MASTER'S THESIS | LUND UNIVERSITY 2015

A Module-Based Skill Ontology for Industrial Robots

Ludwig Jacobsson

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2015-48

A Module-Based Skill Ontology for Industrial Robots

Ludwig Jacobsson
knd09lja@student.lth.se

June 26, 2015

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Jacek Malec, jacek.malec@cs.lth.se

Examiner: Klas Nilsson, klas@cs.lth.se@cs.lth.se

Abstract

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

Keywords: robot skill, ontology, modularization, industrial robots

Contents

Chapter 1
Introduction

Ever since their arrival in the second half of the 20th century, industrial robots have been well established in the manufacturing sector. Basic tasks such as welding, painting, stacking and sorting proved to be ideal applications. At workstations where success is characterized by repetitiveness, speed, and precision, robots successfully replaced humans. However, development seem to have hit to a road block as the task complexity increase.

The current state of robotics is costly for enterprises in terms of skill and time. Programming is limited to skilled engineers and is often done using proprietary software. Programming at low abstraction levels often involve trajectory-planning and manual setting of abstract parameters. Profitable investments is thereby limited to large production series, hence large enterprises. Moving the skill set of industrial robots to the next level of complexity has proven difficult since the current state of robot programming is limited to relatively low-level instructions.

By introducing reusable skills, capable of representing combinations of primitive robot instructions, the goal is to allow for easier programming and more complex tasks. As a result, experts in the area of robot programming can apply themselves to the development of skills and primitive instructions while experts in the task domain can instruct the robots at the workplace. In the end, the time needed for instructing robots will be shortened and their skill set greatly increase.

Further, the intent is to let this lead the way for future concepts within the industrial robotics field. Such as skill libraries where users can access and download skills to their robots. It will also present a great opportunity for reasoners to automatically pick a suitable skill for the task at hand, thereby further automating the process.

1.1 Problem definition

During our life and daily routines communication is key to everything we do. For this to function seamlessly and without misinterpretations a well defined vocabulary agreed upon beforehand is a necessity. Its is often of great importance that the right information is exchanged or we will likely see negative consequences. This communication is not limited to individuals/pairs or groups of humans, but also to human-computer and computer-computer interactions. This is where the area of knowledge representation and semantics come in. The introduction of knowledge such as skills and the need for human-robot or robot-robot communication in a domain requires increasing complexity of the knowledge representation. With the need for a well-defined model capturing this information, ontologies have come to play a fundamental role.

"An ontology is a formal, explicit specification of a shared conceptualisation" [1]

With increasing complexity, quantity and size of domain, the struggle to manage the knowledge-base becomes apparent. With a large monolithic knowledge-base, merely the intent to find existing knowledge may become non-trivial. Extending, removing and modifying even more so. As a result we find an increased effort in the research community in finding techniques to help organizing the knowledge. Modularization introduces such a technique. By letting a large knowledge-base be represented by a set of modules, ie. smaller knowledge-bases we have achieved less complexity, less quantity and a smaller domain. The reader might identify this as an analog to the long-established complexity management technique "divide and conquer" commonly used in computer science areas such as algorithm theory.

The problem with large monolithic knowledge-bases grows with usage. Without proper structure and modularization, implementation or modification of knowledge may become unclear, which will bring the former structured ontology into a state of perplexity.

1.2 Project goal

The problems described in the section above have been identified in the knowledge-base of the high-level task description tool-chain at LTH. The project goal can be summarized as:

- present an ontological model for skill representation.
- present a modularized ontology structure for the model.
- present a proof of concept for accessing the skills.

Developing a model for skill representation will result in declarative, portable and reusable skills. The skills are aimed at instructing of industrial robots in manufacturing tasks. The model is achieved by analysing currently implemented concepts regarding skill representation at LTH. The model is realized as an ontology, a common concept in knowledge representation.

Modularizing the model into separate ontologies will result in more reusable knowledge and easier management. By limiting the scope of each ontology and developing a hierarchy for the ontology interactions both skills and other knowledge are more reusable. This will be achieved by isolating domains of knowledge with strong internal coherence and loose coupling to other domains.

To lead the way for a future skill library we intend to present a proof of concept for accessing skills from the knowledge base. The developed software will present an opportunity to check the correctness of the developed model and provide some starting ground for the future development of a skill library.

Project scope

Premises for the work regarding ontology development presented in this thesis are limited to the ontologies described as *plug-in* ontologies, i.e., all ontologies except the core ontology ROSETTA. ROSETTA is the result of several EU Projects and has been properly developed and evaluated. While there may be room for maintenance it is not in the scope of this thesis.

1.3 Related work

The European FP6 RoSta Project, "Robotics Standards and Reference Architectures" successfully terminated in 2009 with a positive result regarding the use of ontologies in robotics research and development. Below is an excerpt from the presented results concerning the creation of ontologies.

"[...] A proof of concept of a robotic ontology scheme and its exemplary use in industrial and service robotics domains demonstrated the benefits of using ontologies in robotics research and developments." [2]

In [3] a case is made for the use of ontologies in the robotics field in general and in the industrial robotics field in particular. Initial definitions of compound tasks, devices, and skills are presented and a proof of concept for using ontologies as a design tool in robotics is introduced. Multiple critical points necessary for success is identified, one of them being the need for a distributed database storing both the functional ontologies and the device ontologies. This research was supported by the RoSta Project.

The European FP6 SIARAS project, running 2005-2008, focused on knowledge-based automatic reconfiguration of automation systems. Example of the results from LTH can be seen in [4]. Concluding the project the knowledge-base of the system was determined as underdeveloped, resulting in the launch of the ROSETTA project, 2008-2012. ROSETTA focused on simplifying interactions between the user and the robotized automation system, and thereby demanded knowledge-based solutions.

[5] describes one of the insights from the ROSETTA project as the need for a modularization of the ontology used for supporting the connected subsystems. A preliminary solution is presented with a call for further research and development in the area. [5] also presents a vision of a future app store and shared knowledge-bases but identifies a need for knowledge bases, services and solution distribution systems to be in place, while these in turn need knowledge and solutions to be distributed to get started.

1.4 Thesis outline

The general outline of the thesis is as follows. The theory in Chapter 2 is applied to the domain described in Chapter 3 and the resulting model is presented in Chapter 4. This result is then discussed in Chapter 5 and conclusions on the results are drawn in Chapter 6.

The general approach of this thesis is studying methodologies for ontology creation, strategies to ontology modularization and finally the current tool-chain at LTH. Chapter 2 presents the relevant theory on knowledge representation, ontology model development and ontology modularization. Chapter 3 presents the study of the research domain of high level instructions for industrial robotics at LTH. In fact, what we are developing is a model for representing the knowledge describing a skill for industrial robots at LTH. To achieve this we need to know how skills for robots are used in the context of the LTH tool-chain. In Chapter 4 we present the results of the thesis. The chapter is divided into three parts, the developed model, the modularized ontology structure and finally the proof of concept development. The model is introduced by describing the domain and concepts of the skill representation we want our model to represent. This can be loosely seen as the requirements of the model. Then the class hierarchy and properties of the developed model is presented. The second part regarding the modularization describes the scopes and domains of the ontologies and their hierarchy. The third part concerning the proof of concept briefly describes the techniques used in development and its results. Chapter 5 provides discussion of the results and finally, in Chapter 6 we present some conclusions on the results.

Chapter 2
Approach

This chapter begins with a general introduction on knowledge representation and knowledge for machines. This will provide the general idea behind the work in this thesis. This is followed by theory and methodologies for ontology development and modularization.

2.1 Knowledge representation

"Scientia potentia est" is a Latin aphorism often claimed to mean "knowledge is power". Despite its disputed origins its implications are clear, the one who possesses knowledge also possesses power. But what is knowledge? We will not address this question since we are unlikely to find a conclusive answer.[1]. But since this is a deep philosophical subject we feel empty-handed without some ideas and thoughts before we dive into the technical aspects.

Knowledge has a strong relation to *symbolic representations* [6]. Even before we had any kind of language there was likely symbolic representations of who is who, what is what, and skills we might have acquired. Another important part of knowledge and symbolic representations is communication [7]. *Communicating knowledge* is likely a major reason behind the development of spoken and written languages. The letters and words of our languages are nothing without their symbolism and *semantics*. Imagine reading a book without knowing the words or ordering food without knowing the language. You realize that when communicating knowledge an agreed upon vocabulary is a necessity. Establishing this vocabulary is a matter of formal naming and defining the types, properties, and interrelationships of the entities in the vocabulary.

Much of the semantics and symbolism come from associations. If we have learned something once, we can recognise patterns from our previous knowledge and apply these to new situations. When describing stored knowledge, for example that lamp on our office desk, we can provide a good model for that knowledge using a graph. We know it provides

[1] 42

light, the light is directional, its color is white, it is from IKEA, it is not very heavy, it is a bit dusty and so on. We can immediately connect this information to other things that provides light, are white, are not very heavy or are bit dusty. By remembering these new objects we can find new descriptors and find additional objects, and so on. It is not far-fetched to describe desk lamp as a subclass of lamps with some common attributes and descriptors. We can also specify the relations between the lamp and its descriptors: white is a color of the lamp, IKEA is a manufacturer, provides light is a functionality etc. It is like building a graph network with nodes, edges and properties to represent what we know and what these things mean to us.

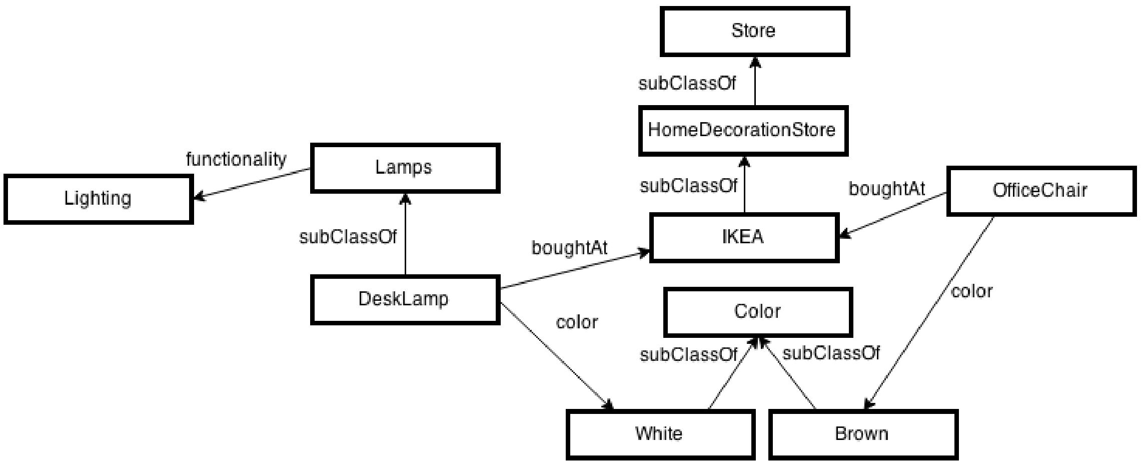


Figure 2.1: Knowledge about our desk lamp.

2.2 Knowledge for machines

The reasons for storing and representing knowledge for machines are practically the same as why knowledge and knowledge communication for humans is important. In [8] the author begins by presenting a few.

- To share common understanding of the structure of information among people or software agents,
- To enable reuse of domain knowledge,
- To make domain assumptions explicit,
- To separate domain knowledge from the operational knowledge,
- To analyze domain knowledge.

The graph network model described in Section 2.1 and Figure 2.1 is a gross simplification of how humans relate to knowledge, still the model presents an attempt at formalizing the structure of knowledge. A formalization that can be implemented for machines to enable computational reasoning about knowledge. By using the same principle of nodes, edges and properties we can construct a knowledge model and use it to build knowledge-based systems. The structure **subject**, **predicate** and **object** can store relations between objects much in the same way we described the `desk lamp` above, see example in Figure 2.2.

Subject	Predicate	Object
DeskLamp	subClassOf	Lamp
Lamp	functionality	Lightning
DeskLamp	color	White
DeskLamp	boughtAt	IKEA
IKEA	subClassOf	HomeDecorationStore
OfficeChair	subClassOf	Chair
OfficeChair	color	Brown
...	...	...

Figure 2.2: A machine knowledge representation using subject - predicate - object relations.

We will make use of this representation of knowledge later when we wish to display examples of contents and usage of the implemented ontologies.

2.3 Definitions, vocabulary and terms

Many subjects and terms mentioned and discussed in this thesis are foreign to the uninitiated. The academic fields of knowledge engineering, high-level robot instructions, or the like, have a large lingo with tasks, skills, ontologies, etc. While these terms may mean little to the uninitiated, many of them have an array of different definitions, each with passionate followers with strong bias. For the sake of this thesis, and the discussions herein, we will discuss the meaning and definitions of the most important terms.

Ontology

The definition of ontology applied in this thesis is software oriented; the term has a etymological definition concerning the philosophical study of the nature of being, becoming, existence, or reality.

