to extensive work and standardization, for instance according to AutomationML that has a basic structure for products, processes and resources [1], [2]. However, while such models facilitate transfer of production setup data, the approach is classical and limited in the sense that systems are engineered based on the nominal/desired behavior, with variations during production taken care of by sensing and actions that are engineered to manage those variations. For instance, variations in the location of a part can be sensed (e.g., by a touch sensor or via vision), or motions can be designed to physically cope with the expected variations.

The necessary adjustments during startup of a new production line can of course be extensive and costly, and that is also why automation is not yet productive in one-off (or very short series) production. Problems to consider include:

- Competing stakeholders and technology providers try to make customers dependent on their specific solutions, and incompatibilities follow when there are business reason for going outside standards.
- When standards are appropriate or enforced by customers or legal aspects, they are typically too slow in adopting new technologies.
- Even when standards apply and vendors agree, there are human errors and product generations that make things incompatible, typically in a way that no single component can be pinpointed as the failing one.

The situation is particularly difficult in automation due to the large number of heterogeneous techniques that are to be integrated, and robots for assembly and machining are perhaps the most difficult case due to uncertainties in combination with force interaction. Thus, efficient engineering works within business units or stakeholder organizations, but for the overall system integration we have to realize that a different way of thinking is crucial. Since part of the thinking has to be embedded into the systems, we have arrived to the need for cognitive systems.

978-1-61284-343-8/11/$26.00

B. Cognitive robots in manufacturing

In the EU FP7 work program [3] that is the basis for our projects, Section 2.4 (p 27) about Challenge 2 on "Cognitive Systems, Interaction, Robotics" states:

> *Engineering systems with the capability to sense and understand an unstructured environment is a challenge which goes beyond today's systems engineering paradigm. Present day systems engineering relies on specifying every eventuality a system will have to cope with in the execution of its task(s), and programming the appropriate response in each case.*
>
> *Challenge 2 aims to extend systems engineering to the design of systems that can carry out useful tasks ... in circumstances that were not planned for explicitly at design time.*
>
> *Research and development efforts should aim at generating actual design principles. They will contribute to establishing scientific foundations for such principles. Alternatively, they may aim to achieve significant engineering progress, e.g. through integration.*

The anticipated approaches can of course be seen as extensions of model-driven systems engineering, with more dynamism (design subject to change during system operation) and flexible definitions of what is data and what is meta-data. Nevertheless, the quote very well captures the challenges of our work, and it means we need to bring semantics into system integration such that not only data is structured into information, but also that there are machine readable definitions of the meaning of information. That forms what we refer to as knowledge, which should be declarative rather than normative.

Robots comprise the key challenge due to their flexibility (not designed for a specific application), their compliance due to the dexterous mechanical design (e.g. compared to CNC machines), and their motion deviations when process forces are present (e.g. during assembly and machining). The nominal models (such as inverse kinematics) only covers a small part of the information needed for productivity in manufacturing. Robots at manual workplaces, in close interaction with humans that have a variety of expectations on what a robot can do, make the cognitive approach even more tractable.

Today, most knowledge about how to deal with uncertainties during setup and programming resides in the head of system integrators and advanced users. Reuse and support for less advanced users would require knowledge to be acquired and utilized within the system and its components. Many such attempts have been proposed over the years, but it is important to take into consideration:

- Experienced system integrators and expert users are very busy persons that make a profit out of their knowledge, and they will of course not spend time on entering their knowledge into some fantastic expert system unless they get immediate return on investment.
- Tasks today describe the nominal case, while management of variations are implicit in terms of the sensor data processing it entails. To enable reuse, the use of sensing (and motion settings that contribute to robustness) need to be made explicit in, telling "why" and "when" in addition to the standard "how".
- Basically production systems today do rarely fail, because when they did during startup there were the experts fixing it, which means making the system robust in accordance with available human experiences. Hence, a cognitive approach should gather information from failure situation and how they were fixed.
- Solutions have to match business models (should be profitable to use, from week one), responsibilities (who are to be contacted in each error situation), safety (can certification be done easily when needed), and IPR issues (who owns the obtained knowledge).

A widely applicable approach considering these items such that it builds on existing tools and practices, but enables cognitive functions to be added, is the topic of this paper.

C. Approach and methodology

Cognitive robotics is still in its beginning with many fundamental difficulties when it comes to autonomous behaviors and human interaction in fully unstructured environments [4], [5]. To reach industrial use within a reasonable number of years, our approach is to do full scale prototyping to confront scientific claims with real application settings. As a step in that direction, industrial collaboration with relevant laboratory setups forms our methodology.

We start with actual data from description of systems and processes (but not with predefined models), then making use of whatever structure there is (e.g., from XML schemas and from data types that are part of component interfaces) but converting it into the RDF-based triple store [6]. By linking the data together based on the relation, that RDF data forms graphs that we also refer to in terms of predicate argument structures. That also permits graphs to be combined in new ways over Internet connections [7], and representation of semantic information is supported, just like the semantic web but now focused on robotics in manufacturing. The predicate-argument structure is also the main representation in natural language understanding, which we are also interested in for the human-robot interaction later on.

