Knowledge-Based Instruction of Manipulation Tasks for Industrial Robotics[*]

Maj Stenmark[†] and Jacek Malec
Department of Computer Science, Lund University,
Box 118, 221 00 Lund, Sweden
email: [maj.stenmark,jacek.malec]@cs.lth.se

Tuesday 5th August, 2014

Abstract

When robots are working in dynamic environments, close to humans lacking extensive knowledge of robotics, there is a strong need to simplify the user interaction and make the system execute as autonomously as possible, as long as it is feasible. For industrial robots working side-by-side with humans in manufacturing industry, AI systems are necessary to lower the demand on programming time and system integration expertise. Only by building a system with appropriate knowledge and reasoning services can one simplify the robot programming sufficiently to meet those demands while still getting a robust and efficient task execution.

In this paper, we present a system we have realized that aims at fulfilling the above demands. The paper focuses on the knowledge put into ontologies created for robotic devices and manufacturing tasks, and presents examples of AI-related services that use the semantic descriptions of skills to help users instruct the robot adequately.

Keywords: knowledge representation, robot skill, industrial robotics ontology, assembly, service-oriented architecture

1 Introduction

The availability of efficient and cheap computing and storage hardware, together with intensive research on big data and appropriate processing algorithms on one hand, and on semantic web and reasoning algorithms on the other, make the existing results of artificial intelligence studies attractive in many application areas.

[*]Accepted to Robotics and Computer Integrated Manufacturing Journal, DOI: 10.1016/j.rcim.2014.07.004

[†]corresponding author, tel. +46 46 2221667

The pace of adoption of the knowledge-based paradigm depends on the complexity of the domain, but also on the economic models used and the perspective taken by the leading actors. It may be quite well illustrated by opposing the service robotics area (mostly research-oriented, mostly publicly funded, using open source solutions, acting in non-standardized and not-yet-legally-codified domain) with industrial robotics (application-oriented, privately funded, using normally closed software, enforcing repeatability and reliability of the solutions in legally hard-controlled setting).

When robots are working in dynamic environments, close to humans lacking extensive knowledge of robot programming, there is a strong need to simplify the user interaction and make the system execute as autonomously as possible (but only as long as it is reasonable). This also motivates the integration of AI techniques into robotics systems. For industrial robots working side-by-side with humans in manufacturing industry, AI-based systems are necessary to lower the programming cost with respect to required time and expertise. We believe that only by building a system with appropriate knowledge and reasoning services, we can simplify the robot programming sufficiently to meet those demands and still get a robust and efficient task execution.

In this paper, we present a knowledge-based system aimed at fulfilling the above demands The paper is focusing on the knowledge and ontologies we have created for the robotized manufacturing domain and is presenting examples of AI-related services that are using the semantic descriptions of skills to help the user instruct the robot adequately. In particular, the adopted semantic approach allows to treat skills as compositional pieces of declarative, portable and directly applicable knowledge on robotized manufacturing.

The paper is organized as follows: first we introduce the robot skill. Then we describe the system architecture. Next section introduces our robot skill ontology and other relevant ontologies available in the knowledge base, as well as some services provided by the system. Next we introduce the interface towards the user, i.e. the Engineering System, and briefly describe the program execution environment exploiting the knowledge in a non-trivial way. Then we describe related research. We conclude by suggesting future work.

2 Robot Skills

Our approach is anchored on the concept of a *robot skill.* As it may be understood in many different ways, both by humans and machines, it needs to be properly defined and made usable in the context of our domain of applications. The presentation in this section adopts a historical perspective, showing how our understanding of skills pushed forward the capacities of systems we have created.

Our earliest deployed system has been developed in the context of the EU project SIARAS: *Skill-Based Inspection and Assembly for Reconfigurable Automation Systems.* Its main goal was to build fundamentals of an intelligent system, named *the skill server*, capable of supporting automatic and semiautomatic reconfiguration of existing manufacturing processes. Even though the concept of skill was central, we have assumed *devices* as the origin of our ontology. Our idea then has been that skills are just capabilities of devices: without them no (manufacturing) skill can exist. A device can offer one or more skills and a skill may be offered by one or more devices. We have not introduced any granularity of such distinction; all the skills were, in a sense, primitive, and corresponded to operators as understood by AI planning systems (models of operations on the world, described using preconditions, postconditions, sometimes together with maintenance conditions). This understanding laid ground to the development of a robotic skill ontology, `siaras.owl`, that has been used to verify the configurability of particular tasks given current robotic cell program expressed as a (linear) sequential function chart (SFC). This approach has been proven valid, but the ontology grew quite fast and became problematic to maintain, given dozens of robots with a number of variants each, thus multiplying the number of devices. The details of SIARAS approach have been described in [17]. Figures 1 and 2 illustrate the basic hierarchy of skills available in the `siaras.owl` ontology.

The dual hierarchy, that of devices, has been illustrated on Figures 3 and 4, while Figure 13 shows some of the properties that can be attributed to devices.

The deficiencies of the siaras ontology: atomicity of skills and devices, fixed parameterizations and scalability issues, have led us to reconsider the idea. This time devices did not play a central role any longer, but rather skills have been put in the center. In the ROSETTA project[1] the definition of skills has been based on the so-called production (PPR) triangle: *product*, *process*, *resources* [10] (see Figure 6). The *workpieces* being manufactured are maintained in the product-centered view. The manufacturing itself (i.e., the process) is described using concepts corresponding to different levels of abstraction, namely *tasks*, *steps*, and *actions*. Finally, the resources are materialized in *devices* (capable of sensing or manufacturing). The central notion of *skill* links all three views and is one of the founding elements of the representation.

In case of a robot-based production system, skills may be defined as *coordination of parametrized motions*. This coordination may happen on several levels, both sequencing (expressed, e.g., via a finite state machine or a similar formalism), configuring (via appropriate parametrization of motion) and adapting (by sensor estimation). On top of this approach, based in our case on *feature frame* concept [11], we have built a set of reasoning methods related to task-level description, like e.g., task planning. The details are presented in the following sections.

[1]RObot control for Skilled ExecuTion of Tasks in natural interaction with humans; based on Autonomy, cumulative knowledge and learning, EU FP7 project No. 230902, `http://www.fp7rosetta.org/`.