As mentioned in the introduction and hinted upon in Section 2.1 an ontology is a description of a limited domain containing a formal naming of the entities in the domain and a description of their relations and properties. The knowledge presented in Table 2.2 can be thought of as a formal naming and definition of types, properties, and interrelations of entities in the domain of a desk lamp. We would then call this a *desk lamp ontology*, although, its usefulness and structure can certainly be questioned. An ontology often describes a limited domain within certain area, for example anatomical structures in animals [9] or gene products [10].

In [8] the authors present a functional definition of an ontology, a definition that clarifies the different parts and opens up for development. They divide the definition into parts.

Classes to represent concepts, such as our lamp.

Subclasses to provide a more specific representation, such as the lamp - desk lamp relation.

Slots to represent various features and attributes of the classes, such as color, functionality or bought at.

Facets to provide restrictions on the slots. For example we can restrict the value of the slot 'color' to subclasses of the class 'color'.

When using the term knowledge base in this thesis we refer to the entire set of ontologies and their combined knowledge.

Task

The work to be done by an robot is defined as a task. The task denotes the assignment and the undertaking of the robot with a final goal. For example the task might be for the robot to assemble a stop button box or to build a table.

Skill

The skill definition might be the most debated among these terms with as many approaches as there are researchers. The description here is merely pedagogic as a relief for uninitiated the reader.

The task is achieved by the utilization of skills. A skill denotes a competence and a "know-how" of the robot. One or several skills may be applied to finalize a task. In the "build a table" example, skills might be "drill" and "put-leg-in-hole". The skills are thought of as reusable items so that the skills of the table manufacturing may well be reused in any constellation for the completion of other tasks.

2.4 Model Development

Most research on ontology engineering presents some kind of step by step guide called a methodology. The majority of the methodologies are aimed at creating ontologies from scratch. This is not analog to the situation in this thesis. Several skills are already present in the knowledge base. These skills are represented in multiple different ways and do not adhere to any standard. Our goal of a skill model is achieved by combining the features and concepts of the existing knowledge with new ideas.

A successful model of a concept is is easy to work with and easy to grasp. When introducing a new person to the model it should feel natural. On the same note, when implementing new concepts it should be clear where this knowledge belongs. All in all, what we are describing is an unambiguous model of a domain. To achieve this we need some supporting structure during the development process, which is presented below.

2.4.1 Ontology development

With increasing demand for ontologies, ontology engineering and design is an active area of research. Multiple popular methodologies and work flows for ontology development have been developed in the last decades [8, 11]. The sections below are the result of studying these methodologies and can be interpreted as steps in the process of developing an ontology. During the development process we have followed these steps, they will also be used when presenting the results in Chapter 4

A. Knowledge acquisition

In the ontology development methodology and ontology life-cycle presented in [11], *knowledge acquisition* is referred to as one of the first steps of ontology development. The domain knowledge of the developer is fundamental to ontology development. Without it, making the necessary design decisions will prove a hard task. This first step is carried out by a thorough domain study using all available sources of knowledge such as literature, the actual system and dialogues with researchers.

B. Determine the domain and the scope of the ontology

Limiting the domain of the ontology is crucial to further development [8]. Subjects and information related to but not relevant for the domain is easily included during development without a well defined scope of the ontology. With the help of basic questions the domain and scope of the ontology can be made clear.

The following questions are suggested as guidelines for scope limitation. By referring to, and if the need arises, altering them during development of the ontology it may become less diluted and more faithful to the intended domain.

*What is the **domain** that the ontology will cover?*
*What is the **application** of the ontology?*
*Who is the **user** and **maintainer** of the ontology?*

The example from Section 2.1 is a typical ontology with the need of a scope. A limitation would for example be for the products of IKEA. More general knowledge on lamps or home decoration stores would be the scope of other ontologies.

C. Define the classes and the hierarchy

Defining the classes and their properties, are two closely intertwined steps that take place intermittently during the development process [8]. They are also the most important steps since their results are what constitutes the ontology. During the process of defining the classes and the class hierarchy the important concepts of the domain and their relations are determined. These concepts are then turned into the classes of our model.

For example we can turn the concept *lamp* into the class `lamp` and decide it has a parental relation to the concept and class `desk lamp`. This will give us classes and hierarchy for the lamps in our example ontology in Section 2.1.

There are multiple strategies for developing the ontology class hierarchy [12]: **top-down**, **bottom-up** and **combination**.

The **top-down** approach starts with the most general concepts in the domain and then proceeds to the mote specialized concepts. This approach would begin with the definition of `store` and move on to `IKEA`, `lamp` and `desk lamp`.

A **bottom-up** development does the inverse. It begins with the most specific concepts and then move up in the hierarchy to more general concepts. `desk lamp`, `lamp`, `IKEA`, `store`.

A **Combination** process is a combination of the top-down and the bottom-up approaches. We begin with the definition of the most prominent concepts and specialize and generalize appropriately.

D. Define the properties of the classes

To further describe our domain we proceed by defining the properties of the classes in our hierarchy [8]. This will provide our concepts with internal structure and enrich the knowledge in our ontology. We call these properties *slots*. Subclasses inherit the slots of their superclass/parent, this enables us to define more general properties as we climb the class hierarchy. A slot should be defined for the most general class that has that property.

In our lamp ontology we would have slots such as *color* and *functionality*. Following the generality guidelines, the functionality slot should be defined as *lighting* for the superclass `lamp` instead of `desk lamp`.

E. Consider reusing external ontologies

Given the time researchers and developers spend on ontology engineering in a wide range of domains, considering existing external ontologies might prove time well spent [8]. Even if the ontology might need some alignment to work with the internal ontologies, using external ontologies may have several advantages. Incorporation of existing, hopefully well, researched knowledge, compatibility with other applications and possibly conforming to developed standards are the most prominent advantages. With some effort the hard work of others may be incorporated into the ontology, a feat not so easily achieved in other fields. In recent years a large effort has put into the development of standardized ontologies, as mentioned in Section 1.3. Incorporating these standards will hopefully grant a longer lifetime of the developed knowledge.

Regarding our lamp ontology, one or several of the ontologies suggested in the section above regarding limiting the scope may exists and prove useful, such as home decoration stores or general knowledge on lamps.

2.5 Modularizing the ontology structure

With a model for skill representation we have achieved part of our goal. The other issue we are attempting is the modularization of the ontologies in the knowledge base. **Modularization** is the process of extracting modules of knowledge from the knowledge base. The modules are pieces of knowledge with strong internal coherence but relatively loose coupling. The pieces are specified in separate ontologies that represent smaller domains of knowledge. Separately, they can successful represent their respective domains. Combined, the ontology modules are used as building blocks to encompass the entire knowledge base.

When discussing the ontologies we will relate to their level, this is a way of describing their reusability. A general ontology will be more reusable and therefore described as a higher level ontology, whereas a more specific ontology has less reusability and is a lower level ontology. A word on the usage of *ontology* and *ontology module*: each of the ontologies constitutes a module, therefore the meaning of the words is in effect the same.

2.5.1 Module development

Extracting useful modules from a knowledge base is a task with several possible approaches and results. Suggestions range from algorithmically extracting modules using the description logic of the ontology [13] to using natural viewpoints or base categories to identify smaller domains with high coherence. Below we will present some way leading points to successful modularization. These points were utilized in the process of extracting modules in this thesis.

A. Goal driven

The extraction of modules should to be based on the goal [14] of the modularization. Different goals might call for different modules. If the modularization is performed for scalability or performance, modules that limit the domain would be preferred. If the goal is for example maintenance, modules with natural domains would perhaps be the way to go. It is important to identify the different needs and goals of the modularization and adapt the strategy based on this.

B. Semantic-driven strategy

In [14] the authors conclude the importance of a module to "make sense". Each module is an ontology representing a subset of the original domain. Semantic interpretation of knowledge relies on the human expertise of the persons in charge of maintaining and extending existing knowledge base. This includes the allocation of the implemented knowledge to a specific module. In order for this to work the modules needs to make sense from a human perspective. By identifying natural viewpoints [15] we can modularize the domain knowledge into relatively small ontologies centered around a base category or a certain viewpoint. Making sure the modules makes sense will provide modules with less ontological commitment and is therefore more reusable than the entire domain ontology.

Using this strategy our IKEA ontology may be modularized into a module for example representing each furniture type.

Chapter 3

General overview of LTH tool-chain

In this section we will present a general overview of the tool-chain used at LTH for instructing industrial robots using high-level task descriptions. In parallel to a literature study, valuable discussions with Jacek Malec and Maj Stenmark have provided great insight to the current tool-chain and the ontological efforts at LTH, resulting in the contents of this chapter.

The approach at LTH is of the type *bottom up*, the system is developed from the low level robot control mechanisms up to the high-level task descriptions. Here we will present a brief overview based on the work presented in [16, 17, 18, 19].

The general focus is on manipulation tasks for small part assembly, where the tasks are solved by performing a sequence of skills. Skills are basically semantically annotated state machines and at the lowest level specified by simple motion constraints. To enable reuse of and reasoning on skills, the semantic annotation and its state machine counterpart are uploaded to a skill server, named Knowledge Integration Framework (KIF)[1]. KIF contains ontologies semantically describing skills, devices and workpieces. KIF is queried by services such as graphical task specification tools where the user specifies the desired sub goals of the task which in turn is translated into robot code.

[1]not to be confused with Knowledge Interchange Format [20]

3.1 Architecture

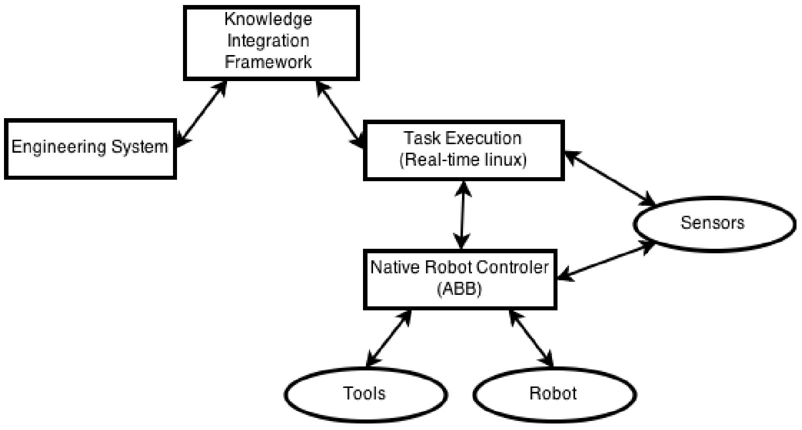

Knowledge Integration Framework
Engineering System
Task Execution (Real-time linux)
Sensors
Native Robot Controler (ABB)
Tools
Robot

Figure 3.1: LTH tool-chain architecture [17].

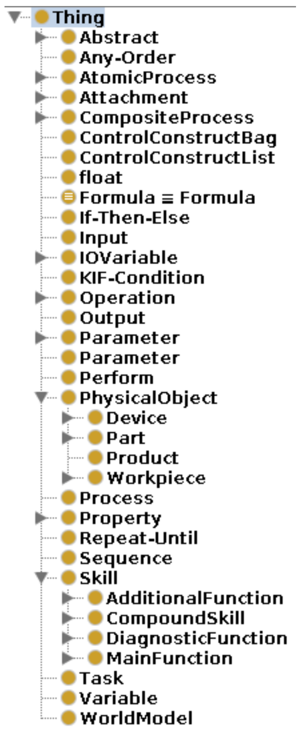
A general overview of the LTH Tool-chain is presented in Figure 3.1. The KIF server serves two types of clients, the Engineering system and the robot task execution system. The task execution system is built on top of the native robot controller. When presented a task, it generates run-time code utilizing on-line code generation, then proceeds by compiling and executing the code. The Engineering system is a robotic programming environment such as ABB Robot Studio. It utilizes the ontologies presented by KIF to model the workspace objects and download skills and tasks from the skill libraries. New objects and skills can be added to the knowledge base by uploading them from the engineering system.

3.2 Ontologies

The ontologies provided by KIF are the result of research conducted at LTH mainly in FP6 and FP7 EU projects SIARAS, RoSta and ROSETTA.

3.2.1 ROSETTA

The core ontology, ROSETTA, is the result of several EU programmes. It originates in the SIARAS programme and was further developed in the RoSta Program. During the Rosetta Programme it was refactored, extended and incorporated into the KIF server. The scope of ROSETTA is aimed at a generic domain ontology for industrial robotics. The focus is on robotic devices and skills, every device can offer one or more skills and every skill may be offered by one or more devices. Manufacturing processes are divided into tasks, each realized by some set of skills. Skills are either primitive or compound items. See Figure 3.2 for an overview of the ontology.