Based on actually obtained data, structure and knowledge will be inferred. For building the actual systems, considering robotics research being a small domain compared to networked software in general, it is crucial to build systems based on available standards and (freely) available software packages. The aim is then to create a portable system that seamlessly integrate with current engineering practices and facilitates stepwise refinement of knowledge, ranging from low-level motion control with adaptive functions and up to interactive task definitions including explicit models of uncertainties and the intention of motions in the context of manufacturing.

II. Preliminaries

With a goal of easy and productive usage of robots, we need to care about system integration and robot programming steps to reach that goal, but software and feedback-control difficulties should be abstracted away from production engineering. While that is the aim of the engineering tools, we are far away from that situation today. This section presents some of the overall issues that should be understood before an integrated and practically applicable approach can be developed, and it also contains references to related work.

A. Robot systems engineering

Since the start, some 50 years ago, of the domain of robotics, large amounts of useful robotics functionalities have been developed: controllers, planners, kinematics and dynamics transformations, estimation and learning algorithms, constrained and multi-objective optimization, etc. An impressive amount of human ingenuity, knowledge and experience has found its way into many millions of lines of robotics software code, integrating developments and insights from various "supporting" domains, such as mechatronics, systems and control theory, computer science, machine learning, (probabilistic) logic, or computer vision.

The bad news, however, is that this steady growth of useful functionalities has outpaced the speed with which developers can use all those functionalities effectively, in ever more complex robot systems. A major bottleneck is the amount of knowledge that human developers have to master, before they can integrate components, in such a way that they can work together in the first place, and be exploited to their full potential in the second place. As part of so called Model-Driven Engineering (MDE), software functions are put into composable components that should be designed such that "pure" knowledge is separated from application-specific configuration in particular contexts.

The trend in MDE is to generalize by defining (and standardizing) meta-data descriptions, and to improve efficiency by better engineering principles and more powerful software-development tools. Despite that, the closed-world mindset remains, for instance in XML schemas that define what meta data (or model data) is correct. However, things change, and so do meta-data and agreed interfaces. Therefore, the semantic web techniques provide a better basis for the composition and usage of components, which means integration of knowledge in an incremental manner that can stand inconsistencies and human preferences. That is the aforementioned Open World Assumption (OWA), which (at Page 11) in [8] is formulated as :

> *An open world is one in which we must assume at any time that new information could come to light, and we may draw no conclusions that rely on assuming that the information available at any one point is all the information available.*

Wikipedia provides a similar formulation according to (http://en.wikipedia.org/wiki/Open_world_assumption):

> *Heuristically, the open world assumption applies when we represent knowledge within a system as we discover it, and where we cannot guarantee that we have discovered or will discover complete information. In the OWA, statements about knowledge that are not included in or inferred from the knowledge explicitly recorded in the system may be considered unknown, rather than wrong or false.*

Whereas MDE and software engineering teaches design first, the OWA permits a data-first approach, and then gradual introduction of the modeling concepts that actually appear. To connect to actual engineering principles, the approach taken here is to have components with self-descriptive interfaces that also can be embedded in physical devices for real-time operation, to store interface descriptions as initial knowledge (see next subsection), and to support semantic tagging (gradually by users) that is then stored in the knowledge base. That way closed-world components can meet the open-world usage, and loose coupling between subsystems is promoted. The self-descriptive interfaces must use semantic tags that are described in an ontology.

B. Obtaining initial knowledge

A large amount of information about products, equipment (and other resources) and production processes, is available in terms of documentation, programs, configurations or data-sheets for devices. That information is mainly in text (or available software can convert from native formats to text with structure), and to separate structure from content it is common to use XML. Clearly, it is useful to have some basic structure for the fundamental definitions that even competitors can agree on, which is what an ontology is useful for as explored in our earlier work [9]. Such an ontology should be small, not covering parts that competitors will not agree on anyway, and it should not be normative in case of modeling alternatives. Ontology data that fulfills certain requirements on consistency and completeness can also be used to generate compilers and for compilation into imperative code definitions [10].

Based on these insights, our project Rosetta started with implementation of a Knowledge Integration Framework (KIF) built on mainstream semantic web standards such as RDF, and including transformation from text to knowledge-base data [11]. Primary source documents are in the AutomationML XML format. During import of such documents, the information is converted to RDF and some basic knowledge about the separation of products, resources and processes is used to maintain some basic structure of the knowledge, such that automation engineers can continue utilizing the information via the well-known hierarchies (such as instances, system units, and roles).