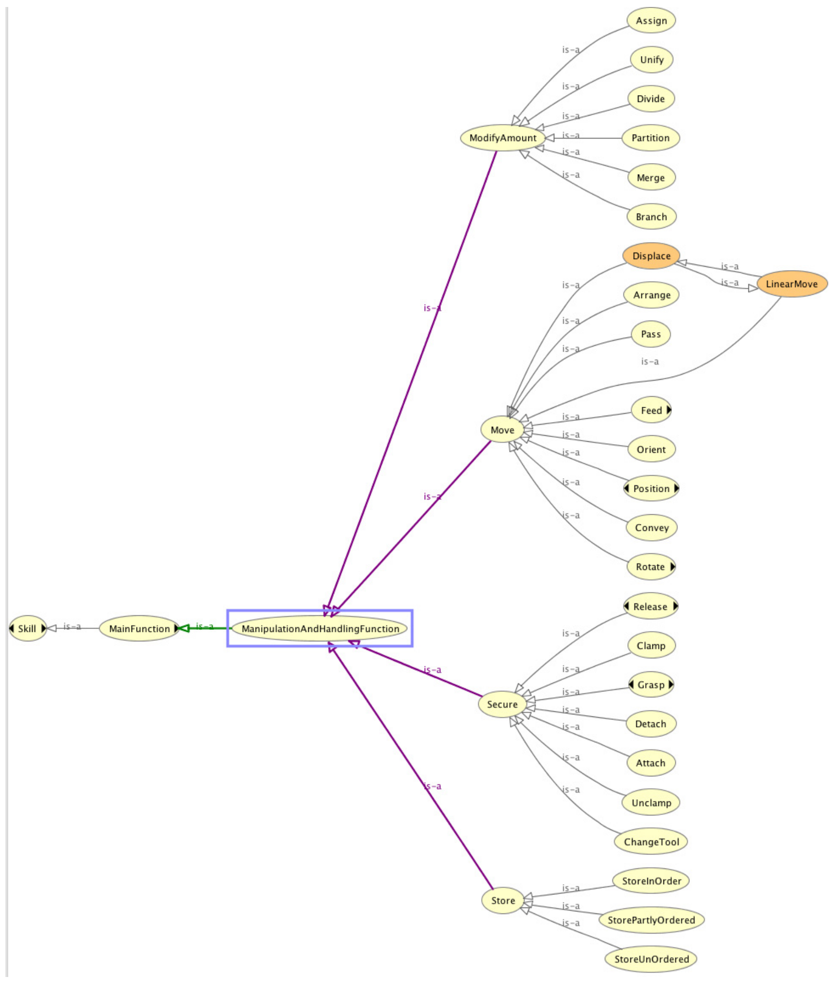


Figure 1: Manipulation and handling skills, as defined by SIARAS ontology.

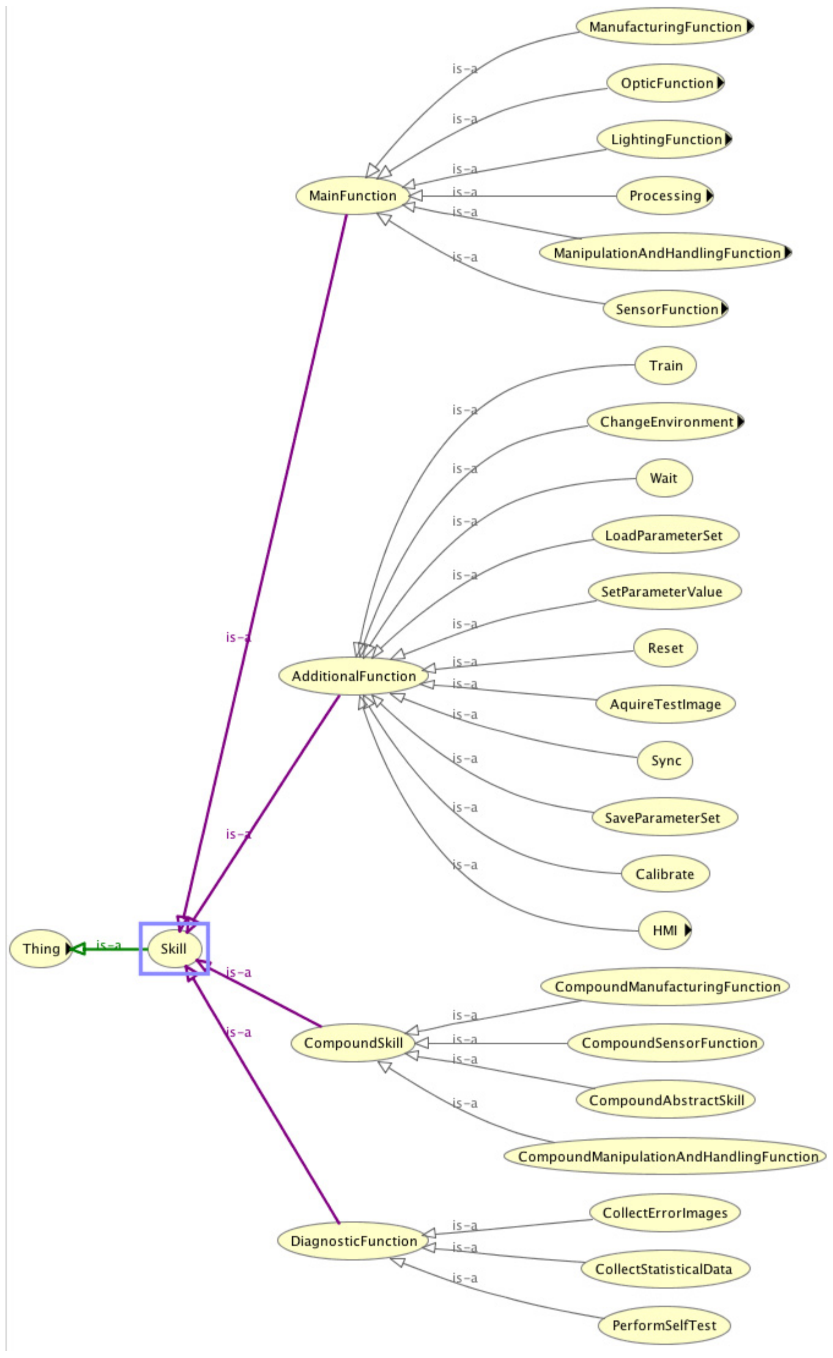


Figure 2: Top skill classification, as defined by SIARAS ontology.

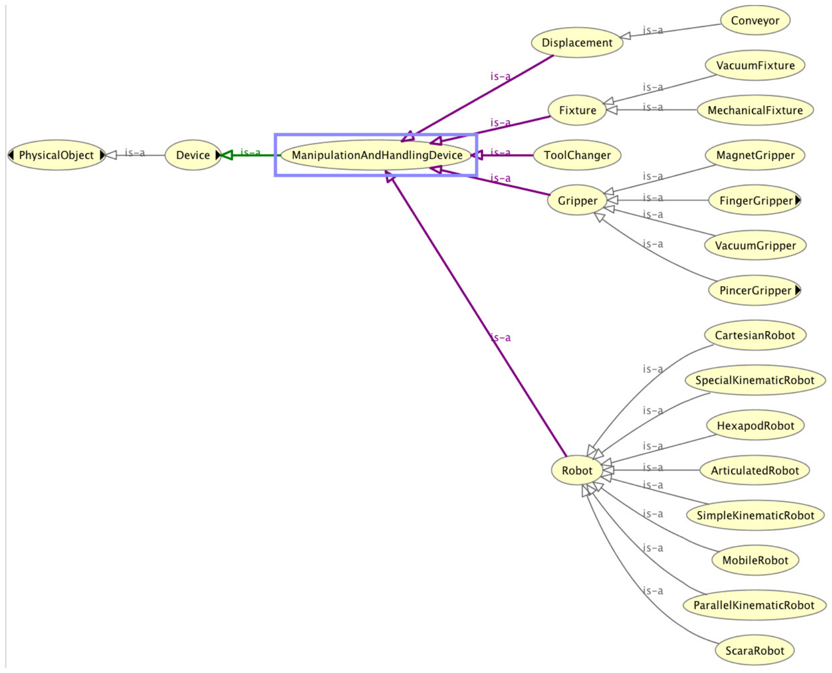


Figure 3: Manipulation and handling devices, as defined by SIARAS ontology.