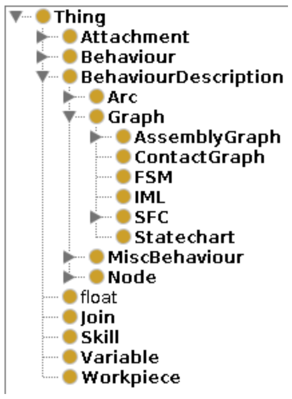


```
▼ Thing
  ▶ Abstract
    Any-Order
  ▶ AtomicProcess
  ▶ Attachment
  ▶ CompositeProcess
    ControlConstructBag
    ControlConstructList
    float
    Formula ≡ Formula
    If-Then-Else
    Input
  ▶ IOVariable
    KIF-Condition
  ▶ Operation
    Output
  ▶ Parameter
    Parameter
    Perform
  ▼ PhysicalObject
    ▶ Device
    ▶ Part
      Product
    ▶ Workpiece
    Process
  ▶ Property
    Repeat-Until
    Sequence
  ▼ Skill
    ▶ AdditionalFunction
    ▶ CompoundSkill
    ▶ DiagnosticFunction
    ▶ MainFunction
    Task
    Variable
    WorldModel
```

Figure 3.2: Partial ROSETTA ontology structure.

3.2.2 SFC

SFC describes various behaviour representations using different kinds of executable state machines. It also contains the semantic description of several graph-based representations of assembly that may also be considered behaviour specifications. See Figure 3.3 for an overview of the ontology.



```
▼ Thing
  ▶ Attachment
  ▶ Behaviour
  ▼ BehaviourDescription
    ▶ Arc
    ▼ Graph
      ▶ AssemblyGraph
        ContactGraph
        FSM
        IML
      ▶ SFC
        Statechart
    ▶ MiscBehaviour
    ▶ Node
    float
    Join
    Skill
    Variable
    Workpiece
```

Figure 3.3: Partial SFC ontology structure.

3.2.3 FRAMES

An initial proposal of an ontology describing the frames of the iTasC framework presented in Section 3.3.1. Experimental development during research to enable representation of task modelling concepts such as frames, degrees of freedom and relative positions. The ontology is rudimentary and does not allow complete task modelling with the iTasC framework. See Figure 3.4 for an overview of the ontology.

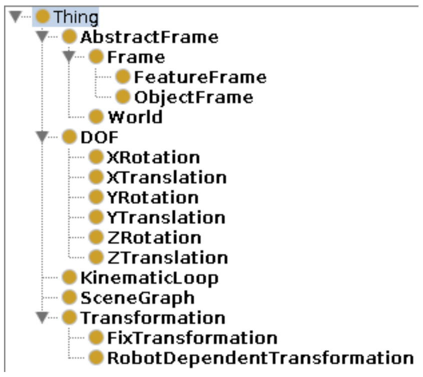


Figure 3.4: Partial FRAMES ontology structure.

3.2.4 SHIELDCAN and PENCALIBRATION

Ontologies with concepts concerning specific skills. SHIELDCAN contains concepts of the skills **Align**, **AssembleButtonBox**, **Shieldcan** and **Snapfit**[2], while PENCALIBRATION only contains the **Pencalibration** skill. The different skills have different vocabulary and do not adhere to a skill representation model. In Section 4.1.1 we present the identified concepts of the various skills in these ontologies along with the development of the new model.

The ontologies also contains general concepts that are not specific to a certain skill. This results in cluttered and unstructured references since skills must depend on knowledge represented in other skills.

See Figure 3.5 for an overview of the SHIELDCAN ontology.

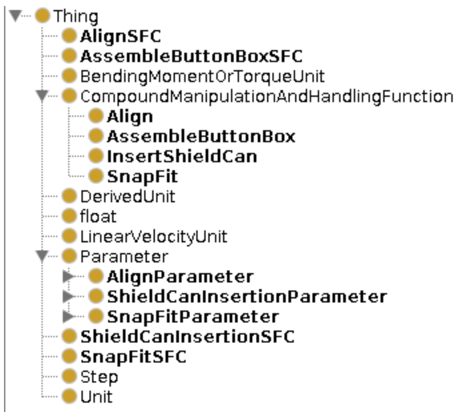


Figure 3.5: Partial SHIELDCAN ontology structure.

[2]**Snapfit** is described in more detail in Section 3.3.5

3.2.5 Ontology hierarchy

The initial ontology structure is presented in Figure 3.6. ROSETTA is the main ontology and imports only the QUDT ontology. SFC only imports the ROSETTA whereas SHIELDCAN imports the ROSETTA and the SFC ontology. The PENCALIBRATION ontology imports ROSETTA, SFC, SHIELDCAN and FRAMES ontologies. The structure suffers from duplicated imports since when you import for example SHIELDCAN, the imports of SHIELDCAN are imported indirectly. This leads to a complicated knowledge base that is hard to grasp.

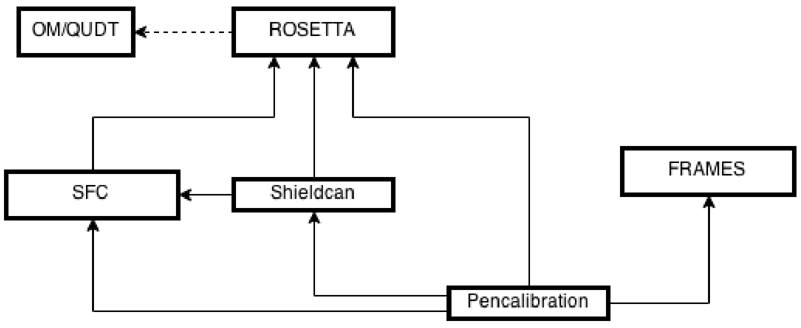


Figure 3.6: The structure of the ontology imports.

3.3 Robot skills

In [21] a skill description is presented as a combination of primitive robot capabilities that are non trivial and have production value. These capabilities can be combinations of the previously mentioned state machines or manipulation primitives and motion constraints. By combining this with a semantic approach skills may be treated as compositional pieces of declarative, portable and directly applicable knowledge on robotized manufacturing [17].

3.3.1 Motion constraints

At the lowest level the skills are described by motion constraints based on the iTasC framework [22]. iTasC provides a constraint-based approach to specify the motion in complex tasks for general sensor-based robot systems. The approach uses feature coordinates, defined with respect to objects and feature frames, to express the relative motions and dynamic interactions of the task. Task specification is achieved by first identifying task relevant objects and features and assign frames to them. Each feature is assigned to an object that represents a physical entity of that object, such as an edge or a surface. The relative motion between two objects is specified by imposing constraints and the relative motion between feature frames of the two objects. The connections of a task forms a closed loop and is called a kinematic chain. The kinematic chain and its constraints is used by the solver to execute the task.

3.3.2 Robot programming languages

Robot programming languages are mainly designed for motion specification. Move arm to target position 1, close gripper, move to target position 2, etc. Most industrial robot manufacturers have developed languages targeted at their own machines, for example KRL for KUKA, INFORM for Motoman and RAPID for ABB [23].

3.3.3 State machines

The skills composed of motion constraints and/or robot code described above are executed in a state machine. Each state is represented by a block with initial conditions, exit conditions and error handling, to ensure proper execution. When one block exits the next follows if the initial conditions are fulfilled. Parameters for these blocks with semantic value are stored in KIF.

3.3.4 Qualifying aspects

Skills are compositional items. They can be non-divisible or compound items and have a large number of different attributes and descriptors. At the lowest level are the **primitives** that are non-divisible items to be used for composition of skills. To be qualified as a **skill**, a declarative definition is required, otherwise they are referred to as skill candidates. A **skill candidate** augmented by a declarative description gives a skill. All behaviour descriptions, such as state machines, are assumed to be declarative.

3.3.5 Skill example: SnapFit

The **snapfit** skill was developed as part of the assembly of a stop button box. During the assembly an item needs to "snap" into place and this is were this skill was used. This skill has an SFC behaviour description and a declarative model in the SHIELDCAN ontology.

The declarative ontological representation of the **snapfit** skill can be viewed in Appendix A. This Figure shows only the contents of the SHIELDCAN ontology which are related to the **snapfit** skill. As can be seen, the structure is not straightforward. The skill is connected to some of its classes such as the conditions, but lacks a declarative connection between itself and for example its parameters or behaviour description. Further, some concepts more general than the individual skill is described here. For example the conditions of a skill is described in the context of the **snapfit** skill. This description is used for referring to conditions in other skill descriptions, which indicates that it should be implemented on a more general level.

Chapter 4

Results

In this chapter we will present the results of the thesis. The chapter is divided into three parts. First we present the developed model, then the modularization of the ontologies. Finally, we introduce the developed proof of concept for accessing the knowledge.

The resulting class hierarchies and properties can be found specified in SUO-KIF in Appendix B. SOU-KIF stands for Standard Upper Ontology Knowledge Interchange Format and is language designed for use in the authoring and interchange of knowledge [20]. This language is in general not interpretable by machines. Therefore we have developed an OWL version. OWL is a knowledge representation language for authoring functional ontologies. The OWL ontologies are accesible from `kif.cs.lth.se/ontologies/[ontology_name].owl` by replacing `[ontology_name]` with the name of the desired ontology, for example `kif.cs.lth.se/ontologies/rosetta.owl`.

4.1 Developing a skill model

In this section we will present the developed model for representation of a skill. The presentation is based on the points presented in Section 2.4.1.

The material in this section is the result of studying the aspects of the LTH tool-chain. The implementation of the tool-chain described in Chapter 3 contains a large amount code and parameters, executed in a complex tool chain. Much of this information has no semantic value in the perspective of declarative, reusable skills. However, parts of it is descriptive and this constitutes the valuable semantics of a skill. With the goal of declarative, portable and directly applicable knowledge we identify and expose high level semantic descriptors of a skill and incorporate them into our model.

4.1.1 Knowledge acquisition

By studying currently implemented skills four primitive concepts have been identified as descriptors of a skill. They can be found in the various skill representations of the LTH skills, but never in a complete manner. The contents of this section can be seen as the requirements of the representation model in the following section.

BehaviourDescriptions of the runtime behaviour of the skill.

Parameters of the behaviour descriptions.

Frames for the task description.

Requirements for performing the skill.

Describing how these concepts are relevant for the skill and the tool chain will lead to knowledge that can be utilized as requirements for the modeling of a skill.

BehaviourDescriptions

The behaviour descriptions describe the run-time behaviour of the skill. The implementation varies between manufacturers and tool-chains. Low level robot control code may be manufacturer specific, such as RAPID code from ABB, while state machines may be in one of the multiple implementations available, such as Sequential Function Charts (SFC) [24].

When modeling the behaviour descriptions of a skill it is important we get a clear distinction between the different approaches.

Parameters

Depending on the granularity level of control and the run-time behaviour descriptor type we can expose parameters with a high intrinsic semantic value, such as force applied or joint velocity, from the behaviour representations. These parameters are skill-specific and may be changed for adaptation of the skill to a new workpiece, workplace or task.

The parameters may come with a default value to revert to if the user chooses not to set the parameter. These values require a systematic way of representing units and measures. Each parameter may also have an intrinsic measure such as *Newton* or *m/s* which also need to be easily represented. Reasoning on the type of measure and converting between units may prove beneficial to the user and the system.

When modeling the parameters we need support for the intrinsic measures of the parameters and/or their default values.

Frames

Frames and their frame coordinates form the foundation for the motion constraint control based on the iTasC framework. They have high semantic value since they describe the task model and the geometry of the workplace.

The ITasC formalism began with the task frame [25] which was replaced by and extended to multiple feature and object frames [22]. The object frames represent objects in the task space, such as a workpiece or a robotic arm, while the feature frames represent features on the objects, such as a corner or a face.

By using multiple frames the task geometry can be partially modeled by each frame using translational and rotational directions along the axes of a frame, called feature coordinates. Each frame specifies part of the constraints and the model consists of the partial descriptions from the individual frames. The complete set of frames expressing the model is connected by a closed kinematic chain. Finally, each frame can be accompanied by an uncertainty frame, representing the geometric uncertainty for each of the feature coordinates of the frame.

The frame connects position and orientation into a complete entity. Each frame is positioned relative to a reference frame forming a hierarchical system, grounded in a world coordinate frame. A position is a 3-dimensional vector describing a position relative to a reference position. The values of the vector depends on the chosen coordinate system: Cartesian (XYZ), Spherical($r\theta\phi$) or Cylindrical ($\rho\theta z$). An orientation is the final piece needed to describe the frame. It may consist of for example a rotation matrix, Euler-angles or a quaternion.

The kinematic chain connects the frames of a model into a closed loop. By letting each frame reference the next frame the loop is achieved. The direction of the loop is important for the correct execution of the movements. Representing the kinematic chain of the task description requires a hierarchical notion of frames and their coordinate systems.

To support the iTaSc model ontologically we need unambiguous representation of position and orientation. We need to support the frames by coordinates and coordinate systems expressing both positions and orientations quantitatively.

Requirements

Requirements are divided into two parts, device requirements and conditions. Device requirements are qualifying preconditions on the devices involved in the task, such as a mounted force sensor. Conditions may be pre- or post-conditions and have a broader definition since the notion of a condition is arbitrary. Examples are the existence of certain frames or their geometrical relations in the task descriptions.

When representing the requirements of a skill we need clear distinction between device requirements and conditions as well as qualitative temporal description as pre and post execution.

4.1.2 Domain of the skill model

The **domain** of the skill model is the representation of concepts relevant to skills for the tool-chain at LTH. The **application** is to produce compositional pieces of declarative, portable and directly applicable knowledge on robotized manufacturing. The **user** is a robot execution system or an engineering station where the user can download skills. The **maintainer** is a human working with the ontology engineering software.