Our conversion of the original tree-structured XML documents into graph-based knowledge (information interconnected with relations and semantic data expressing the meaning) has worked well for a variety of test cases. Therefore, although much work remains to make it industrially useful and maintainable, we do not see that part as the main problem. It

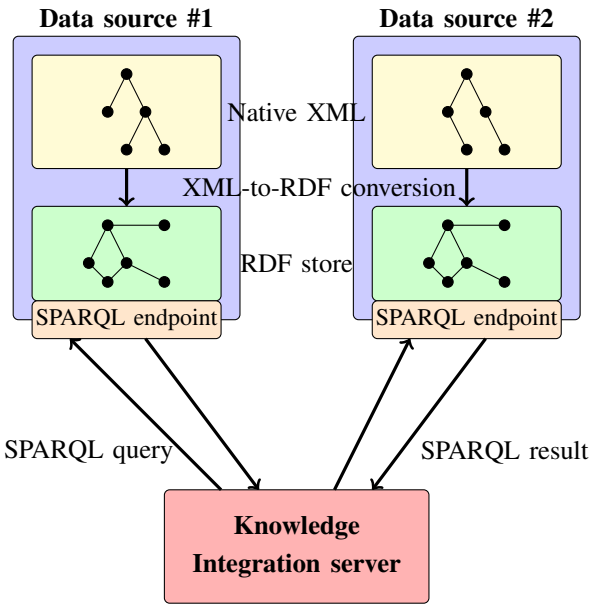

Fig. 1. Harvesting information from XML documents for establishment of automation knowledge.

should be noted, as indicated in Figure 1 and commented in [11], data and knowledge sources can be distributed over the Internet by means of the utilized semantic web techniques and software packages. The top three remaining issues are:

1) AutomationML (and other sources) is focused on explicit production plant data, with the task descriptions (e.g., in terms of state machines, robot programs, and PLC data) being implicit and difficult to reuse. To prepare for reusable task descriptions, we need a way to represent both desired and actual engineering/user decisions such that it can be captured and gradually enhanced when things change.
2) Knowledge-based solutions (e.g., to the previous item) need to be connected to existing engineering tools such that current practices continue to work, but also such that new knowledge-based features can be tried out without major thresholds.
3) Productivity implies performance demands, and in combination with OWA (the Open World Assumption) it can be concluded that third parties should be able to extend the sensing and motions control of any robot. The issue is then how the high-level engineering tools and knowledge-based techniques can be connected to the low-level real-time control of physical motions.

The third item also implies that application-specific motion control should be possible to map back to the user/engineering level for human interaction, including the case with virtual control when physical dynamic models are missing. For instance, for visualization in an engineering environment without a physics engine, force controlled motions can be emulated by direct commanding of the position variations it is expected to cope with.

C. Separation of concerns

The lack of reuse of information such as software and formalized knowledge is a problem in many areas, and since robotics and automation has to build on tools and principles from other domains, we better ask ourselves what remedies can possibly be applied. First of all, keeping the OWA in mind, it is not about defining one new tool chain (in the open world you cannot prescribe engineering environments), but portable reference implementations of compositional solutions can be useful. Most important is to find simple principles that can be applied in different environments and for different types of systems. As confirmed within our earlier FP6 projects RoSta, SMErobot and PalCom the most central engineering approach is to keep apart the following concerns [12]:

- **Communication** defines how agents communicate with each other.
- **Computation** defines the implementation of the behavior of individual agents. It thus determines what is being communicated.
- **Configuration** defines the interaction structure, or configuration. It states which agents exist in the system and which agents can communicate with each other.
- **Coordination** defines patterns of interaction, i.e., it determines when certain communications take place.

That is also well in line with other published results [13], [14], [15], although there are variations such as splitting configuration or treating configuration and coordination in the same way [16].

The concept of separation of concerns is classic within computer science, but we need practical examples from application areas. Communication is typically based on connection-based asynchronous messages, so that is not a major issue. In automation there is already a good practice about configuration, which is what is done before the system is put into operation. Thus: Keep computation and coordination apart.

D. Constraint-based task specification

The constraint-based task specification framework specifies the relative motion of objects by imposing constraints. To be able to specify these constraints a so called kinematic chain is needed, and it consists of two object frames and two feature frames. The first object frame is usually attached to the object one wants to manipulate and the second object frame is usually attached to the robot. The feature frames should be attached to features on the object to manipulate and on the robot. They should be chosen in such a way that the task constraints become as easy as possible to specify.

A kinematic chain should have 6 degrees of freedom, and they are distributed over the transformations between the feature and the object frames [17]. These six degrees of freedom are represented by χ_f, the so called feature coordinates. There might also be uncertainties in the pose between the previously defined coordinate frames, which is represented by an additional transformation between each of the previously mentioned coordinate frames is introduced. These uncertainty coordinates are denoted χ_u.

III. Example applications and experiments

The specifics for the following two application examples (in terms of application context, algorithms, and control implementations) are detailed in other recent publications. The focus here is to illustrate the task specification and system integration that we have used to experimentally verify the targeted semantic approach. A complete description does not fit into a few pages, but it deserves to be mentioned that all the proposed concepts (although not yet in a single packaged system) have been implemented and experimentally verified.