3 Architecture

The generic setup describing the intended usage of our approach is illustrated in Fig. 7. The system architecture is very roughly depicted in Fig. 8. The Knowledge Integration Framework (KIF) is a server that contains data repositories and ontologies. It provides computing and reasoning services. There are two main types of clients of the KIF server, the Engineering System, which is a robot programming environment, and the robot task execution system.

The task execution system is a layer built on top of the native robot controller. Given the task, the execution system generates the run-time code files utilizing online code generation (see Sec. 5), then compiles and executes the code.

The Engineering System uses the ontologies provided by KIF to model the workspace objects and downloads skills and tasks from the skill libraries. Similarly, new objects and skills can be added to the knowledge base by the Engineering System. Skills that are created using classical programming tools such as various state machine editors (like, e.g., JGrafchart[2] [34]), can be parsed, automatically annotated with semantic data and stored in the skill libraries.

The services, described later in the paper, are mainly used by the Engineering System to program, plan and schedule the tasks.

[2]`http://www.control.lth.se/grafchart/`

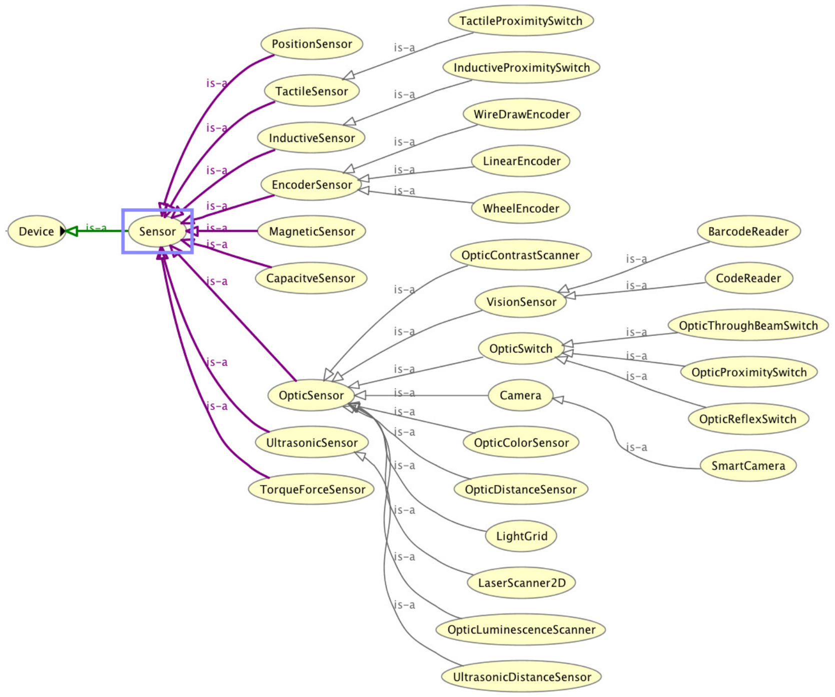
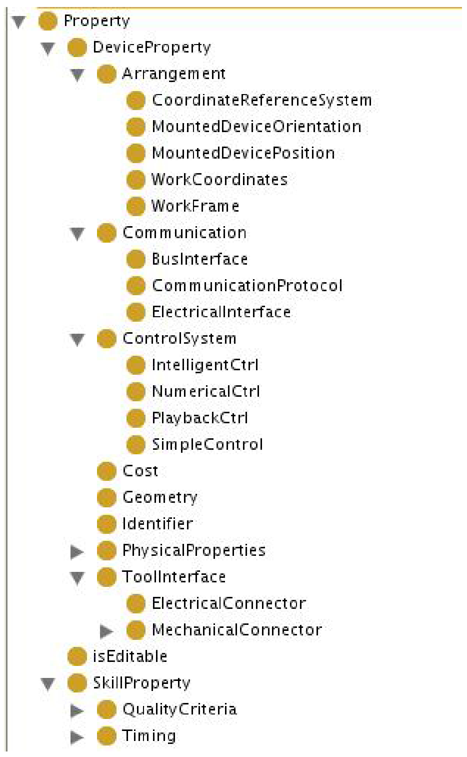


Figure 4: Sensor devices, as defined by SIARAS ontology.



```
▼ Property
  ▼ DeviceProperty
    ▼ Arrangement
        CoordinateReferenceSystem
        MountedDeviceOrientation
        MountedDevicePosition
        WorkCoordinates
        WorkFrame
    ▼ Communication
        BusInterface
        CommunicationProtocol
        ElectricalInterface
    ▼ ControlSystem
        IntelligentCtrl
        NumericalCtrl
        PlaybackCtrl
        SimpleControl
      Cost
      Geometry
      Identifier
    ▶ PhysicalProperties
    ▼ ToolInterface
        ElectricalConnector
      ▶ MechanicalConnector
    isEditable
  ▼ SkillProperty
    ▶ QualityCriteria
    ▶ Timing
```

Figure 5: Device properties, as defined by SIARAS ontology.

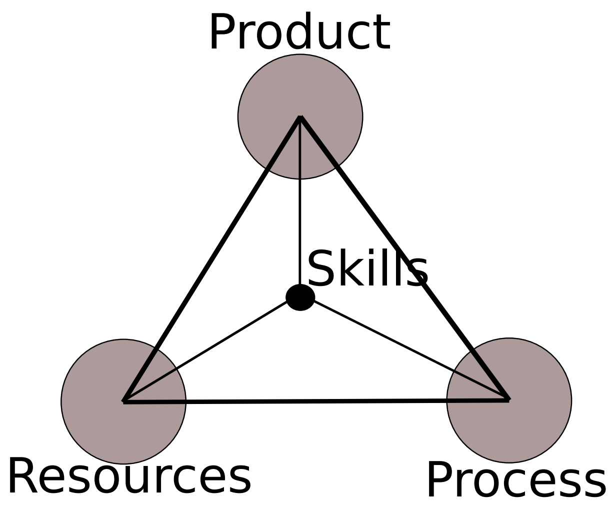


Figure 6: The PPR model, with skills as common coordinating points for the three views.

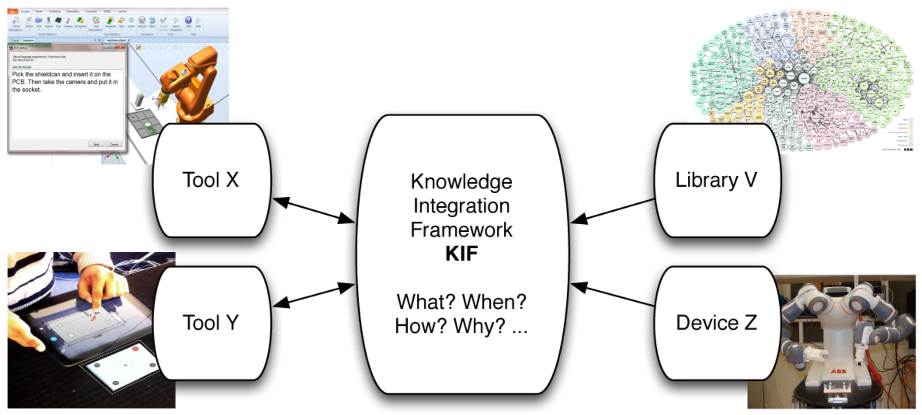

Pick the shieldcan and insert it on the PCB. Then take the camera and put it in the socket.