4.1.3 Class hierarchy

Because of the amount of existing knowledge, see Section 3.2, the **combination** approach was applied to the development of the classes and the hierarchy. The existing knowledge was incorporated into our new model and augmented by new concepts and class hierarchies. By using existing knowledge as part of our model we get a tightly integrated model with the existing knowledge base.

The classes developed mirror the concepts and requirements presented in Section 4.1 by using classes and subclasses, see Figure 4.1.

- Our main class `Skill` is part of the ontology ROSETTA. This way we make use of and integrate the skill concept into the existing model using the ROSETTA ontology as our core.
- `BehaviourDescription` is part of the ontology SFC. This ontology has proven to successfully describe behaviour descriptions in previuos work at LTH and is therefore unaltered. The notion of `CodeImplementation` was not implemented in the ontology SFC but was implemented separately for a skill. Its general purpose was identified and it was been added class hierarchy of the behaviour description.
- The `Frame` class is partitioned into `FeatureFrame` and `ObjectFrame`
- The `Parameter` class is an existing class from the ROSETTA ontology, it is augmented by the optional property of a unit.
- The `Requirement` class is partitioned into `Condition` and `DeviceRequirement`. `Condition` is further partitioned into `Precondition` and `Postcondition`.

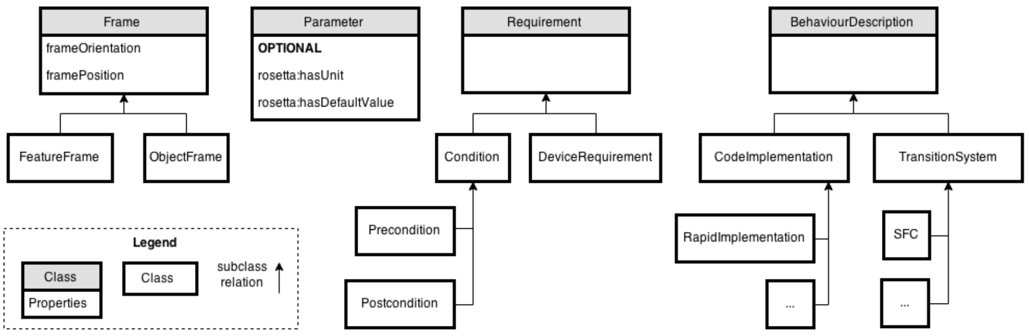


Figure 4.1: The class hierarchy of the skill model.

4.1.4 Properties of the classes

The properties of the classes are modeled by using predicates. The predicates of the model represent the connections between the classes, see Figure 4.2. These predicates are intended to be links between the skill and its concepts. Our main class `Skill` will be connected to the other classes of our model, and this way pointing to the properties of the skill. The superpredicates have an organizational purpose along with the ability to impose general properties on the subpredicates.

- *hasFrame* connects the skill to the frames.
- *hasParamter* connects the skill to the parameters
- *hasRequirement* serves as a superpredicate for the requirements.
- *hasCondition* connects the skill to the conditions.
- *hasDeviceRequirement* connects the skill to the DeviceRequirements
- *hasBehaviourDescription* serves as a superpredicate for the behaviour descriptions
- *hasCodeImplementation* connects the skill to a robot code implementation.
- *hasTransitionSystem* connects the skill to a transition system.

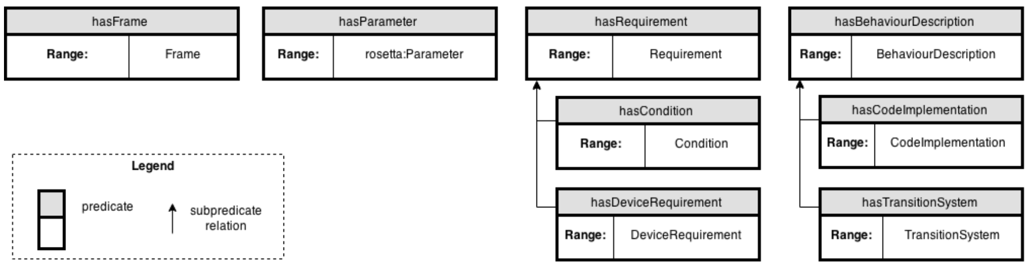


Figure 4.2: Predicates for the skill model.

4.1.5 External ontologies

Two domains have been identified where the use of external ontologies will be beneficial.

- Units and Measures
- Positioning and Orientation

These domains have a long history of ontological research and there are multiple attempts at standardizing ontologies in each of the domains [26, 27, 28]. Both areas present unique challenges that needs to be solved in a satisfactory way for the user.

Units and measure

The previous representation of units and measure at LTH was achieved by exploiting the QUDT ontology [26]. New options were needed since the ontology was inconsistent. The OM ontology [29] is the result of evaluation of other ontologies in the domain which finally led to the development of the ontology. Note that the OM ontology only is consistent with some of the reasoners, such as FaCT++. The OM ontology is incorporated into the skill model by using the ROSETTA predicate *hasUnit*. The resulting model is straight forward and non intrusive to deeper parts of the knowledge.

Object	Predicate	Subject
Skill1Parameter	hasUnit	OM:Newton

Figure 4.3: The usage of the OM ontology for units and measure.

Positioning and orientation

The need for a generally accepted representation of position and orientation is not unique to our effort. A recent development by the IEEE [30] has led to a spatial position and orientation ontology combining both qualitative and quantitative representations [28]. We have chosen to make use of these results with a few modifications.

CORA and thereby CORA:POS is based on the upper ontology SUMO[1]. SUMO is a major effort to provide an upper ontology for a variety of computer information processing systems. The focus of CORA:POS is on the positioning and orientation of objects. An object is a subclass of **SUMO:Physical** in SUMO.

The ontologies at LTH are based on the ROSETTA ontology with no roots in SUMO. This causes discrepancies between the positioning of objects in CORA:POS and our intended positioning and orientation of frames. A frame is an abstract concept in the ROSETTA ontology and therefore not comparable to a **SUMO:Object**, see Figure 4.4 and Figure 4.5.

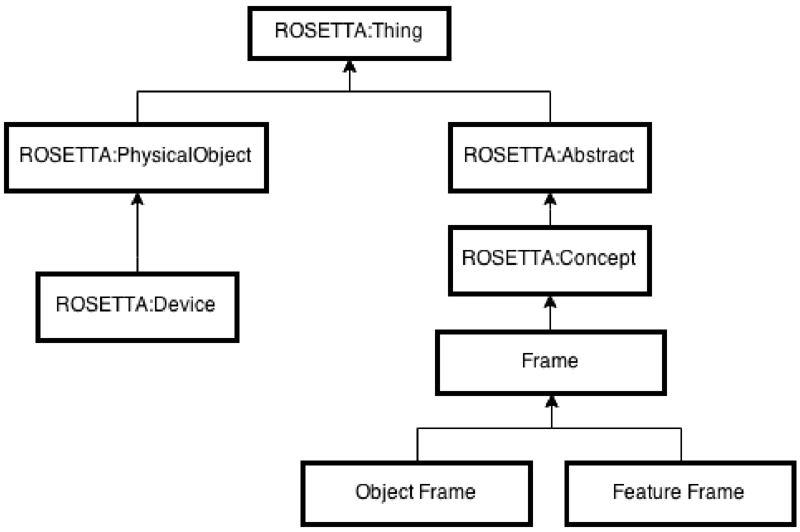


Figure 4.4: Major concepts of the ROSETTA ontology

An important note here is the notion of pose. POSE is a part of the CORA:POS ontology and its is clearly closely related to a FRAME. Purely comparing descriptors, a pose and our frame have little to no difference, both contain a position and an orientation. The separation is based on their intrinsic semantic value, the pose is an attribute of an entity whereas the frame is an entity in itself. Further a frame has a clear connection to task modeling using an iTasC formalism, while a pose is a more general concept.

These discrepancies has led to the development of a layer on top of CORA:POS mimicking the object positioning and orientation but with regards to a FRAME instead, see Figure 4.6. These are used in the same way that the object positioning is made in the

[1]Suggested Upper Merged Ontology

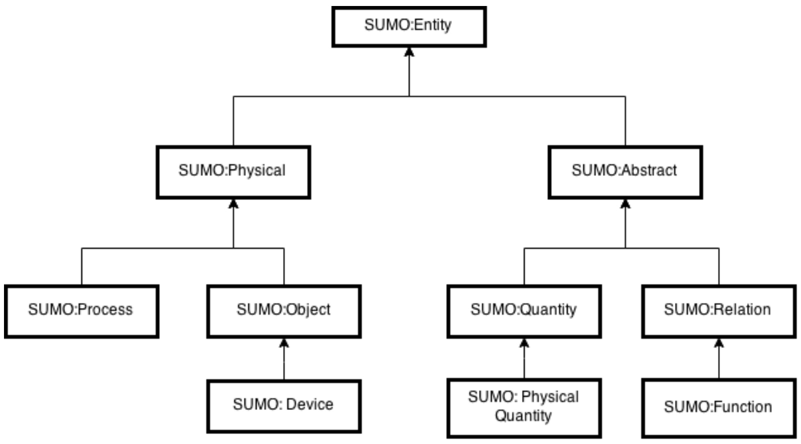


Figure 4.5: Major concepts of the SUMO ontology.

CORA:POS ontology [30, 31]. Figure 4.7 and Table 4.8 demonstrates the use of frames for modelling the task.

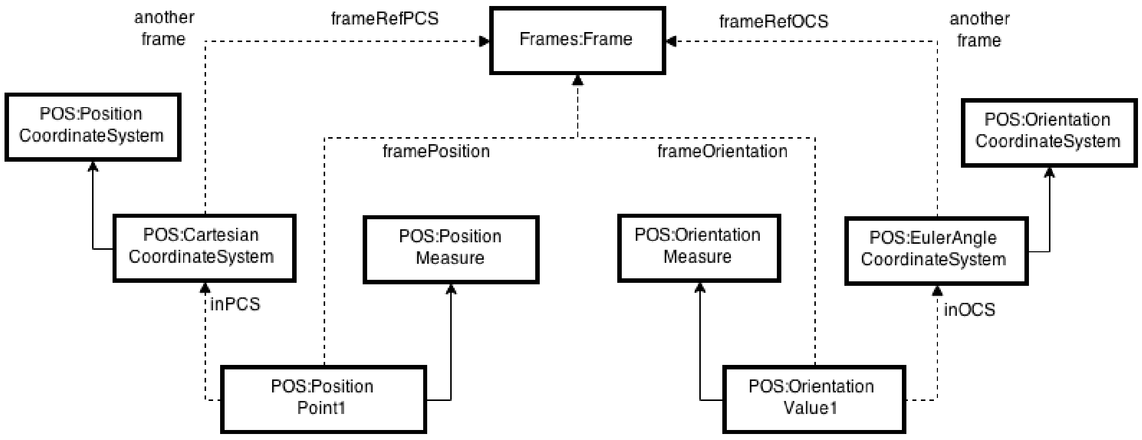


Figure 4.6: Frame positioning and orientation.

4.1.6 Example of a skill in the model

Here we present an example of a skill using the developed model. As can be seen in Figure 4.9 the skill is described by the predicates pointing at the classes that comprise the concepts of the skill. Each class has subclasses that indicate the actual property. The concepts of each class is modelled by the class hierarachy. For example, the class `Skill1Frame1` is a subclass of `Skill1Frame` to indicate its connection to `Skill1`. It is also a subclass of `FeatureFrame` to indicate that it is of the type Feature Frame. This way we get a clear connection between the skill and its concepts and clear class hierarchy for describing the connected concepts.

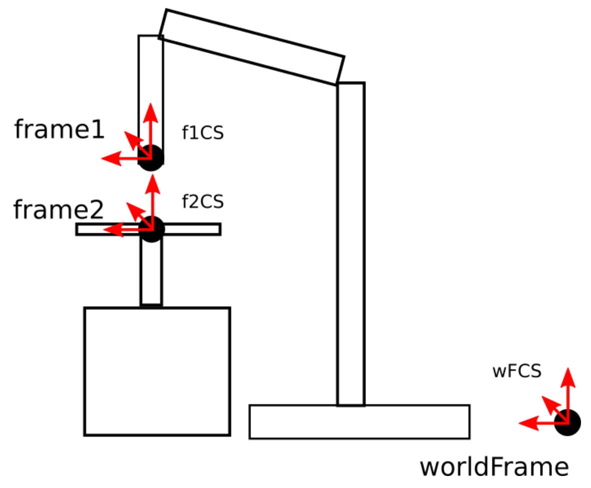


Figure 4.7: Sketch of a robot with frames and coordinate systems. This figure demonstrates the classes from Table 4.8 and it is not a complete example of the usage of object, feature and world frames.