A. Force-controlled assembly

The assembly scenario in this paper is to assemble the internals of an emergency stop button. An electric breaker should be snapped into the middle one of five available slots, and in this case the breaker can be grasped either on its long or its short sides using a parallel gripper. Optionally, however, more breakers for additional switching functions (such as connecting alternative equipment upon the emergency stop) can be placed in the adjacent slots, and then the grasping in practice is constrained to the short sides of the breaker. With the short side contact only, there is a rotational uncertainty around the axis intersecting the two gripping points, which can be handled in different ways:

a) All uncertainties in fixtures, feeder locations, grasping, work-piece variations, and variations between robot individuals can be analyzed and (for the actual type of robot at hand) motion specifications in terms of robot programs can be engineered.
b) If the bottom part is accurately fixed at a known location relative to the robot, we can make use of product data. That is, by using angled fingers that grasp on the short side but touches one long side such that the undesired rotation would be fixed.
c) Sensor-based motions can be used, which could mean vision and/or force. While vision would be appropriate for locating parts, we decided to implement the mounting by means of force-torque (F/T) control, which permits an online adaptation during the physical contact and the F/T sensor can be used to detect when the mounting is completed (by the force transient from the final snap).

Alternatives a and c were fully implemented (in laboratory environments) and b will simply be finished just for completeness. The algorithmic aspects including the F/T-control and the detection of the final snap are detailed in [18]. Alternative b is a production-suitable way of limiting the uncertainty (physical solution by means of product-related finger design) avoids the extensive production engineering of item a, as well as the need for external sensors of item c (but then without snap feedback).

In the following alternative c implementation, force sensing is used to resolve these uncertainties, and the motions have been specified using the constraint-based task specification methodology (iTaSC) [17]. The assembly task and the object and feature frames related to it are shown in Figs. 2 and 3, whereas Fig. 4 shows how the frames are related to the feature coordinates.

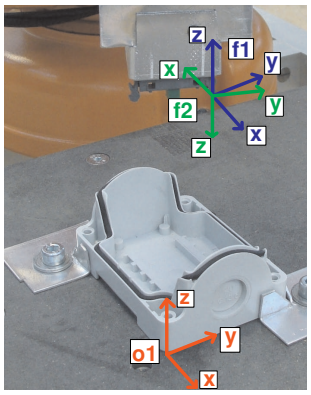

Fig. 2. Illustration of the different coordinate frames in the assembly task.

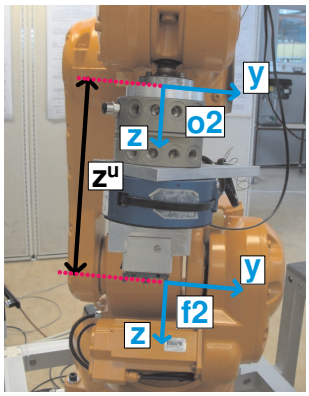

Fig. 3. Illustration of the uncertainty coordinate z^u between the two frames $f2$ and $o2$.

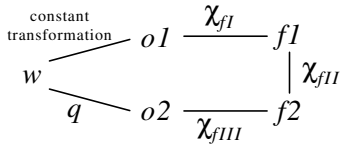

Fig. 4. Schematic description of the kinematic chain in the assembly task. w denotes the world coordinate frame and q denotes the robot joint coordinates.

- Object frame $o1$ is attached to the box. It is related to the world coordinate frame by a constant transformation.
- Feature frame $f1$ is attached to one end of the switch. The orientation is the same as $o1$.
- Feature frame $f2$ has its origin in the same position as $f1$, but the orientation is the same as the robot flange frame.
- Object frame $o2$ coincides with the robot flange frame.

The feature coordinates χ_f are divided into three groups depending on which frames they relate. The coordinates are

$$\chi_{fI} = (x, y, z) \qquad o1 \rightarrow f1$$
$$\chi_{fII} = (\varphi, \theta, \psi) \qquad f1 \rightarrow f2$$
$$\chi_{fIII} = (-) \qquad f2 \rightarrow o2$$

The coordinates χ_{fI} give the position of the origin of $f1$ using Cartesian coordinates in $o1$. χ_{fII} describe the rotation from $f1$ to $f2$ using Euler ZYX-angles. χ_{fIII} has no feature coordinates, since the transformation between $f2$ and $o2$ is fix. All feature coordinates were chosen as outputs:

$$y_1 = x \quad y_2 = y \quad y_3 = z$$
$$y_4 = \varphi \quad y_5 = \theta \quad y_6 = \psi$$

Uncertainties in the task include the exact location of the box and its orientation. They are however resolved using guarded search motions, i.e., the motion is velocity controlled in the search direction and stopped when a contact force is detected. Once contact is made, it is maintained by using force control, and hence no explicit uncertainty coordinates are used to model this uncertainty. The exact position of the grasp is also assumed to be uncertain, and the z-distance from $f2$ to $o2$ (Fig. 3) is therefore modeled as an uncertainty coordinate z^u.