Figure 7: 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.

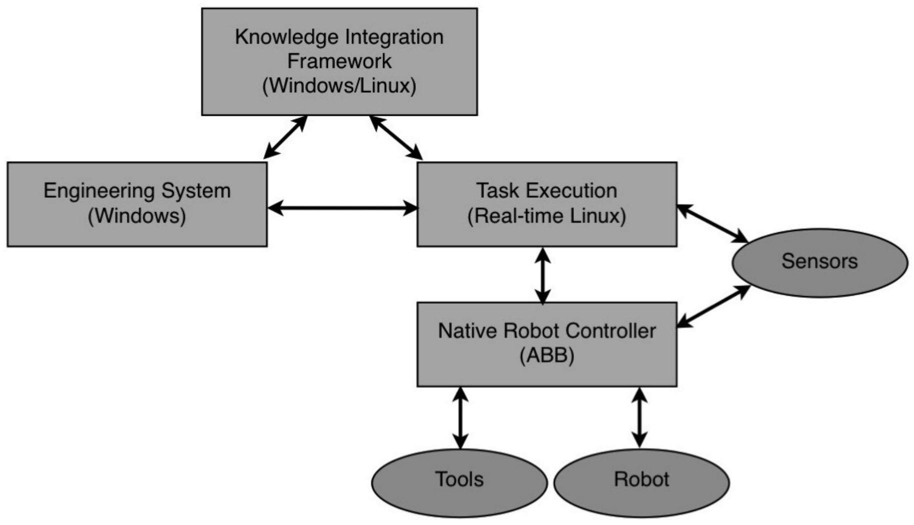

Knowledge Integration Framework (Windows/Linux)
Engineering System (Windows)
Task Execution (Real-time Linux)
Sensors
Native Robot Controller (ABB)
Tools
Robot

Figure 8: 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.

4 Knowledge Integration Framework

The Knowledge Integration Framework, KIF[3], is a module containing a set of robotics ontologies, a set of dynamic data repositories and hosting a number of services provided for the stored knowledge and data. Its main storage structure is a Sesame[4] triple store and a set of services stored in Apache Tomcat[5] servlet container[6].

The ontologies we use in our system come from several sources and are used for different purposes. The main, core ontology, `rosetta.owl`, is a continuous development aimed at creating a generic ontology for industrial robotics. Its origins is the FP6 EU project SIARAS described earlier in Sec. 2. It has been further modified within the FP6 EU project RoSta (Robot Standards and reference architectures, `http://www.robot-standards.eu/`, [25]). Within the FP7 EU Rosetta project this ontology has been extended, refactored and made available online on the public KIF ontology server `http://kif.cs.lth.se/ontologies/rosetta.owl`. However, this is just the first of a set of ontologies available on KIF and useful for reasoning about robotic tasks.

The ontology hierarchy is depicted in Fig. 9, where arrows denote ontology import operations. We used extensively the QUDT ontologies and vocabularies (Quantities, Units, Dimensions and Types, initiated by NASA and available at `http://www.qudt.org`) in order to express physical units and dimensions. This ontology has been slightly modified to suit the needs of our reasoner. However, as QUDT ontologies led to inconsistencies, we have introduced the possibility to base the quantities, units and dimensions on the alternative OM ontology[7] [29].

The core Rosetta ontology (as its predecessors) is focusing mostly on robotic devices and skills. According to it, every device can offer one or more skills, and every skill is offered by one or more devices. Production processes are divided into tasks (which may be considered specifications), each realized by some skill (implementation). Skills are compositional items: there are primitive skills (non-divisible) and compound ones. Skills may be executed in parallel, if the hardware resources and constraints allow it.

On top of the core ontology we have created a number of "pluggable" ontologies, serving several purposes.

Frames The `frames.owl` ontology deals with feature frames and object frames of physical objects, normally workpieces involved in a task. In particular, the feature frames are related to geometrical locations and therefore the representation of location is of major importance here. The constraints among feature

[3]We realize the name coincidence with Knowledge Interchange Format [14], but as this name has been used for more than six years by now, we have decided to keep it.

[4]`http://www.openrdf.org`

[5]`http://tomcat.apache.org`

[6]Technically speaking, the triple store is also a servlet.

[7]`http://www.wurvoc.org/vocabularies/om-1.6/`

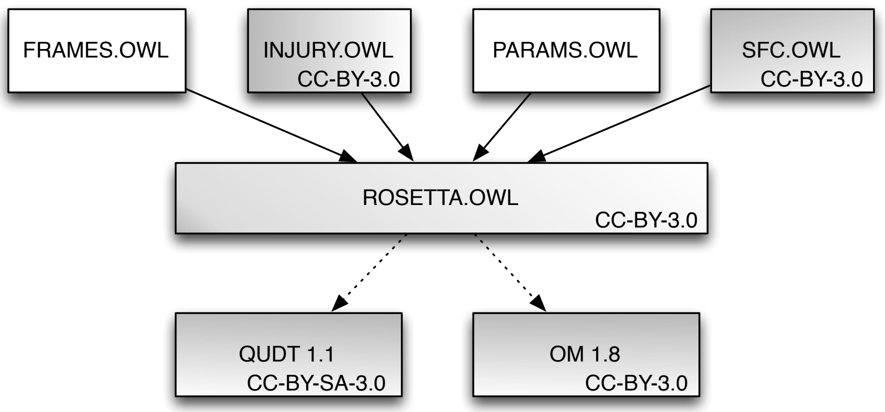


Figure 9: The KIF ontologies used by the Rosetta project. In case an ontology is openly available, the type of license is quoted.

frames are expressed using *kinematic chains* [11], also introduced by this ontology.

Injury The `injury.owl` ontology deals with the levels of injury risks when humans and robots cooperate, or at least share common space. The ontology specifies the possible injury kinds, while the associated data, either extracted from earlier work [12], or gathered during the Rosetta project [22], are provided as the upper limit values that may be used in computations of injury risks or of evasive trajectories for a robot.

Params Each skill may be parameterized in a number of ways, depending on the granularity level of control, available information or the demands posed on the skill. In order to provide knowledge about skill parameterization for knowledge services (like, e.g., task consistency checking), the `params.owl` ontology describes skills and their mandatory and optional parameters, their units and contraints.

SFC The `sfc.owl` ontology characterizes various behavior representations using variants of executable state machines (Sequential Function Charts are one of them; the others included are OpenPLC, Statecharts, rFSMs and IML). It also contains the semantic description of several graph-based representations of assembly, like assembly graphs, constraint graphs or task graphs [21], that may also be considered to be behavior specification, although at a rather high level of abstraction.

This solution illustrates two important principles of *compositionality* and *incrementality*: every non-trivial knowledge base needs to be composable out of simpler elements, possible to be created by a single designer or team without the need to align it with all the other elements. The alignment, or conflict resolution (e.g., inconsistency), should be performed (semi-)automatically, after plugging the element into the system. So, every "top" ontology should only be forced to adhere to QUDT (or OM) and ROSETTA ontologies, possibly neglecting other elements existing in parallel.