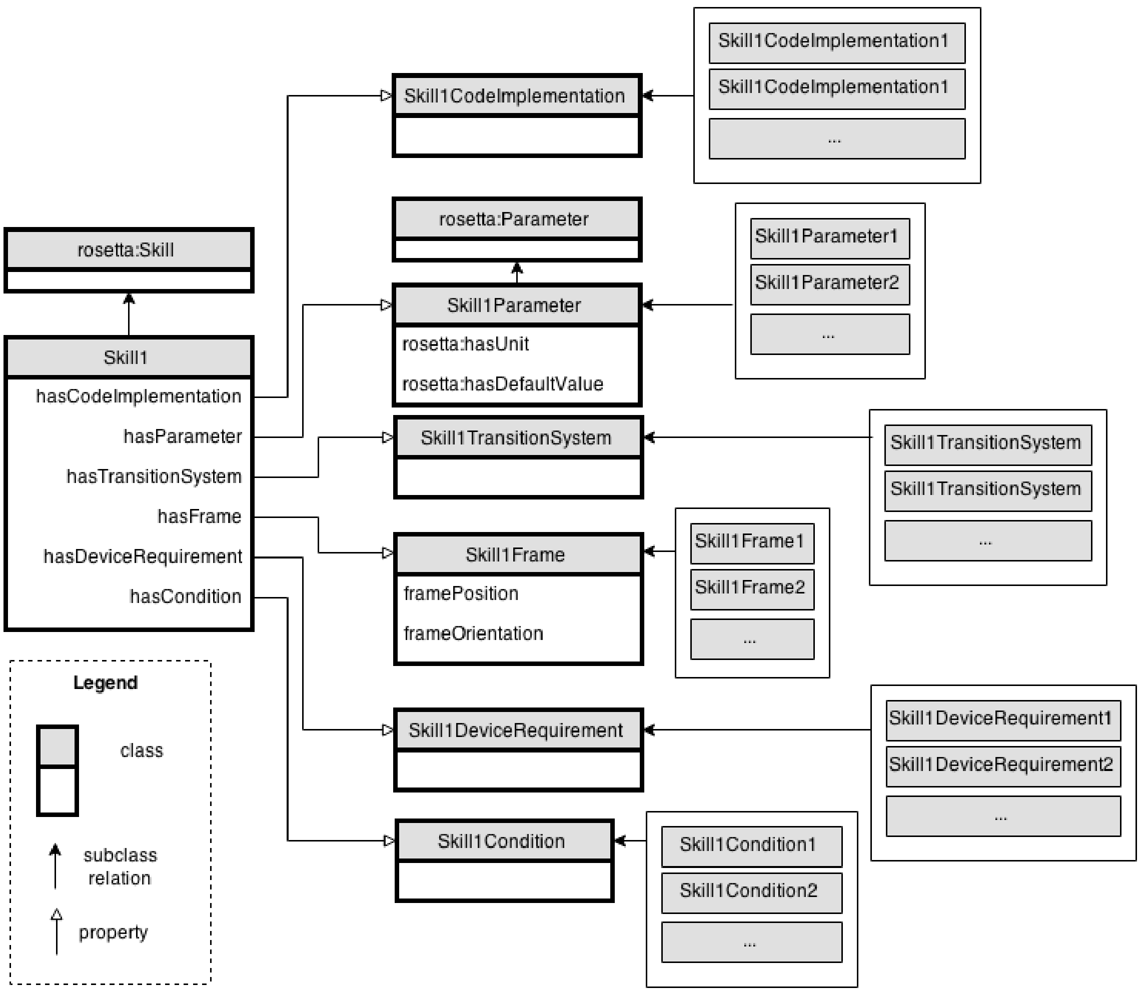


Figure 4.9: Outline of a skill. Skill1 is an example to demonstrate the model.

Object	Predicate	Subject
worldFrame	subClassOf	ObjectFrame
wFPosition	subClassOf	CartesianPositionPoint
wFPCS	subClassOf	CartesianCoordinateSystem
worldFrame	framePosition	wFPosition
wFPosition	inPCS	wFPCS
frame1	subClassOf	ObjectFrame
f1Position	subClassOf	CartesianPositionPoint
f1PCS	subClassOf	CartesianCoordinateSystem
frame1	framePosition	f1Position
f1Position	inPCS	f1PCS
f1PCS	frameRefPCS	wFPCS
frame2	subClassOf	ObjectFrame
f2Position	subClassOf	CartesianPositionPoint
f2PCS	subClassOf	CartesianCoordinateSystem
frame2	framePosition	f2Position
f2Position	inPCS	f2PCS
f2PCS	frameRefPCS	wFPCS

Figure 4.8: Positioning of the frames in Figure 4.7.

4.2 Modularizing the ontology structure

To begin with we will provide an overview of the ontologies and the process of extracting them. Later we will go into more detail on each of the ontologies.

4.2.1 The goal of the modularization

The goal of the ontology-modules for the knowledge base is to increase re-usability and manageability. Extracting meaningful modules will help management by clearly showing were knowledge is intended to be implemented. Finding these modules requires the identification of separate areas of the skill concepts. These modules will also contribute to the re-usability by constituting natural areas of meaningful knowledge. The second aspect to the modularization is the separation and extraction of the skills. By letting each skill be represented by a module they can easily be reused and managed.

Meaningful modules

The semantics-driven strategy for modularization calls for the identification of disparate areas of the modelled domain, pieces of knowledge with high coherence and relatively low connection to the other concepts of the domain.

In [32] the authors presents an idea to declaratively define a skill as the configuration and coordination of capabilities according to the intentions. The insights derive from the work in [33] where the separation of computation, coordination and configuration is recommended for evolving large-scale software systems. Further the research from [34]

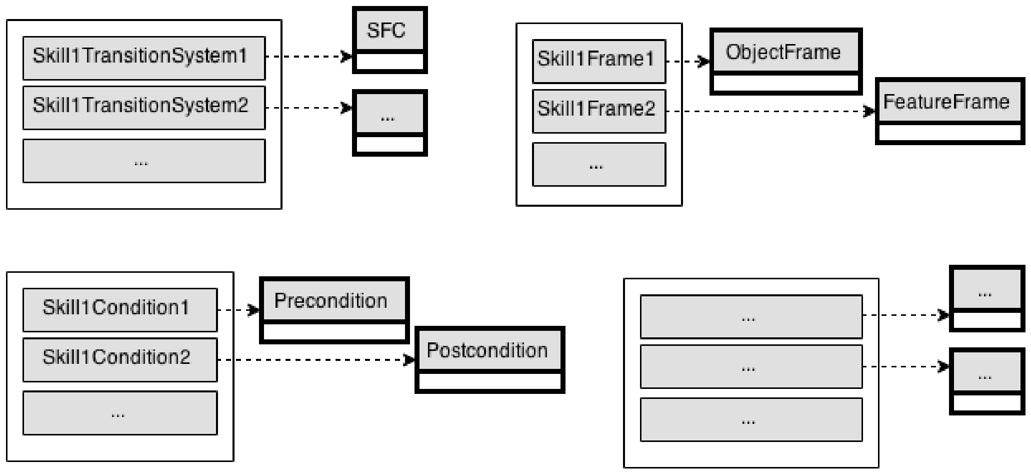


Figure 4.10: Example of inheritance for concepts of skill example.

presents a separation of concerns into communication, computation, coordination, configuration and composition as a recommended best practice when developing large software frameworks.

The definition of the terms configuration and coordination is as follows:

coordination - the harmonious combination or interaction, as of functions or parts.

configuration - the relative disposition or arrangement of the parts or elements of a thing.

With the semantics-driven approach to modularization, presented in Section 2.5, and the ideas from [32], the middle level ontologies CONFIGURATION and COORDINATION were developed. Following the definitions of the terms, interpretations are made with regards to skill representation.

Coordination is interpreted as the items describing the actual run-time behaviour.

Configuration is interpreted as the arrangement and setting of specifications determining the run-time behaviour.

As a result the ontologies CONFIGURATION and COORDINATION are grounded in natural language and represent two disparate areas of the concepts of a skill.

The extraction of skill-specific ontologies enables the use of skills as reusable items. This extraction is also semantics-driven but on a lower level. This is a matter of generality and specificity. All of the knowledge of the skill-specific ontologies would fit conceptually in either CONFIGURATION or COORDINATION. But, the knowledge in each of the skill modules concerns only the specifics of that skill. This way, all general knowledge on skills belongs in either CONFIGURATION or COORDINATION, depending on its nature, while all skill-specific knowledge belongs in a skill module.

4.2.2 The ontologies

In order to make sure that future knowledge is incorporated in correct place we will present some guidelines to the scopes of the resulting modules. Each module is a ontology describing its domain in itself. This is related to *step B* of the model development process of an ontology, presented in Chapter 2.

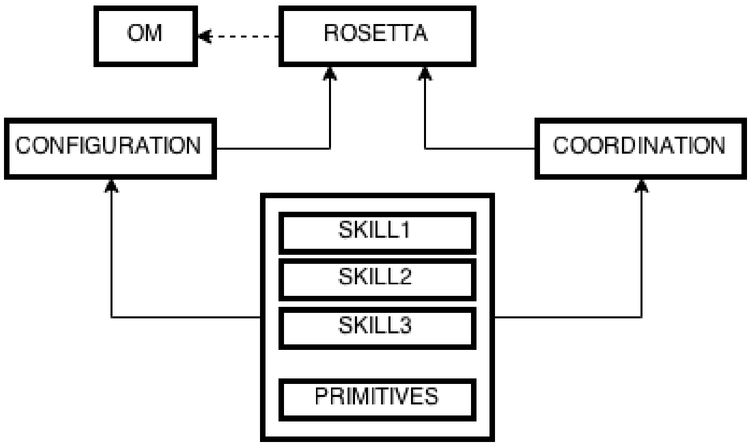


Figure 4.11: The final structure of the ontologies.

ROSETTA

The ROSETTA ontology is outside of this thesis as it is previously defined and the result is an iteration of several projects.

The **domain** of ROSETTA is the general concepts of industrial robotics such as devices, workpieces and skills. The generality of the ontology is key only and major concepts qualify. The **application** of ROSETTA is extracting knowledge and categorizing to support these devices, skills and workpieces. The **user** is either an engineering station or the execution system of the robot while the **maintainer** is a human using some ontology engineering software.

CONFIGURATION

The **domain** of CONFIGURATION is the configuration of skills. Conditions for execution and finalization and configurations of input and output values, semantically relevant parameters, and frames. The **application** is supporting the skill representation and clarifying the semantic skill composition. The **user** is either an engineering station or the execution system of the robot while the **maintainer** is a human using some ontology engineering software.

The CONFIGURATION ontology is based on the previous knowledge from SHIELDCAN, FRAMES and PENCALIBRATION.

COORDINATION

The **domain** of COORDINATION is the representation and combination of parts that describe the run-time behaviours of the skill. This includes executable state machines and robot code, as well as several graph-based representations of assembly such as assembly graphs, constraint graphs and task graphs. The **application** is supporting the skill representation and clarifying the semantic skill composition. The **user** is either an engineering station or the execution system of the robot while the **maintainer** is a human using some ontology engineering software.

The COORDINATION ontology is based on the previous knowledge in SFC and SHIELD-CAN.

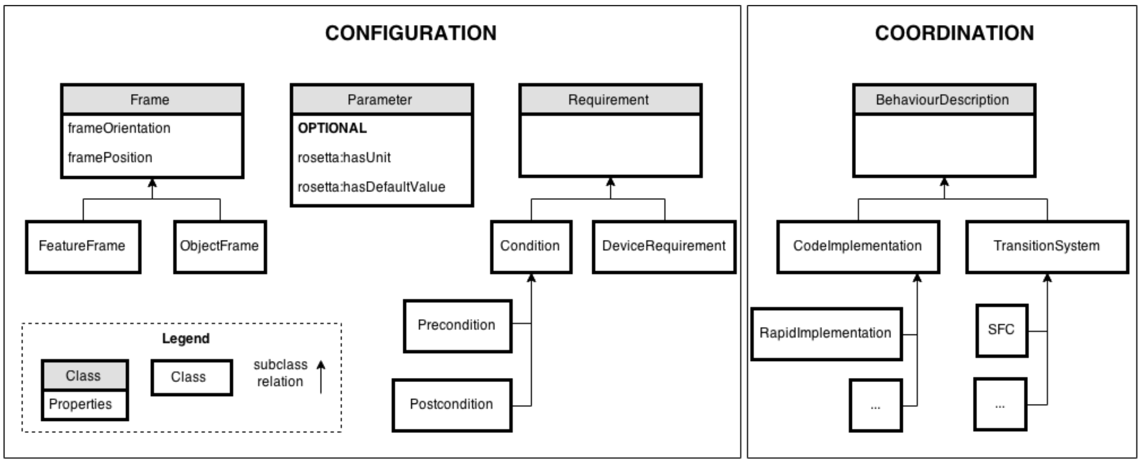


Figure 4.12: The contents of CONFIGURATION and COORDINATION.