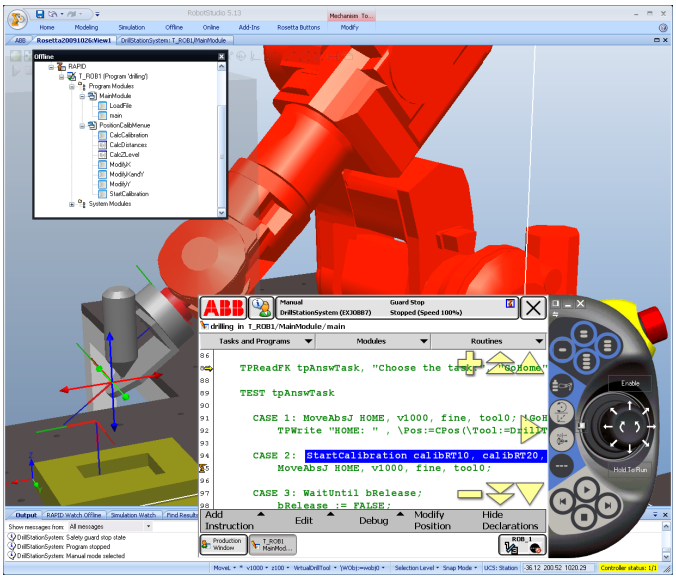

Fig. 5. Interactive testing of the drilling skill with motions constraints evaluated within the user dialog but in generic external code (as dynamically loaded Java code) from the knowledge base.

Fig. 6. Physical testing of the drilling skill with manual guidance (force-controlled motions) implemented as a real-time extension with 4ms sampling rate. Boundary conditions are evaluated by calling the corresponding C code.

The portability of the motions specifications were successfully verified by letting the neighbor robot hold the bottom part, which corresponds to a dual-arm assembly operation. The point is, since the iTaSC constraints are declarative with the property of compositionality, it is well suited as a basis for specification of motions below the symbolic level and on top of the actual motion controls of a (set of) specific robot(s). We then refer to the breaker mounting in terms of a skill SnapFit, which has been implemented and experimentally verified.

B. Manually guided drilling

While the SnapFit illustrates how declarative (and KIF suitable) knowledge (being part of what we will refer to as a *skill*) can be used to specify the motion and the uncertainties, the following prototyped drilling example exemplifies other compositional skill properties such as boundary conditions (depicted below) and rules for force interaction near singularities (omitted for brevity). The following was implemented and tested for the PlanarDrilling (Fig 6):

- F/T-control parameters (exposed and semantically tagged for the KIF as for SnapFit) were set to force-control only with the tool orientation fixed according to the vertical drilling orientation.
- As for the SnapFit the F/T-control was accomplished via an external skill- and motion-level controller.
- The force control was mapped to position interaction during configuration and task description in ABB RobotStudio (ABB-RS, depicted in Fig 5).
- A bounding-box for permitted work-space for the manual guidance was introduced. That could be from the production engineering by selecting boundary conditions for the skill at hand, or imposed from rules in the KIF and then extending the description of the skill instance. Based on the declarative description, either code for real-time force control or for the ABB-RS interaction control can be generated.
- The overall coordination permits hand-over from iTaSC motions to teach-pendant interaction and to explicit motions according to the robot language. In this case, using ABB robots, there is always a piece of Rapid code that represents the skilled motion on the user-programming level, and this also works with the virtual control as part of ABB-RS (see Fig 5 with the bounding-box visualized as a semi-transparent surface to the left around the tool).

Note that the integration with KIF was done such that knowledge **about** the native properties of the engineering tool (ABB-RS) is kept on the KIF side, and code extending the ABB-RS with semantic interactive functions is dynamically generated and loaded into a JVM that interacts with native ABB-RS code (in C#) via self-descriptive communication interfaces ([19] with descriptions generated from KIF data). This way, production knowledge can be separated from engineering tool design, and reuse is also improved by having configuration, coordination and computation handled as separate concerns.

IV. Bringing Semantics into Robot Automation

The experiments verified the applicability of having declarative information as the basis for the task and motion descriptions including uncertainties and boundary conditions, thereby facilitating the reuse of manufacturing solutions also when products, processes and resources change (within those stated boundaries). The two main questions now are: Who will have the reasons to start using such an approach, and how can initial usage contribute to gradual improvement of the knowledge? The following proposes a possible solution based on integrating semantic level information into existing engineering practices and tools.

A. Skills

A key concept for the highly desired reuse of robot programs is that of so called *Skills*. At a first glance skills can be perceived as capabilities of devices or equipment, provided by abstract services that also can be compositions of lower level skills/capabilities [20]. However, for robots, we found that to be just a special case of a skill concept that better promotes reuse. The reason is that robots can be comparably inaccurate machines that (like humans but opposed to CNC machines) need to adapt their motions to the task and to the environment, and to also support interaction with humans the knowledge how that adaptation is done need to be explicit.