The incrementality principle ensures that every "top" ontology should be amenable to incremental change without the risk of breaking the whole system. Thus, changes to e.g., Params ontology should not affect the consistency and utility of e.g., SFC ontology. On the other hand, one can imagine situations where changes in one module (e.g., introduction of a new constraint type between feature frames, described in `frames.owl`) might facilitate improvements in another (e.g., easier specification of parameters for a given skill, described in `params.owl`).

Besides storing the ontologies, the triple store of KIF provides also a dynamic semantic storage used by Engineering System to update, modify and reload scene graphs and task definitions. Depending on the kind of repository used, some reasoning support may be provided for the storage functionality. More advanced reasoning, and a generic storage of arbitrary kind of data, is provided by KIF services, described below.

5 Knowledge-Based Services

The knowledge base provides storage and reasoning services to its clients. The most basic service it offers is access to libraries with objects and skills, where the user can upload and download object descriptions and task specifications. Some of them are stored with semantic annotations, as triples, e.g., workpieces, scene graphs or skill definitions. Others are stored as uniform chunks of data without semantically visible structure (e.g. RAPID programs or COLLADA files), although other tools may access and meaningfully manipulate them for various purposes.

The services are mostly user-oriented, providing programming aid, and can be used step-by-step to create a workspace and then to refine a task sequence from a high-level specification to low level code. The workspace is created by adding a robot, tools, sensors and workpieces to the scene, giving the object properties relevant values and defining relations between objects (see Sec. 6).

The user specifies a task using the workpieces and their relations. On the highest level, the task is represented by an assembly graph [21]. An example assembly graph of a cell phone is shown in Fig. 10. The assembly graph is normally a tree (not necessarily binary). The leaves are the original workpieces which are joined into subassemblies represented by parent nodes and the full assembly is represented by the root. Each subassembly can be annotated by more information, such as geometrical relations between the objects, or what type of joining mechanism to use (e.g., glueing, snapping, screwing). The tree imposes a partial order on the operations, where child assemblies have to be

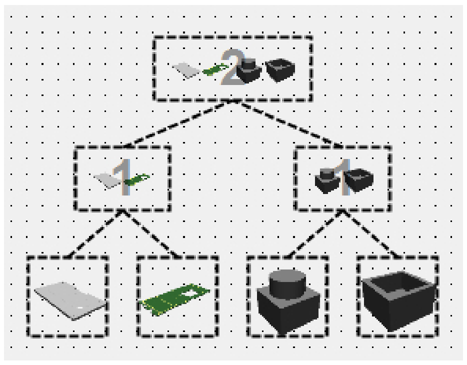
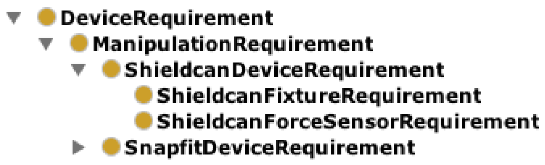


Figure 10: An assembly graph for a partial assembly of a cell phone. A metal plate, a *shield can*, is pressed onto a printed circuit board (PCB) and a cell phone camera is inserted into a socket. The camera socket is then fastened on the PCB.



- ▼ ● DeviceRequirement
  - ▼ ● ManipulationRequirement
    - ▼ ● ShieldcanDeviceRequirement
      - ● ShieldcanFixtureRequirement
      - ● ShieldcanForceSensorRequirement
    - ▶ ● SnapfitDeviceRequirement

Figure 11: The device requirements of the skill ShieldCanInsertion are modelled in an ontology. The **ManipulationRequirement** which several skills inherit from, is that a gripper has to be mounted on the robot. The **ShieldcanFixtur-eRequirement** and the **ShieldcanForceSensorRequirement** list that there must exist a fixture and a force sensor that have to be vertically aligned (not shown in the picture).

carried out first. When going from the task specification given by the assembly graph to an executable program, the task has to be sequentialized. Depending on the robot, or on the number of collaborating robots, the sequence can be realized in several ways, hence, an assembly graph specification can be shared by several robot systems, even though the sequences realizing it will differ.

KIF provides a planning service that transforms an assembly graph to a sequence of operations using preconditions and postconditions of the skills. Initially the service verifies the device requirements of a skill. Fig 11 displays the device requirements of an implementation of the skill that inserts a shield can onto a printed circuit board (PCB). This skill has only three device requirements: a mounted tool (which is a manipulation requirement), a fixture and a force sensor, which (though it is not displayed in the figure) must be aligned vertically. When planning the sequence, the planner adds actions that fulfill the preconditions (see the example in Fig. 12), such as moving objects into place.

However, the sequence can also be created directly by the user, either man-

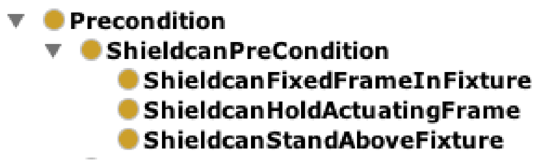
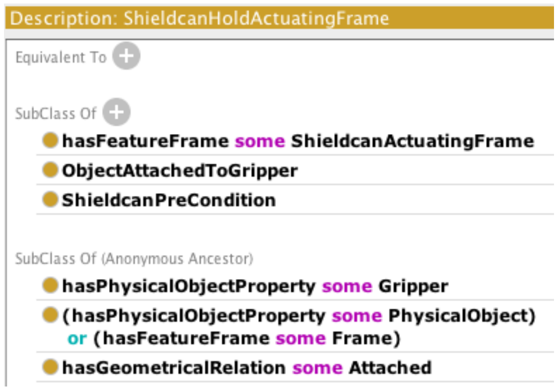

▼ Precondition
▼ ShieldcanPreCondition
ShieldcanFixedFrameInFixture
ShieldcanHoldActuatingFrame
ShieldcanStandAboveFixture

Figure 12: There are three preconditions to the ShieldCanInsertion skill. The skill has two *feature frames* (relative coordinate frames) as input parameters, where one is a reference object frame and the other is attached to the object in the gripper, i.e., an actuating frame. The first precondition is that the object with the reference frame has to be on the fixture. Secondly, the object with the actuating frame should be attached to the gripper, see Fig. 13, and finally, the position of the actuating object should be above the fixture. Imprecise geometrical relations such as “Above” are given concrete values by the Engineering System.


Description: ShieldcanHoldActuatingFrame

Equivalent To

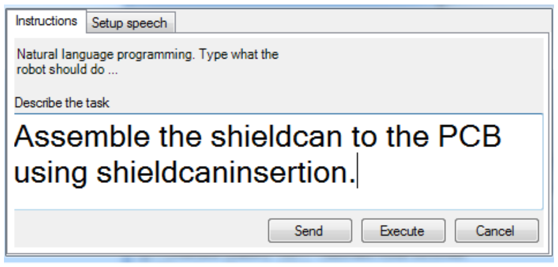
SubClass Of
hasFeatureFrame some ShieldcanActuatingFrame
ObjectAttachedToGripper
ShieldcanPreCondition

SubClass Of (Anonymous Ancestor)
hasPhysicalObjectProperty some Gripper
(hasPhysicalObjectProperty some PhysicalObject)
or (hasFeatureFrame some Frame)
hasGeometricalRelation some Attached

Figure 13: The ontology description of the precondition **ShieldcanHoldActuatingFrame** which is a subclass of **ObjectAttachedToGripper**.

ually or by using a natural language instruction interface. Later, the same planner can be used to verify that the sequence fulfills the preconditions of each action.