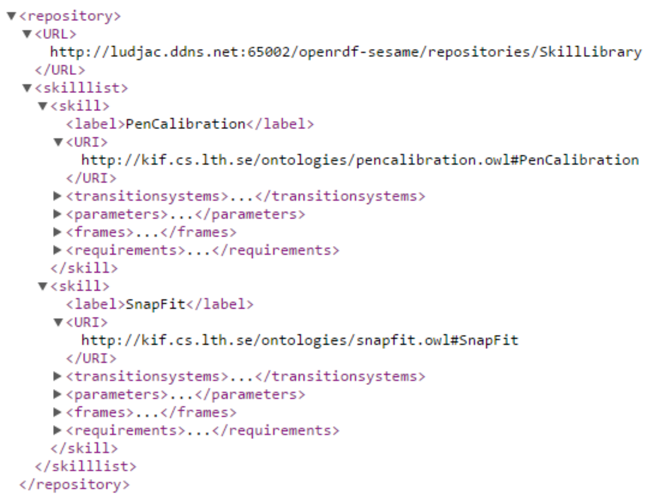
Scope of skill ontologies

For the sake of consistency we refer to a made up skill called **Skill_1** in the following paragraph.

The **domain** of SKILL_1 is the description and definition of the concepts relevant only to the **Skill_1** skill. This is in general subclasses of classes from the higher level ontologies CONFIGURATION, COORDINATION and ROSETTA. The **application** is the manifestation of *Skill_1* for use as a declarative reusable skill. The **user** is either an engineering station or the execution system of the robot while the **maintainer** is a human using some ontology engineering software.

4.3 Accessing the skills

To demonstrate the capabilities of the model we have developed a conceptual version of a skill-knowledge access API. This illustrates that a future skill library is within reach. The results of the demonstration is also an indication of the correctness of our model. By showing that we can extract skills from the knowledge base we have indicated that our model successfully represents the modelled domain.



```
▼<repository>
  ▼<URL>
     http://ludjac.ddns.net:65002/openrdf-sesame/repositories/SkillLibrary
   </URL>
  ▼<skilllist>
    ▼<skill>
       <label>PenCalibration</label>
      ▼<URI>
         http://kif.cs.lth.se/ontologies/pencalibration.owl#PenCalibration
       </URI>
      ▶<transitionsystems>...</transitionsystems>
      ▶<parameters>...</parameters>
      ▶<frames>...</frames>
      ▶<requirements>...</requirements>
     </skill>
    ▼<skill>
       <label>SnapFit</label>
      ▼<URI>
         http://kif.cs.lth.se/ontologies/snapfit.owl#SnapFit
       </URI>
      ▶<transitionsystems>...</transitionsystems>
      ▶<parameters>...</parameters>
      ▶<frames>...</frames>
      ▶<requirements>...</requirements>
     </skill>
   </skilllist>
 </repository>
```

Figure 4.13: The outline of the xml served by the server.

As the development is of the conceptual kind we have decided not to delve deep into the technicalities and solutions of the resulting software, but we will present a brief review.

The resulting software is divided into a back-end and a front-end. The back-end provides the interactions with the knowledge base and serves the results to the front-end, which interacts with the user. The back-end is intended to serve platform independent results so that any front-end can utilize the service. This way the front-end can be any skill consuming user, be it ABB RobotStudio or any other agent. This division of labour ensures that the skills are delivered in a unified matter by the back-end, and that development and management of the back-end and front-ends can be performed independently.

The ontologies are formalized in OWL[2][3] and are hosted in a Java framework for processing and handling RDF data databases called, rd4j Sesame[4]. Sesame provides a SPARQL[5] endpoint to enable querying the ontologies. Our developed back-end queries the Sesame endpoint and constructs java objects from the results representing the skills. The java objects are then serialized into XML and served by a java servlet. This way we provide a web service that serves the skills in XML format to virtually any front-end.

The implementation can be found at:
`https://git.cs.lth.se/jacek/skill-ontologies/`.

By accessing the URL `http://[serverurl]/skillendpoint/skills` on the server where the servlet is deployed, it list the skills in the openrdf repository called ’Skill_Library’ on that same server.

[2]`http://www.w3.org/TR/owl-features/`
[3]`http://www.w3.org/TR/owl2-overview/`
[4]`http://rdf4j.org/`
[5]`http://www.w3.org/TR/sparql11-query/`

Chapter 5
Discussion

5.1 Separation of existing knowledge and our results

The existing ontologies at LTH have been in use for several years. They are the result of several EU projects and have been extended multiple times. The implemented knowledge is the result of previous or current research projects. Since much of this work, especially regarding skills, is of experimental nature, with trial and error-like work processes, the majority of the implemented knowledge is undocumented and largely unstructured. This means it is hard to outline the previous state of the knowledge base when we initiated the project in a clear way. Without this outline, the distinction between the results of this thesis and the existing previous knowledge is sometimes unclear. To amend this we have tried to show the state of the ontologies in Chapter 3 by showing pictures of their class hierarchy providing a diagram of the skill representation in Appendix A.

5.2 The model

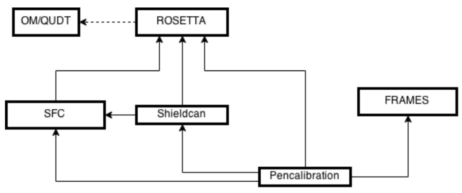
The developed model provides a representation for the parts of a skill we have identified as relevant. This is based on the current state of the skills at LTH and the way they are used as of today. As we initiated the project we identified confusion and a non-standardized approach to using the ontologies. Several systems tap into the knowledge base and make use of the implemented knowledge. Because of this need for backward compatibility we had to exercise caution when altering the knowledge base. When we had the chance we made use of existing knowledge in order to reduce the size and redundancy of the final model. Because of the knowledge being largely undocumented reusing knowledge proved hard.

Another conclusion to be made from the state of the model before this thesis is the need to commit to the model. Each time knowledge is inserted into the knowledge base the user has to make sure that the knowledge is aligned to the model and is placed in the correct place. Otherwise the state of the knowledge base is soon back in the previous state of confusion.

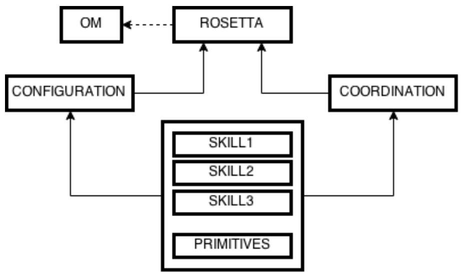
The need to change the approach for knowledge for industrial robots from a *closed world assumption* to an *open world assumption* is supported by the ontological approach to knowledge representation. A closed world assumption means that the complete nature of the environment the robot acts in is known. The exact position and nature of everything currently in the workplace as well as everything that will ever appear is known by the robot. This works fine when the robot does not have to act or react on new information. The industrial robot of the future is adaptable to new situations and new information where not everything is known, this is called the open world assumption. This type of reasoning is closer to the way we humans interact with out surroundings. A robot with this assumption can learn things about its surroundings and incorporate this information into its knowledge base. In order to support this approach we need a knowledge representation system that is easily extended with new knowledge in a systematic way. The nature of ontologies support the open world assumption. New knowledge can easily be added in such a way that knowledge that was learned can be related to what we knew before. When the need arises for new concepts of a skill they can easily be incorporated in the model by adding classes to the class hierarchy. The developed model can successfully represent the current skills in the LTH knowledge base and we feel confident that it is flexible enough for the representation of future skills. It has a clear class hierarchy that incorporates the different parts of a skill.

The development of a knowledge representation model is ridden with situations where one has to make decisions regarding the model where the solution is ambiguous. The resulting model is a reflection of the understanding and perception of the developer. This leads to a model that is correct in the eye of some, and incorrect in the eyes of others. During the development of our model we have encountered these situations on some occasions, for example during the choice of primitive classes. Is the class `Requirements` properly partitioned by its subclasses `Condition` and `DeviceRequirements`, or is `Condition` a candidate itself for a primitive class? The decision was made based on discussions and previously implemented knowledge but it is clear that there is not an unique solution.

Regarding the general development of ontologies is is our understanding that the properties of the classes provide important richness in the represented knowledge, as is supported by the authors of [8]. By augmenting each class with properties the machine can get a better understanding of each class than by merely relying on class hierarchies. A class `Desk Lamp` that is the subclass of `Lamp` knows only that relation without any further properties. If the class is augmented by the properties color, size, weight, and so on, the knowledge gets richer. The ontologies at LTH could make stronger use of properties for its classes. The general approach to the ontology development seems to be based on constructing a rich class hierarchy for the concepts in the knowledge domain. This has of course influenced our process, and our model resembles the other LTH ontologies as they are intended to coexist and interact. We believe it would be worthwile to augment the current LTH knowledge base with more properties, this will aid future algorithmic reasoning on the use of skills during manufacturing.



(a) Old structure and modules.



(b) New structure and modules.

Figure 5.1: Ontology modules before and after modularization.

Comparing the original **snapfit** description in Section 3.3.5 and the resulting model in Section 4.1.6 we see the change of skill representation. The developed model has a clear connection between the skill class and its concepts. The class hierarchy is used to describe aspects of the skill concepts. This hierarchy allows for a clear differentiation between what is relevant to a specific skill and what is more general by using subclasses of the concepts of a skill to describe skill-specific knowledge.

5.3 The ontology modules

There are multiple strategies and approaches to ontology modularization, *semantic*, as we used in this thesis and also *algorithmic* and *logical* solutions. The drawbacks of the semantic approach is that the resulting module structure needs strong commitment to remain structured. Other approaches to modularization include using algorithms to extract areas that have close connection according to the logic of the ontology. Using this method the modularization can easily be run again at a later stage if needed. However, there is no guarantee that the resulting modules will make sense from a human perspective. We felt that the human perspective is important since the modular structure is largely there for the human users that extend and maintain the knowledge base.

By identifying domains for the ontology modules using a semantic strategy we achieve module domains grounded in the intrinsic characteristics of skill representation. The domains represent unambiguous coherent domains of knowledge. When implementing new knowledge a clear structure of knowledge domains gives the support needed to make sure the ontologies remain organized. This is the big advantage of the semantic strategy to modularizing knowledge bases. The resulting modules have clear scope and their structure adheres to the nature of the modeled domain.

The result of the modularization can clearly be seen in Figure 5.1. The resulting structure allows for the use and development of "pluggable" skill modules. The dependencies are clear and recursive imports are avoided. Finally, by using the semantic approach we have identified intrinsic characteristics of a skill and extracted meaning full modules that allow for easy development of future skill concepts.

5.4 Knowledge access

Following the development of the proof of concept it is our understanding that the resulting model is correct in the representation of the skill. We can successfully deliver skills extracted from the knowledge base to a front-end client.

We believe the development of a back end will prove useful to the skill research at LTH. Current approaches uses queries directly from the client to the knowledge base which results in a non-unified access to the skills. The development of clients then also relies heavily on the developer’s expertise in extracting meaningful data from the LTH knowledge base and may result in different realizations of the skills depending on the front-end developer. A back-end should also provide a unified approach to uploading skills.

A back-end with easy access to the skills will promote the development of future clients. This will provide the skill library with the much needed user base to kick off the concept. The skill library needs skills to attract users, and the library needs users to produce skills.

Chapter 6
Conclusions

As was made clear in Chapter 5 the resulting skill model and modularization provides a clear representation of a skill.

The model for representing a skill provides the support needed for the existing skills and for the development of future skills. New knowledge may easily be implemented by adhering to the developed class hierarchy and extending it if necessary.

The modularization provides an environment for the skills where they are supported by the more general ontology modules. The semantic approach resulted in the extraction of modules that made sense.

We can conclude the need of commitment to a model. The knowledge base does not remain structured unless its users maintain the structure of the model. It could prove beneficial to research the development of an algorithmic approach to the modularization. This could help the correct placement for future knowledge and help the user maintain an organized knowledge base.

By using the model in this thesis as a base for future skills and proceeding by developing more skills the idea of a skill library seems closer than ever. The proof of concept skill library further showed that this is possible. By allowing skills to be downloaded and uploaded through a library, the skill based approach to the instruction of industrial robots may expand rapidly, both in terms of users and skills. The next step in this process is the development of a proper back-end with standardized upload and download features. This will result in a knowledge base with maintained structured knowledge and may relieve the effort for a strong commitment to the model.

A large amount of the existing knowledge in the LTH knowledge base is in need of maintenance and documentation. This would lead to much easier usage which in turn would lead to further development.

Bibliography

[1] Gruber, Thomas R. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):pages 199–220, 1993. doi:10.1006/knac.1993.1008.

[2] RoSta Project. `http://www.robot-standards.eu/index.php?id=8`. (Visited on 04/25/2015).

[3] Nilsson, Anders, Muradore, Riccardo, Nilsson, Klas, and Fiorini, Paolo. Ontology for Robotics: a Roadmap. In *Proc. ICAR: 2009 14th International Conference on Advanced Robotics, vols 1 and 2*, pages 291–296. IEEE, 2009.

[4] Haage, Mathias, Malec, Jacek, Nilsson, Anders, Nilsson, Klas, and Nowaczyk, Slawomir. Declarative-knowledge-based reconfiguration of automation systems using a blackboard architecture. In *Proc. 11th Scandinavian Conference on Artificial Intelligence, Eds: Anders Kofod-Petersen, Fredrik Heintz and Helge Langseth, IOS Press*, pages 163–172. 2011.