At the same time, that knowledge also needs to be explicit with respect to its dependencies and its properties (e.g., concerning its usage in the control systems for productive work). To that end, and based on the separation of concerns presented earlier, we bring forward the definitions of a Skill by putting it into the context of hierarchical control as configured coordination of actions/motions, which we found could be accomplished in a coherent way on all levels. Specifically:

Task:
It was determined that the task level should be based on symbolic information with compositional/additive entities. Both the desired (abstract) task and the actual (executable) task should be explicitly represented. Sharing, reuse and enhancement of knowledge can be facilitated by a knowledge-integration framework that interacts with the engineering and operator interaction in a mixed-initiative manner.

Skill:
The coordinating skill level should accept robot independent symbolic specifications (including uncertainties) of actions and motions. By means of computations (generic solvers and code generated from the symbolic level) and coordination (of real-time interaction with the motion level), the robot specific parts are synthesized such that motion specifications get reusable. The same applies to perception skills.

Motion:
The combination of proprietary motion control (optimized for the robot and key applications) and external control was shown to provide the desired combination of flexibility and performance. Self-descriptive interconnections with the skill and task levels promote reuse of sensor-based motions.

The constraint-based task specification methodology (iTaSC) [17] is a skill-suitable general framework that makes it easy to algorithmically incorporate multiple sensors, geometric uncertainties and to handle both redundant manipulators and redundant tasks. Note that in the solution approach b for the implementation for the SnapFit skill above, the geometries imply that multiple breakers no longer can be placed in *any-order* [21], but have to be placed sequentially with the supporting side of the fingers on the free side. Thus, low-level physical aspects relate to high-level symbolic information; this is about knowledge integration not to be confused with the common hierarchical control systems.

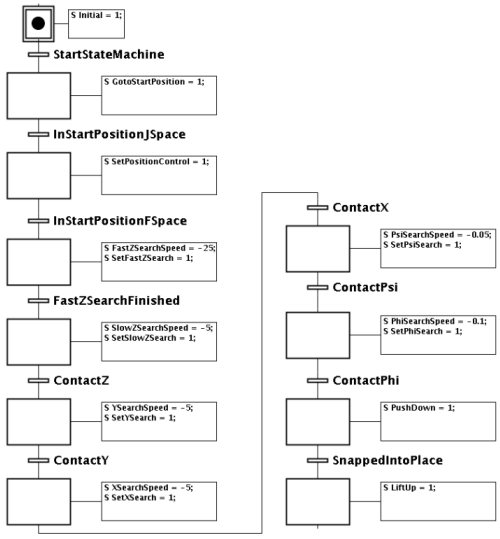

Fig. 7. The SnapFit state machine as an SFC

The SnapFit application example shows that not only high-level knowledge (such as an assembly graph) and reusable motion specifications (such as those based on iTaSC) are needed, but also the real-time motion control that combines microsecond latencies with semantic information about its interfaces, parameters and properties. The benefit of having coordination as a separate concern was experienced in the SnapFit example.

An initial implementation using Stateflow was made as it is an integrated Matlab tool. Due to the tight connection with computations as done in Matlab however, all variables need to have predefined sizes, which limited reuse. It is further on difficult to export the state machines in some appropriate format for storage in the KIF-server, and the separation between the state machine and the rest of the program is not very clear.

An alternative to implementing the state machine was done by separating it totally from the control computations, by replacing it with input and output ports communicating via a real-time network connection [19]. The coordination in terms of a state machine now being a separate concern, permitted an implementation based on SFC as in AutomationML (or JGrafChart, a Java based graphical SFC implementation). A screenshot of the used state machine for the SnapFit is shown in Fig 7.

Note that the SFC above expresses the nominal task in an executable manner and without all possible error situation. Since application errors are expected to be unexpected (otherwise they would be taken care of already), we need a compositional meta-level description that can be queried via the KIF upon operator interaction concerning the error. That is what our RDF-based implementation is facilitating; Fig 8 shows the KIF-level representation that is more suitable for error analysis.

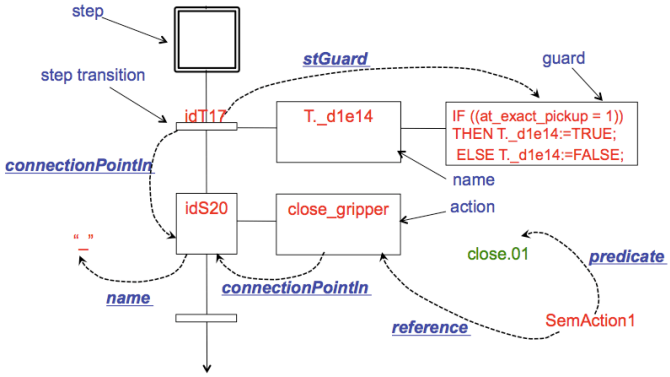

Fig. 8. Finding the transition and the guard using KIF/RDF predicates, which are represented semantically such that it provides meaning to an operator reporting an error.