The natural language interface is described in more detail elsewhere [32, 31]. The user either dictates or types English instructions into a text field (see an example in Fig 14). The input text is sent to a natural language service on KIF where the sentences are parsed into predicates (verbs) and their corresponding arguments. Each verb has several different *senses* depending on the context and meaning, e.g., the predicate *take* in *take off the shoes* has sense *take.01*, but in the sentence *Take on the competition* it has sense *take.09*. *The shoes* and *(on) the competition* are arguments to the predicates. Each sense has a number of predefined arguments for, e.g., the actor doing the deed, the object being manipulated, the source or the destination. These arguments are labelled *A0*,



Natural language programming. Type what the robot should do ...

Describe the task

Assemble the shieldcan to the PCB using shieldcaninsertion.

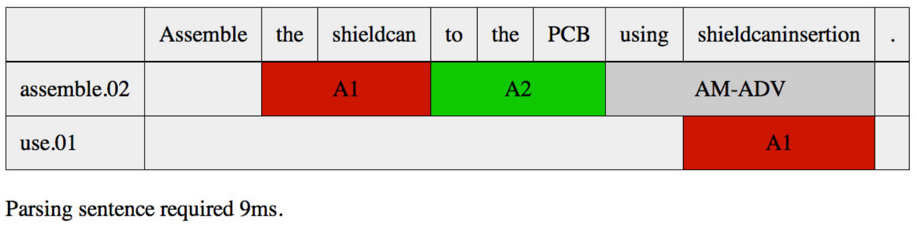
Figure 14: The user can describe the task using English sentences.

A1, etc. Both the sense of the verbs and the arguments are determined using statistical methods described in [4].

The natural language service outputs a preliminary form of program statements derived from the sentences However, the matching to actions and objects existing in the world is done in the Engineering System. In the simplest form a program statement contains an action (the predicate) and a few arguments (objects). The action is then mapped to a robot program template while the arguments are mapped to the physical objects in the workspace, using their names and types. Actions described this way can be picking, placing, moving and locating objects. More complicated program structures can be expressed using conditions that have to be maintained during the action or for stopping it, as in the sentence *Search in the x-direction until contact while keeping 5 N in the z-direction.* The example sentence *Assemble the shieldcan to the PCB using ShieldCanInsertion* given in Fig 14 has a skill, *ShieldCanInsertion*, as argument to *use* (which in turn is a nested argument to *assemble*, see bottom of Fig 15). *Use* is not mapped to a robot action, but rather prompts a search for a corresponding skill in the KIF libraries. The skill is instantiated with the arguments as parameters or, when no matching parameter can be found, with default values. For example, the *ShieldCanInsertion* is described in the ontology with an actuated object and a fixated object, which are mapped to *A1 -the shieldcan* and *A2 - the PCB*, respectively.

These programs can be further edited or directly executed on a physical robot or in the virtual environment of the Engineering System.

There exists also a scheduling service that helps the user to assign actions to a system with limited resources. The current implementation of the service is based on list-scheduling. The manipulation skills require different end effectors, e.g., for gripping and for screwing. By adding a tool changer to the cell, the robot can change end effectors during the task. The time it takes to change tools is added as penalty on the priority of the actions. When there are multiple arms, one arm can of course change a tool while waiting for the other arm to finish its operation during a two-arm manipulation skill. A typical input to the service can be to schedule a partially ordered task on a two-armed robot with three tools and one force sensor. Each action lists its estimated time and the resource requirements, required tool(s) and resources. Given the estimated time



	Assemble	the	shieldcan	to	the	PCB	using	shieldcaninsertion	.
assemble.02		A1		A2			AM-ADV		
use.01								A1	

Parsing sentence required 9ms.

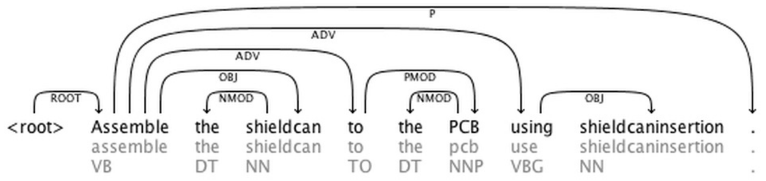


Figure 15: The result given by the parser of the sentence from Fig. 14. At the top, each line displays a found predicate with its arguments. Assemble was evaluated to *assemble.02* with the arguments *the shieldcan (A1)*, *to the PCB (A2)* and a manner *using shieldcaninsertion.* The bottom of the picture displays the dependency graph (actually a tree). The arrows point, beginning from the root of the sentence, from parents to children. Each arrow is labelled with the grammatical function of the child. Under each word the corresponding part-of-speech tag (determiner - DT; noun - NN, etc) can be found.

to change tools and the number of cycles, the service will output a suggested schedule that minimizes the total time.

The last service named here is a code generation service used by the task execution system. It is described below in Sec. 7.

6 Engineering System

The Engineering System is a high level programming interface implemented as a plugin to the programming and simulation IDE *ABB RobotStudio*[8], shown in Fig. 16. When creating a station, objects such as the robot, workpieces, sensors, trays and fixtures, can be manually generated in the station or downloaded from KIF together with the corresponding ontologies.

A physical object is characterized by its local coordinate frames, the *object frame* and a number of relative coordinate frames called *feature frames*, see Fig. 17. Geometrical constraints are expressed as relations between feature frames, and may be visualized as in Fig. 18.

An example program sequence is shown in Fig. 19. The program has a nested hierarchy, where steps (such as *pick* or *place*) may contain atomic motions and