[5] Stenmark, Maj, Malec, Jacek, Nilsson, Klas, and Robertsson, Anders. On distributed knowledge bases for robotized small-batch assembly. *Automation Science and Engineering, IEEE Transactions on*, 12(2):pages 519–528, 2015.

[6] Jung, Carl Gustav and von Franz, Marie-Luise. *Man and his symbols*. Laurel, 1968.

[7] Jovchelovitch, Sandra. *Knowledge in context: Representations, community and culture*. Routledge, 2007.

[8] Noy, Natalya F, McGuinness, Deborah L, and others. *Ontology development 101: A guide to creating your first ontology*. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, 2001.

[9] Mungall, Christopher J., Torniai, Carlo, Gkoutos, Georgios V., Lewis, Suzanna E., and Haendel, Melissa A. Uberon, an integrative multi-species anatomy ontology. *Genome Biol*, 13(1):page R5, 2012.

[10] Ashburner, Michael, Ball, Catherine A., Blake, Judith A., Botstein, David, Butler, Heather, Cherry, J. Michael, Davis, Allan P., Dolinski, Kara, Dwight, Selina S., Eppig, Janan T., and others. Gene Ontology: tool for the unification of biology. *Nature genetics*, 25(1):pages 25–29, 2000.

[11] Fernández-López, Mariano, Gómez-Pérez, Asunción, and Juristo, Natalia. Methontology: from ontological art towards ontological engineering. 1997.

[12] Uschold, Mike and Gruninger, Michael. Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(02):pages 93–136, 1996.

[13] Stuckenschmidt, Heiner, Parent, Christine, and Spaccapietra, Stefano. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*. Springer Science & Business Media, 2009.

[14] Parent, Christine and Spaccapietra, Stefano. An overview of modularity. In *Modular Ontologies*, pages 5–23. Springer, 2009.

[15] Borst, Willem Nico. *Construction of engineering ontologies for knowledge sharing and reuse*. Universiteit Twente, 1997.

[16] Stenmark, Maj and Stolt, Andreas. A system for high-level task specification using complex sensor-based skills. In *Proc. Robotics: Science and Systems (RSS) 2013 Workshop on Robot Programming with Constraints*. 2013.

[17] Stenmark, Maj and Malec, Jacek. Knowledge-based instruction of manipulation tasks for industrial robotics. *Robotics and Computer-Integrated Manufacturing*, 33:pages 56–67, 2015. doi:10.1016/j.rcim.2014.07.004.

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

[19] Stenmark, Maj, Malec, Jacek, and Stolt, Andreas. From High-Level Task Descriptions to Executable Robot Code. In *Intelligent Systems' 2014*, pages 189–202. Springer, 2015.

[20] Pease, Adam. Standard upper ontology knowledge interchange format. *Unpublished language manual. Available at http://sigmakee. sourceforge. net*, 2004.

[21] Stenmark, Maj. *Instructing Industrial Robots Using High-Level Task Descriptions*. Licentiate Thesis, Dept. of Computer Science, Lund University, 2015.

[22] Schutter, Joris De, Laet, Tinne De, Rutgeerts, Johan, Decré, Wilm, Smits, Ruben, Aertbeliën, Erwin, Claes, Kasper, and Bruyninckx, Herman. 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):pages 433–455, 2007. doi:10.1177/027836490707809107.

[23] Craig, John J. *Introduction to robotics: mechanics and control*, volume 3. Pearson Prentice Hall Upper Saddle River, 2005.

[24] John, Karl-Heinz and Tiegelkamp, Michael. *IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids*. Springer Science & Business Media, 2010.

[25] Bruyninckx, Herman and De Schutter, Joris. Specification of force-controlled actions in the “task frame formalism”-a synthesis. *Robotics and Automation, IEEE Transactions on*, 12(4):pages 581–589, 1996.

[26] Hodgson, Ralph and Keller, Paul J. QUDT-quantities, units, dimensions and data types in OWL and XML. *Online (September 2011) http://www. qudt. org*, 2011.

[27] Rijgersberg, Hajo, van Assem, Mark, and Top, Jan. Ontology of units of measure and related concepts. *Semantic Web*, 4(1):pages 3–13, 2013.

[28] Carbonera, J.L., Rama Fiorini, S., Prestes, E., Jorge, V.A.M., Abel, M., Madhavan, R., Locoro, A., Goncalves, P., Haidegger, T., Barreto, M.E., and Schlenoff, C. Defining positioning in a core ontology for robotics. In *Proc. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1867–1872. 2013. doi:10.1109/IROS.2013.6696603.

[29] Rijgersberg, Hajo. *Semantic support for quantitative research*. PhD Thesis, Amsterdam: Vrije Universiteit, 2013.

[30] Prestes, Edson, Carbonera, Joel Luis, Rama Fiorini, Sandro, M. Jorge, Vitor A., Abel, Mara, Madhavan, Raj, Locoro, Angela, Goncalves, Paulo, E. Barreto, Marcos, Habib, Maki, Chibani, Abdelghani, Gérard, Sébastien, Amirat, Yacine, and Schlenoff, Craig. Towards a core ontology for robotics and automation. *Robotics and Autonomous Systems*, 61(11):pages 1193–1204, 2013. doi:10.1016/j.robot.2013.04.005.

[31] IEEE Standard Ontologies for Robotics and Automation. *IEEE Standard Ontologies for Robotics and Automation*, no. 1872-2015, 2015.

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

[33] Andrade, L., Fiadeiro, J. L., Gouveia, J., and Koutsoukos, G. Separating computation, coordination and configuration. *Journal of Software Maintenance and Evolution: Research and Practice*, 14(5):pages 353–369, 2002. doi:10.1002/smr.260.

[34] Vanthienen, Dominick, Klotzbücher, Markus, and Bruyninckx, Herman. The 5c-based architectural Composition Pattern: lessons learned from re-developing the iTaSC framework for constraint-based robot programming. *JOSER: Journal of Software Engineering for Robotics*, 5(1):pages 17–35, 2014.

Appendix A

Snapfit Skill

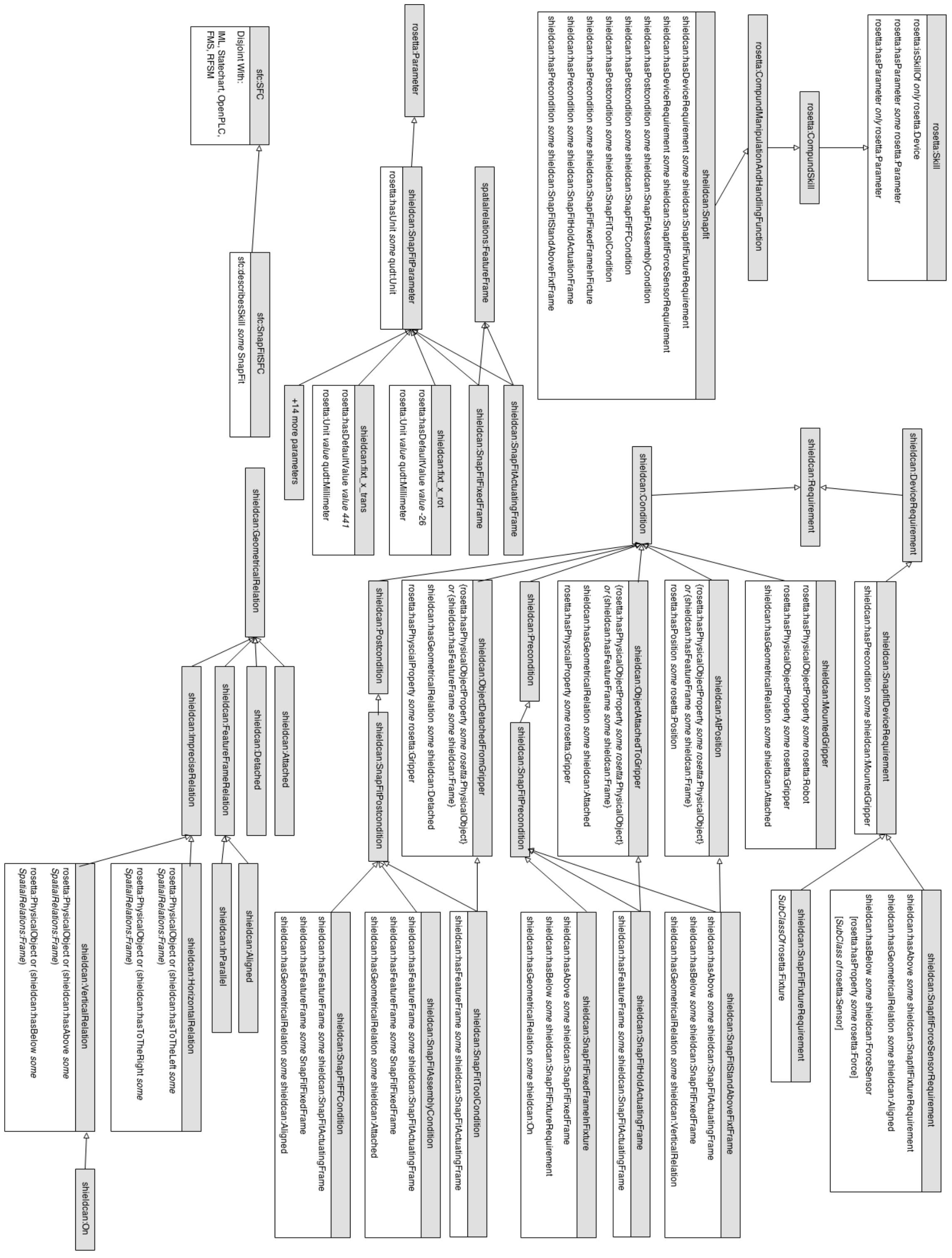

rosetta:Skill
rosetta:isSkillOf only rosetta:Device
rosetta:hasParameter some rosetta:Parameter
rosetta:hasParameter only rosetta:Parameter
rosetta:CompundSkill
rosetta:CompundManipulationAndHandlingFunction
sheildcan:Snapfit
shieldcan:hasDeviceRequirement some shieldcan:SnapfitFixtureRequirement
shieldcan:hasDeviceRequirement some shieldcan:SnapfitForceSensorRequirement
shieldcan:hasPostcondition some shieldcan:SnapFitAssemblyCondition
shieldcan:hasPostcondition some shieldcan:SnapFitFFCondition
shieldcan:hasPostcondition some shieldcan:SnapFitToolCondition
shieldcan:hasPrecondition some shieldcan:SnapFitFixedFrameInFicture
shieldcan:hasPrecondition some shieldcan:SnapFitHoldActuationFrame
shieldcan:hasPrecondition some shieldcan:SnapFitStandAboveFixtFrame
shieldcan:DeviceRequirement
shieldcan:Requirement
shieldcan:Condition
shieldcan:SnapfitDeviceRequirement
shieldcan:hasPrecondition some shieldcan:MountedGripper
shieldcan:SnapfitForceSensorRequirement
shieldcan:hasAbove some shieldcan:SnapfitFixtureRequirement
shieldcan:hasGeometricalRelation some shieldcan:Aligned
shieldcan:hasBelow some shieldcan:ForceSensor
[rosetta:hasProperty some rosetta:Force]
[SubClass of rosetta:Sensor]
shieldcan:SnapFitFixtureRequirement
SubClassOf rosetta:Fixture
shieldcan:MountedGripper
rosetta:hasPhysicalObjectProperty some rosetta:Robot
rosetta:hasPhysicalObjectProperty some rosetta:Gripper
shieldcan:hasGeometricalRelation some shieldcan:Attached
shieldcan:AtPosition
{rosetta:hasPhysicalObjectProperty some rosetta:PhysicalObject}
or {shieldcan:hasFeatureFrame some shieldcan:Frame}
rosetta:hasPosition some rosetta:Position
shieldcan:ObjectAttachedToGripper
{rosetta:hasPhysicalObjectProperty some rosetta:PhysicalObject}
or {shieldcan:hasFeatureFrame some shieldcan:Frame}
shieldcan:hasGeometricalRelation some shieldcan:Attached
rosetta:hasPhyscialProperty some rosetta:Gripper
shieldcan:Precondition
shieldcan:SnapFitPrecondition
shieldcan:SnapFitStandAboveFixtFrame
shieldcan:hasAbove some shieldcan:SnapFitActuatingFrame
shieldcan:hasBelow some shieldcan:SnapFitFixedFrame
shieldcan:hasGeometricalRelation some shieldcan:VerticalRelation
shieldcan:SnapFitHoldActuatingFrame
shieldcan:hasFeatureFrame some shieldcan:SnapFitActuatingFrame
shieldcan:SnapFitFixedFrameInFixture
shieldcan:hasAbove some shieldcan:SnapFitFixedFrame
shieldcan:hasBelow some shieldcan:SnapFitFixtureRequirement
shieldcan:hasGeometricalRelation some shieldcan:On
shieldcan:ObjectDetachedFromGripper
{rosetta:hasPhysicalObjectProperty some rosetta:PhysicalObject}
or {shieldcan:hasFeatureFrame some shieldcan:Frame}
shieldcan:hasGeometricalRelation some shieldcan:Detached
rosetta:hasPhyscialProperty some rosetta:Gripper
shieldcan:Postcondition
shieldcan:SnapFitPostcondition
shieldcan:SnapFitToolCondition
shieldcan:hasFeatureFrame some shieldcan:SnapFitActuatingFrame
shieldcan:SnapFitAssemblyCondition
shieldcan:hasFeatureFrame some shieldcan:SnapFitActuatingFrame
shieldcan:hasFeatureFrame some SnapFitFixedFrame
shieldcan:hasGeometricalRelation some shieldcan:Attached
shieldcan:SnapFitFFCondition
shieldcan:hasFeatureFrame some shieldcan:SnapFitActuatingFrame
shieldcan:hasFeatureFrame some SnapFitFixedFrame
shieldcan:hasGeometricalRelation some shieldcan:Aligned
spatialrelations:FeatureFrame
shieldcan:SnapFitActuatingFrame
shieldcan:SnapFitFixedFrame
rosetta:Parameter
shieldcan:SnapFitParameter
rosetta:hasUnit some qudt:Unit
shieldcan:fixt_x_rot
rosetta:hasDefaultValue value -26
rosetta:Unit value qudt:Millimeter
shieldcan:fixt_x_trans
rosetta:hasDefaultValue value 441
rosetta:Unit value qudt:Millimeter
+14 more parameters
sfc:SFC
Disjoint With:
IML, Statechart, OpenPLC, FMS, RFSM
sfc:SnapFitSFC
sfc:describesSkill some SnapFit
shieldcan:GeometricalRelation
shieldcan:Attached
shieldcan:Detached
shieldcan:FeatureFrameRelation
shieldcan:ImpreciseRelation
shieldcan:Aligned
shieldcan:InParallel
shieldcan:HorizontalRelation
rosetta:PhysicalObject or (shieldcan:hasToTheLeft some SpatialRelations:Frame)
rosetta:PhysicalObject or (shieldcan:hasToTheRight some SpatialRelations:Frame)
shieldcan:VerticalRelation
rosetta:PhysicalObject or (shieldcan:hasAbove some SpatialRelations:Frame)
rosetta:PhysicalObject or (shieldcan:hasBelow some SpatialRelations:Frame)
shieldcan:On