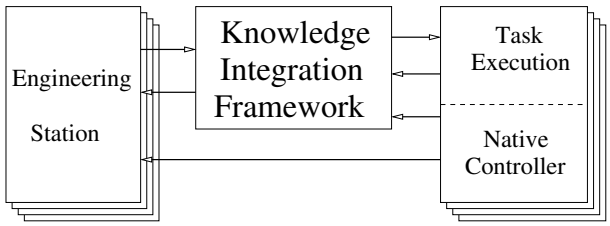

Fig. 9. Native engineering tools and controls coexisting with KIF.

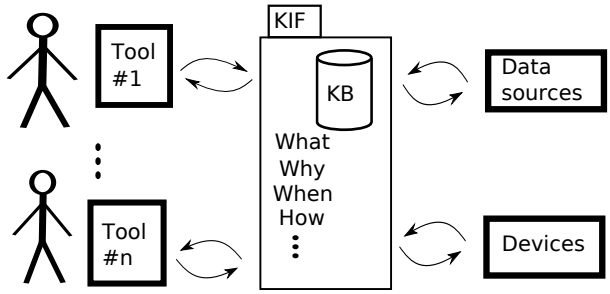

Fig. 10. The Knowledge Integration Framework (KIF) gradually improving the Knowledge Base (KB) by online connections to interactive users.

B. Knowledge-based robot programming

As indicated in the application examples, we keep the established robot programming tools, which in addition to the added KIF also maintains its connection to the native control system (Fig 9). That is perfectly in line with the configuration concern, but semantic tagging is needed for the native connection via KIF. The Task Execution part added contains iTaSC and extended control functions.

It is commonly believed that tool support for knowledge integration may facilitate more flexible robot programming and systems integration and thereby support the use of robots in one-off or short series manufacturing. The experiences reported above support that belief in principle, but the two main questions (reasons to start and gradual improvement) posed at the beginning of this section need to have good answers.

Starting to use knowledge-based features requires those features to be added on the back of the existing tools. The reason is that changing tool-sets for engineering is too costly for the user, and a technology provider has no business reasons for spend resources on vendor-independent solution. Hence, we would like the developed ABB-RS extensions to be freely available including source code. The new business opportunity associated with the values of obtained or learned knowledge we leave to industry. The presented way of obtaining initial knowledge (automatically from XML documents), and the seamless integration is ABB-RS, motivates the initial usage among the ABB customers.

Gradual improvement of knowledge (which similar to an induction proof in mathematics will lead to extensive use) is accomplished by extending the instance hierarchy of the engineering tool with popup menus for querying KIF, which typically includes the registration of available instances (such as work-pieces and selected type of robot) and obtaining suggestions (such as an alternative choice of gripper) that have been inferred from KIF and the available knowledge. The user always stays in charge for selecting the most appropriate option, and when doing so new knowledge is generated. Also when no useful answer can be suggested by the system, new knowledge (about what information that is missing) is generated, and so on as depicted in Fig reffig:data2info.

The resulting in-process mixed-initiative integration of KIF with the engineering tool can now be summarized as:

- Reasoning on the meta level will include also ”why” and ”how”, in addition to the standard ”what”. By linking and storing such data, knowledge is created.
- The connection with engineering tools have to be integrated behind the scenes such that complexity is hidden. Interactivity and online usage of KIF is important. If the knowledge base can initiate a dialog about a potential issue that the user is unaware of.
- The networked data interfaces facilitate spreading and keeping data consistent across an organization or between organizations.
  - Use reasoning and meta-level knowledge to determine if an action may be affected by, or possibly in conflict with, earlier decision made by another party.
  - Help engineers by informing them about previous work that may affect their work, but that they may not be aware of. That is applicable both to purely technical details and standards/safety aspects.
- Another possibility is to let local system integration contribute to knowledge transfer that has a value outside the organization.

A special feature of our implementation (not detailed here for brevity) is that only a small part is dependent on the specific version of the engineering tools, which means that also the knowledge-based support for reuse is reusable.

V. Conclusions

System integration still is a challenge in large-scale manufacturing and traditional automation using machines that perform according to normative models, but progress was made in standardization (e.g., of interfaces, PLC programs, and plant descriptions), modularization (e.g., by self-contained units including processing and standard network interfaces) and in engineering tools (supporting both vertical and horizontal integration). Skilled technicians are still required for the integration work, in which the closed-world assumption applies. The closed-world assumption is a simplification that promotes efficiency during development of specific tools and components, as carried out by a variety of stakeholders using state of the art methods such as model-driven engineering.

Motivated by the needs in small-scale manufacturing and by the aim for reuse of solutions and knowledge in applications such as assembly and machining, in which the deviations between actual and modeled behavior require human know-how, it was found that system integration also should work under open-world assumptions. Similar to Internet content, which is practically unlimited and not complying with common definitions except for some structure of meta-data and underlying protocols, the desire to share (some, not beforehand known) information and knowledge calls for a semantic approach. That is, data and information can get meaning in the context of the (always different) robot user. Therefore, the open-world assumption applies to the integration of future agile manufacturing systems, including robots performing sensor-based motions in semi-structured uncertain environments.