[8]`http://new.abb.com/products/robotics/robotstudio`

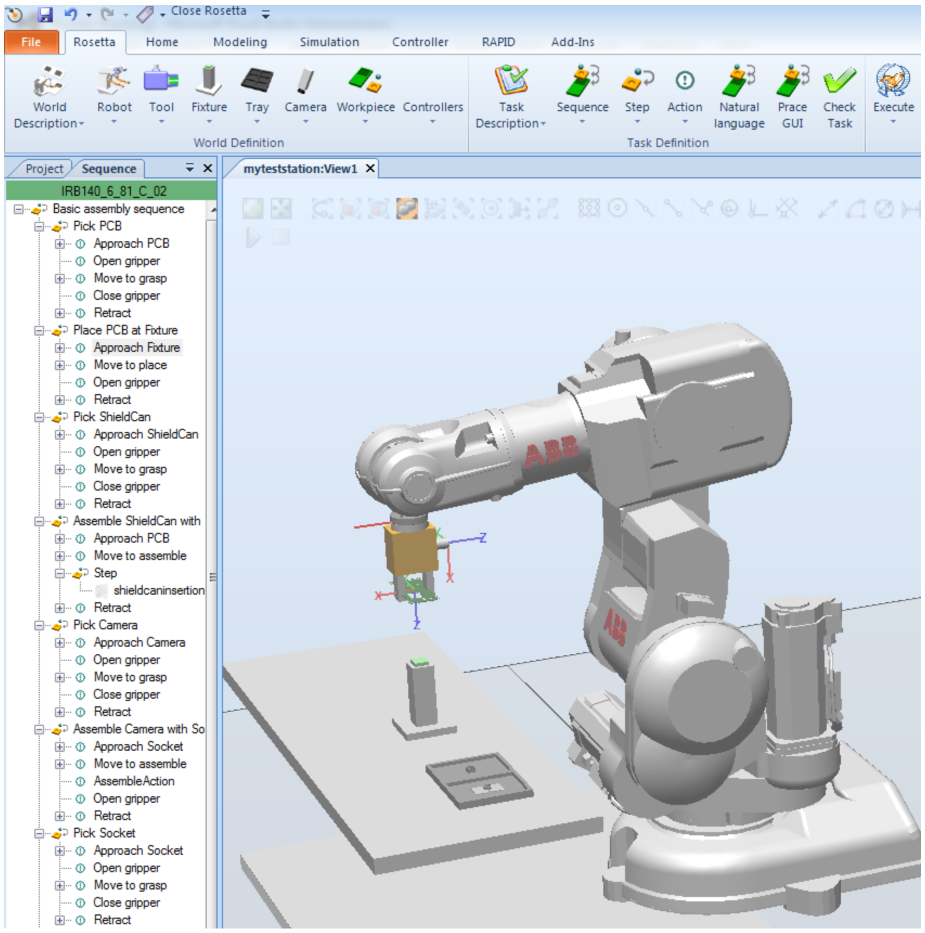


Figure 16: The engineering system is a plug-in to the programming environment ABB RobotStudio.

gripper actions.

7 Execution

The sequence from Fig. 19 is sent to the execution system, which in turn calls the code generation service that returns a complete state machine (serialized in an XML file), which is visualized, compiled and executed using JGrafchart tool [34]. It creates a task state machine, where each state is either a call to primitive functions on the robot, or a nested skill. Fig 20 shows a small part of a generated state machine. Each skill is either retrieved from KIF and instantiated with the new parameters, or generated from scratch by creating a closed kinematic chain for a given robot and the objects. The vendor-specific code is executed using the

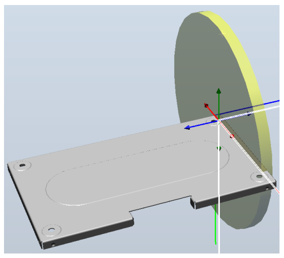


Figure 17: A *feature frame*.

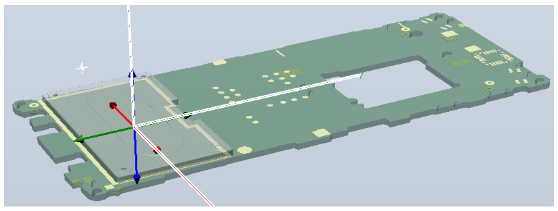


Figure 18: A geometrical relation between two objects.

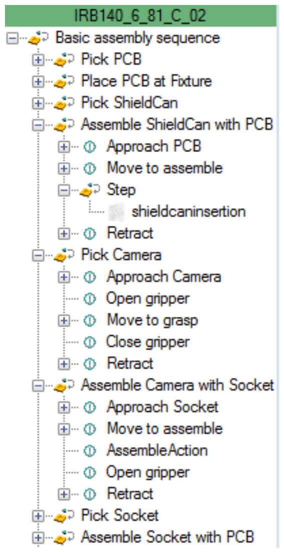
native robot controller, while the more complex sensor-based skills are executed using an external control system [7] and the state machine switches between these two controllers when necessary.

To guarantee a safe execution, the injury risk for different velocities is evaluated using the data stored in KIF and the final robot speed is appropriately adjusted.

8 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 [8]. 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 [27]. This way of thought continued, leading to such efforts as “Naive physics” by Patrick Hayes (see [8]) or “Physics for Robots” [30]. 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 [15] have also advanced much and may be used nowadays for cases of substantial complexity, 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



```
IRB140_6_81_C_02
Basic assembly sequence
    Pick PCB
    Place PCB at Fixture
    Pick ShieldCan
    Assemble ShieldCan with PCB
        Approach PCB
        Move to assemble
        Step
            shieldcaninsertion
        Retract
    Pick Camera
        Approach Camera
        Open gripper
        Move to grasp
        Close gripper
        Retract
    Assemble Camera with Socket
        Approach Socket
        Move to assemble
        AssembleAction
        Open gripper
        Retract
    Pick Socket
    Assemble Socket with PCB
```

Figure 19: An example sequence of the cell phone assembly. First, the PCB is moved to the fixture. Then, the shieldcan is picked and inserted on the PCB using a sensor-based skill called *shieldcaninsertion*. The cell phone camera is assembled with the socket, and then the socket is inserted on the PCB.

robots is prevalent nowadays [3], 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 the 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 [19] 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[9] 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 [16], 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 [17].

Robotics systems are usually build from a number of distributed heterogenous hardware and software components that have to seamlessly interact during

[9]`www.robot-standards.org`

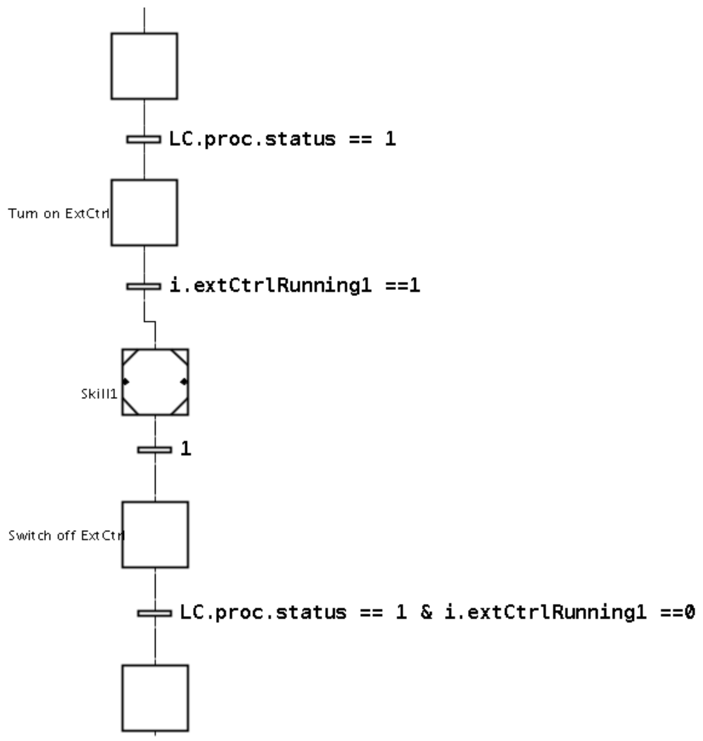


Figure 20: Each box is a state in the task state machine. The state called *Skill 1* with marked corners is a nested state machine containing a (dynamically generated) sensor-based skill. Before and after the sensor-based skill the external controller is started and turned off, respectively.

execution phase. In order to simplify configuration, communication and hide the complexity of the system, as well as to promote portability and modularity, there exist several frameworks for robotics middleware (see comprehensive surveys[13, 24]). Module functionality can be provided as *nodes* in the ROS[10] environment, or as standardized components in RT-components [1], where the modules can provide blackbox-type computations with well-specified interfaces.