Figure A.1: Description in 3.3.5

Appendix B

SUO-KIF formulation of developed ontologies

B.1 configuration

```
;; CONFIGURATION ONTOLOGY MODULE
;;
;; by: Ludwig Jacobsson ludjac@gmail.com
;;
;; DEPENDENCIES:
;; - CORA:POS
;;   - SUMO
;; - ROSETTA
;;
;; PREFIXES:
;; rosetta: [http://kif.cs.lth.se/ontologies/rosetta.owl]
;; POS: [http://www.inf.ufrgs.br/phi-group/ontologies/coraPos.owl]
;;
;; 1. FRAME
;; 2. PARAMETER
;; 3. REQUIREMENT
;; 4. CORA:POS extensions to fit frames
;; 5. Units and measure from OM-ontology

;;;; 1. FRAME

(subclass Frame rosetta:Concept)
(documentation Frame EnglishLanguage
        "Abstract frame concept.
                A frame to describe an object frame in the task-scene.
                Definition from iTAsC formalism.")

(subclass ObjectFrame Frame)
```

```
(documentation ObjectFrame EnglishLanguage
        "Rigidly attached to an object in the scene")

(subclass FeatureFrame Frame)
(documentation FeatureFrame EnglishLanguage
        "Linked to but not necessarily rigidly attached to and object frame")

(instance hasFrame BinaryPredicate)
(domain hasFrame 1 rosetta:Skill)
(domain hasFrame 2 Frame)
(documentation hasFrame EnglishLanguage
        "To indicate relationship skill<->frame")

;;;; 2. PARAMETER

(documentation rosetta:Parameter EnglishLanguage
        "Skill parameter concept")

;; Each parameter has a unit
(=>
    (instance ?PARAM rosetta:Parameter)
    (exists (?UNIT)
      (rosetta:hasUnit ?PARAM ?UNIT)))

 ;;;; 3. REQUIREMENT
 (subclass Frame rosetta:Concept)
(documentation Frame EnglishLanguage
        "Abstract frame concept.
        A frame to describe an object frame in the task-scene.
        Definition from iTAsC formalism.")

(subclass Requirement rosetta:Thing)
(documentation Requirement EnglishLanguage "A skill requirement")

(subclass Condition Requirement)
(documentation Requirement EnglishLanguage "A skill condition")

(subclass DeviceRequirement Requirement)
(documentation Requirement EnglishLanguage "A device requirement")

(subclass Precondition Condition)
(documentation Requirement EnglishLanguage "A skill precondition")

(subclass Postcondition Condition)
(documentation Requirement EnglishLanguage "A skill postcondition")

;;;; 4. CORA:POS extensions to fit frames
;; Based on CORA:POS but altered to fit Frames
;; FOR MORE INFO SEE: CORA documentation [CORA:POS]

(instance framePosition BinaryPredicate)
(domain framePosition 1 Frame)
(domain framePosition 2 POS:PositionMeasure)
(documentation framePosition EnglishLanguage "Frame position")

;; Each frame has a position
```

```
(=>
    (instance ?FRAME Frame)
    (exists (?POS)
      (framePosition ?FRAME ?POS)))

(instance frameOrientation BinaryPredicate)
(domain framePosition 1 Frame)
(domain framePosition 2 POS:OrientationMeasure)
(documentation frameOrientation EnglishLanguage "Frame orientation")

;; Each frame has an orientation
(=>
    (instance ?FRAME Frame)
    (exists (?ORI)
      (framePosition ?FRAME ?ORI)))

(instance frameRelPosition TernaryPredicate)
(domain frameRelPosition 1 Frame)
(domain frameRelPosition 2 POS:PositionMeasure)
(domain frameRelPosition 3 Frame)
(documentation frameRelPosition EnglishLanguage
        "Relative position of two frames")

(instance frameRelOrientation TernaryPredicate)
(domain frameRelOrientation 1 Frame)
(domain frameRelOrientation 2 POS:OrientationMeasure)
(domain frameRelOrientation 3 Frame)
(documentation frameRelOrientation EnglishLanguage
        "Relative orientation of two frames")


(instance frameRefPCS SingleValuedRelation)
(instance frameRefPCS BinaryPredicate)
(domain frameRefPCS POS:PositionCoordinateSystem)
(domain frameRefPCS Frame)
(documentation frameRefPCS EnglishLanguage
        "Frame reference for position coordinate system")

;; Each Coordinate system has a reference frame
(=>
    (instance ?CS POS:PositionCoordinateSystem)
    (exists (?FRAME)
        (frameRefPCS ?CS ?FRAME)))

(instance frameRefOCS SingleValuedRelation)
(instance frameRefOCS BinaryPredicate)
(domain frameRefOCS POS:OrientationCoordinateSystem)
(domain frameRefOCS Frame)
(documentation frameRefPCS EnglishLanguage
        "Frame reference for orientation coordinate system")

;; Each Coordinate system has a reference frame
(=>
    (instance ?CS POS:OrientationCoordinateSystem)
    (exists (?FRAME)
        (frameRefOCS ?CS ?FRAME)))
```

```
(subclass CartesianCoordinateSystem POS:PositionCoordinateSystem)
(documentation CartesianCoordinateSystem EnglishLanguage
        "Cartesian Coordinate System")

(subclass CartesianPositionPoint POS:PositionPoint)
(documentation CartesianPositionPoint EnglishLanguage
        "A position point for cartesian coordinate systems")

(subclass CartesianX Double)
(documentation CartesianX EnglishLanguage "Cartesian X coordinate")

(subclass CartesianY Double)
(documentation CartesianY EnglishLanguage "Cartesian Y coordinate")

(subclass CartesianZ Double)
(documentation CartesianZ EnglishLanguage "Cartesian Z coordinate")

(subclass SphericalCoordinateSystem POS:PositionCoordinateSystem)
(documentation SphericalCoordinateSystem EnglishLanguage
        "Spherical Coordinate System")

(subclass SphericalPositionPoint POS:PositionPoint)
(documentation SphericalPositionPoint EnglishLanguage
        "Spherical Position Point")

(subclass CylindricalCoordinateSystem POS:PositionCoordinateSystem)
(documentation CylindricalCoordinateSystem EnglishLanguage
        "Cylindrical Coordinate System")

(subclass CylindricalPositionPoint POS:PositionPoint)
(documentation CylindricalPositionPoint EnglishLanguage
        "Cylindrical Position Point")

;; Orientation

(subclass ZYXEulerAngleCoordinateSystem POS:OrientationCoordinateSystem)
(documentation ZYXEulerAngleCoordinateSystem EnglishLanguage
        "ZYX Euler Angle Coordinate System")

(subclass ZYXEulerAngleOrientationPoint POS:OrientationPoint)
(documentation ZYXEulerAngleOrientationPoint EnglishLanguage
        "ZYX Euler Angle Orientation Point")

(subclass ZYXEulerAngleAlpha Double)
(documentation ZYXEulerAngleAlpha EnglishLanguage
        "ZYXEulerAngle alpha coordinate")

(subclass ZYXEulerAngleBeta Double)
(documentation ZYXEulerAngleBeta EnglishLanguage
        "ZYXEulerAngle beta coordinate")

(subclass ZYXEulerAngleGamma Double)
(documentation ZYXEulerAngleGamma EnglishLanguage
        "ZYXEulerAngle gamma coordinate")
```

```
(subclass ZYZEulerAngleCoordinateSystem POS:OrientationCoordinateSystem)
(documentation ZYZEulerAngleCoordinateSystem EnglishLanguage
        "ZYZ Euler Angle Coordinate System")

(subclass ZYZEulerAngleOrientationPoint POS:OrientationPoint)
(documentation ZYZEulerAngleOrientationPoint EnglishLanguage
        "ZYZ Euler Angle Orientation Point")

(subclass ZYZEulerAngleAlpha Double)
(documentation ZYZEulerAngleAlpha EnglishLanguage
        "ZYZEulerAngle alpha coordinate")

(subclass ZYZEulerAngleBeta Double)
(documentation ZYZEulerAngleBeta EnglishLanguage
        "ZYZEulerAngle beta coordinate")

(subclass ZYZEulerAngleGamma Double)
(documentation ZYZEulerAngleGamma EnglishLanguage
        "ZYZEulerAngle gamma coordinate")

;;;; 5. Units and measure from OM-ontology
;; Use of the OM-ontology
;; for more info see: http://www.wurvoc.org/vocabularies/om-1.8/
```

B.2 coordination

```
;; COORDINATION ONTOLOGY MODULE
;;
;; by: Ludwig Jacobsson ludjac@gmail.com
;;
;; DEPENDENCIES:
;; - ROSETTA
;;
;; PREFIXES:
;; rosetta: [http://kif.cs.lth.se/ontologies/rosetta.owl]
;;
;;;; 1. BEHAVIOURDESCRIPTION
;; Unaltered class hierarchy from old SFC.owl ontology
;; imported into configuration.owl
```

Reusable skills for industrial robots

"snapfit" and "peg-in-hole", the future of robot programming. Developing a model for reusable skills used to instruct industrial robots in manufacturing tasks.
Robot skills will enable the app-store like skill libraries of the future.

To drive the development and integration of industrial robots in manufacturing tasks new approaches to robot programming are needed. The current state of robot programming involves trajectory planning and low level instructions for grippers and tools. This results in long development processes and is highly dependent on the expertise of the developer. To remedy this limitation reusable robot skills are introduced. A skill describes a "know how",or capability, of a robot. Each skill is designed to be reusable over different types of industrial robots. By combining these skills the robot can be instructed to solve complex tasks. For example, the task of assembling a table can be achieved by combining the imagined skills, "drill" and "peg-in-hole". This can now be achieved by a user not well versed in the area of robot programming. Perhaps even a computer can reason on the needs of the task and the outcome of each skill and instruct the robot without the need of human interaction. In order to enable this usage of skills for robot instruction each skill need to be packaged and stored with rich structured descriptions and in a way that enables a plug-and-play like behaviour.

We have developed a data model for representing and storing these skills. The model was implemented as an ontology. A ontology is a representation of the entities in a domain of knowledge using classes, attributes and relationships. To achieve machine readable knowledge descriptions we use an ontology to describe the intrinsic concepts of a skill. Each skill is essentially a motion control instruction for the robot along with a structured description of important aspects of the skill. To achieve sufficient packaging and storage our model represents these intrinsic concept describing the skill. Our data model consists of a class hierarchy and class relations describing the skill aspects. Reuse, management and extensions on the current model are enabled by organizing the skill knowledge into smaller knowledge domains using separate ontologies. By isolating the knowledge specific to each skill, skills are treated as reusable items and can be applied in a plug-and-play like manner.

Our work shows that the use and reuse of skills is in the near future. The pluggable feature of the skills inspire us to imagine a future "skill-library" where the users can download and upload skills to accommodate for their needs and solve the tasks in front of them.

Ludwig Jacobsson
2015-06-26
Master's thesis title: A Module-Based Skill Ontology for Industrial Robots
Department of Computer Science
Lund University