The practical development of systems should thus be based on techniques that already have proved successful on the market, and extensions should be designed to promote usage of third party solutions. On each level of control, there will typically be a closed and an open part, at large coinciding with native/proprietary subsystems and generic/open-source extension respectively. Access to, and ownership of, acquired or learned knowledge is a non-technical issue that remains to be investigated. The developed prototype systems verify the feasibility of the proposed concepts, which we claim pave the way for system integration with versatile robots in the upcoming open-world era. The described and prototyped technique with piggybacking a semantic-level triple-store to existing engineering tools is believed to be new.

Acknowledgment

The research leading to these results has received funding from the European Union's seventh framework program (FP7/2007-2013) under grant agreements #230902 (Rosetta: RObot control for Skilled ExecuTion of Tasks in natural interaction with humans; based on Autonomy, cumulative knowledge and learning) and #258769 (COMET: Plug-and-produce COmponents and METhods for adaptive control of industrial robots enabling cost effective, high precision manufacturing in factories of the future).

References

[1] AutomationML, "AutomationML specification. Part 1 – Architecture and general requirements," AutomationML Group, Tech. Rep., January 2009.

[2] R. Drath, Ed., *Datenaustausch in der Anlagenplanung mit AutomationML. Integration von CAEX, PLCopen, XML und COLLADA*. Springer, 2010.

[3] European Commission, "FP7 cooperation work programme: Information and Communication Technologies," ftp://ftp.cordis.europa.eu/pub/fp7/docs/wp/cooperation/ict/c_wp_201001_en.pdf, July 2010.

[4] N. Hawes, J. L. Wyatt, M. Sridharan, H. Jacobsson, R. Dearden, A. Sloman, and G.-J. Kruijff, *Architecture and Representations*, ser. Cognitive Systems Monographs. Springer Berlin Heidelberg, April 2010, vol. 8, pp. 51–93.

[5] N. Hawes, M. Zillich, and P. Jensfelt, *Lessons Learnt from Scenario-Based Integration*, ser. Cognitive Systems Monographs. Springer Berlin Heidelberg, April 2010, vol. 8, pp. 423–438.

[6] RDF, "RDF/XML syntax specification," http://www.w3.org/TR/rdf-syntax-grammar/, February 2004.

[7] E. G. da Silva, L. F. Pires, and M. van Sinderen, "Towards runtime discovery, selection and composition of semantic services," *Computer Communications*, vol. 34, no. 2, pp. 159–168, 2011.

[8] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist; Effective Modeling in RDFS and OWL*. Morgan Kaufmann, 2008.

[9] J. Malec, A. Nilsson, K. Nilsson, and S. Nowaczyk, "Knowledge-based reconfiguration of automation systems." IEEE Press, 2007, pp. 170–175. [Online]. Available: http://dx.doi.org/10.1109/COASE.2007.4341829

[10] A. Nilsson, R. Muradore, K. Nilsson, and P. Fiorini, "Ontology for Robotics: a Roadmap," in *ICAR: 2009 14th International Conference on Advanced Robotics, vols 1 and 2*. IEEE, 2009, pp. 291–296.

[11] J. Persson, A. Gallois, A. Björkelund, L. Hafdell, M. Haage, J. Malec, K. Nilsson, and P. Nugues, "A knowledge integration framework for robotics," in *ISR/ROBOTIK 2010 : proceedings for the joint conference of ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, 2010.

[12] M. Radestock and S. Eisenbach, "Coordination in evolving systems," in *Trends in Distributed Systems – CORBA and Beyond*. Springer-Verlag, 1996, pp. 162–176.

[13] D. Gelernter and N. Carriero, "Coordination languages and their significance," *Commun. ACM*, vol. 35, no. 2, pp. 97–107, 1992.

[14] J. L. M. Lastra and I. M. Delamer, "Semantic web services in factory automation: Fundamental insights and research roadmap," *IEEE Trans. Ind. Informatics*, vol. 2, pp. 1–11, 2006.

[15] I. Delamer and J. Lastra, "Loosely-coupled automation systems using device-level soa," in *Industrial Informatics, 2007 5th IEEE International Conference on*, vol. 2, June 2007, pp. 743 –748.

[16] L. Andrade, J. L. Fiadeiro, J. Gouveia, and G. Koutsoukos, "Separating computation, coordination and configuration," *Journal of Software Maintenance*, vol. 14, no. 5, pp. 353–369, 2002.

[17] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, 2007.

[18] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson, "Force controlled assembly of emergency stop button," in *Proceedings of ICRA*, 2011.

[19] "LabComm," https://www2.control.lth.se/trac/R012-Labcomm, 2011.

[20] I. M. Delamer and J. L. M. Lastra, "Service-oriented architecture for distributed publish/subscribe middleware in electronics production," *IEEE Trans. Ind. Informatics*, vol. 2, pp. 281–294, 2006.

[21] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "OWL-S: Semantic markup for web services," http://www.w3.org/Submission/OWL-S/, 2004, site accessed on January 19, 2010.