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 [5], but the way of finding appropriate composition varies heavily, from manual sequencing in many workflows, via AI-influenced task planning [15], hybrid automata development tools [16], Statecharts [18] and Sequential Function Charts (SFCs) [19], iTaSC specifications [11], to development of monolithic programs in concrete robot programming languages, like e.g. ABB RAPID.

There have been several attempts to codify and standardize the vocabulary

[10]`www.ros.org`

of robotics. There exists an old ISO standard 8373 requiring however a major revision to suit the demands of contemporary robotics. IEEE Robotics and Automation Society is leading some work towards standardisation of robotic ontologies. In particular, there are first drafts of robotic core ontology [9], although not as developed as the ROSETTA ontology described in this paper. Regarding industrial robotics, the work on kitting ontologies, originated at NIST [2], may be considered an early attempt to address the problem.

In the area of service robotics there are several systems exploiting the knowledge-based approach, and relying on an underlying ontology, like KnowRob [33] (based on the generic OpenCyC ontology [23]), used in RoboHow project[11] or several participants in the RoboEarth project[12] [35]. However, they do not attempt to standardize the domain, as the variance of tasks and skills in the service robotics is very large. On the other hand, the KnowRob ontology became a de-facto standard used in several experimental robot systems.

9 Conclusions

We have shown a generic knowledge-based system architecture and its possible use in industrial robotic systems. In particular, we have employed the approach for representing and realizing force-controlled tasks realized by one- and two-armed ABB robots in an industrial setting. The presented generic ontologies are either novel, or a derivative of our earlier research. The use of semantic tools and explicit knowledge in industrial robotics is in its early stage, with only a few other published examples [2]. The ideas have been experimentally verified and work well in the currently ongoing EU-projects PRACE[13] and SMErobotics[14]. The implemented system is just a proof of concept, and systems that derive from this work must undergo usability, security and performance testing, before they might be considered ready for industrial practice. But already now it can be stressed that the knowledge-based approach allowed us to create composable representations of non-trivial assembly skills, shown to be reusable among different models of ABB robots, but also portable to other vendors and control architectures (like the one reported in [20] and running on a Kuka LWR4 robot).

The already ongoing continuation of the work presented above involves integration of a heterogenous system consisting of a mobile robot platform (Rob@Work) running a ROS-based control system, and a real-time-enabled ABB-manipulator running the ABB-specific control software, so that the two parts can operate seamlessly together as an integrated, knowledge-based, productive robotic system. This work includes deploying knowledge-based services in the context of chosen robotic middleware.

Future work involves contribution to the IEEE standardization efforts, and aligning and sharing robotic ontologies with other research groups. An on-line

[11]`http://robohow.eu`
[12]`http://roboearth.org/`
[13]`http://prace-fp7.eu/`
[14]`http://www.smerobotics.org/`

documentation of the core ROSETTA ontology is also expected. The number of knowledge-based services should be extended with, e.g., online reasoning during execution, geometrical reasoning and integrated path planning and optimization. We are also verifying this approach in other domains of manufacturing, like wood-working and machining, expecting to extend the ontologies appropriately.

We have found out during the work described in this paper that skills are much more than just a potential to execute coordinated motions. This line of thought has been already present in [26], where business aspects of skills have been pointed to. We plan to explore this topic in the nearest future.

Acknowledgments

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).

The work described in this paper has been done in tight collaboration with many people from the project consortia. The authors are indebted for many fruitful discussions.

An early version of this paper has been presented during the 12th Scandinavian Conference on Artificial Intelligence, Aalborg, Denmark, October 2013.

References

[1] Noriaki Ando, Takashi Suehiro, Kosei Kitagaki, Tetsuo Kotoku and Woo-Keun Yoon. RT-Component Object Model in RT-Middleware - Distributed Component Middleware for RT (Robot Technology). Proc. of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), pp. 457-462, 2005.

[2] Stephen Balakirsky, Zeid Kootbally, Craig Schlenoff, Thomas Kramer, and Satyandra Gupta, *An Industrial Robotic Knowledge Representation for Kit Building Applications*, Proc. 2012 IEEE/RSJ IROS, October 7-12, 2012, Vilamoura, Algarve, Portugal.

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

[4] 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, 2010.

[5] 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.

[6] 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.

[7] Anders Blomdell, Isolde Dressler, Klas Nilsson and Anders Robertsson, *Flexible Application Development and High-performance Motion Control Based on External Sensing and Reconfiguration of ABB Industrial Robot Controllers.* In Proc. of ICRA 2010, pp. 62-66, Anchorage, USA, 2010.

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

[9] 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 IROS, Tokyo, Japan, 2013.

[10] A.F. Cutting-Decelle, R.I.M. Young, J.J. Michel, R. Grangeland, J. Le Cardinal, and J.P. Bourey. ISO 15531 MANDATE: A Product-process-resource based Approach for Managing Modularity in Production Management, *Concurrent Engineering*, vol. 15, 2007.

[11] 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.

[12] Deutsche Gezetzliche Unfallversicherung. BG/BGIA risk assessment recommendations according to machinery directive, Design of workplaces with collaborative robots. Report no. U001/2009e, revised 2011. `http://www.dguv.de/bgia`

[13] Ayssam Elkady, and Tarek Sobh. Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography. Journal of Robotics, Volume 2012, Article ID 959013, doi:10.1155/2012/959013

[14] Michael R. Genesereth and Richard E. Fikes. Knowledge Interchange Format, Version 3.0 Stanford University Logic Group, Technical Report Logic-92-1, 1992.

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

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

[17] 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.

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

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

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

[21] 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.

[22] Björn Matthias, Susanne Oberer-Treitz and Hao Ding, *Collision Testing for Human-Robot Collaboration.* In: Safety in Human-Robot Coexistence and Interaction: How Can Standardization and Research Benefit from each other? IEEE Int. Conf. Intelligent Robots and Systems (IROS) 2012.

[23] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira. An Introduction to the Syntax and Content of Cyc. 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering.

[24] Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. Middleware for Robotics, A Survey. Proc. of the IEEE Int. Conf. on Robotics, Automation and Mechatronics (RAM2008), pp. 736–742, 2008

[25] 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.

[26] Klas Nilsson, Elin Anna Topp, Jacek Malec and Il-Hong Suh. Enabling Reuse of Robot Tasks and Capabilities by Business-Related Skills Grounded in Natural Language. In ICAS 2013, 9th Int. Conference on Autonomic and Autonomous Systems, Lisbon, Portugal, 2013.

[27] Nils J. Nilsson. Shakey the robot. Technical Note 323, SRI International, Menlo Park, CA, USA, 1984.

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

[29] Hajo Rijgersberg, Mark van Assem and Jan Top. Ontology of Units of Measure and Related Concepts. Semantic Web Journal, vol. 4, no. 1, pp. 3–13, IOS Press, 2013.

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

[31] Maj Stenmark and Jacek Malec, *Describing constraint-based assembly tasks in unstructured natural language*, To appear in Proc. of the 19th IFAC World Congress, Cape Town, 2014.

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

[33] Moritz Tenorth and Michael Beetz. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. International Journal of Robotics Research, volume 32, 2013.

[34] Alfred Theorin. Adapting Grafchart for Industrial Automation. Licentiate Thesis, Dept. of Automatic Control, Lund University, 2013.

[35] 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.