Development of an Intermodular Receptacle
A First Step in Creating EAS Modules

Diploma Thesis at the Institute of Applied Computer Science / Automation (AIA)
Prof. Dr.-Ing. habil. Georg Bretthauer
Faculty of Mechanical Engineering
Universität Karlsruhe (TH)

by

cand. mach.
Raphael Adamietz

Supervision:
Forschungszentrum Karlsruhe, Royal Institute of Technology Stockholm,
Institute of Applied Computer Science: Production Engineering Department:
Prof. Dr.-Ing. habil. Georg Bretthauer Prof. Dr. Mauro Onori
Dr.-Ing. Ulrich Gengenbach

Starting Date: 29th October 2007
Completion Date: 14th April 2008
Acknowledgements

I would like to send my gratitude to the staff of the Department of Production Engineering at the Royal Institutes of Technology in Stockholm, which always assisted me during my stay in Sweden. During this work, I collaborated with people from three different countries. Without the assistance of some of those people along way, this work would not have been possible in this form. Therefore I would like to thank

Björn Sandberg, Jan Stamer, Dr. Mats Beijhem and Robert Romejko for their assistance in the lab,

Ann-Sophie Granberg, Marina Öhmann, Margaretha Eneroth and Ingalill Flodmark for their assistance with organisatory issues,

Antonio Maffei, Dr. Kerstin Dencker and Dr. Daniel Tesfaram Semere for encouragement and being open for discussions,

Luis Ribeiro, Pedro Mendes, Pedro Deusdado and Prof. Dr. José Barata for great collaboration, many interesting discussions and hospitality,

Dr. Ulrich Gengenbach for rendering the possibility and supervision of this work,

and Prof. Dr. Mauro Onori, who not only encouraged me persistently, but also was always available as friend, supervisor and teacher and gave me the opportunity to make the experience of working abroad.
Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus anderen Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum.                                                                                             Unterschrift
Abstract

Evolvable Assembly Systems are designed to be truly modular. The realisation of such a system requires a multi-level interface. The goals of this work are to develop such an interface and to deliver the hardware foundation for the conversion of a piece of legacy equipment into an EAS module. Within this work a new concept for an EAS interface has been developed. The proposed solution is called Intermodular Receptacle. Its structure supports the integration of decentralised control paradigms, physical plugability and equips every module with a knowledge base. Furthermore the concept of the skill has been included and further developed. A new process-oriented ontology has been created, that includes the concept of the Intermodular Receptacle and related concepts. Based on the development of the Intermodular Receptacle an attempt to create an EAS module out of legacy components was started. To be able to equip an arbitrary piece of equipment with an Intermodular Receptacle, an integrated computer module was developed. It was applied to a legacy component, a gripper. This effort resulted in the hardware foundation for an EAS compliant gripper module.
Contents

1 Evolvable Assembly Systems 1
   1.1 From Classic to Flexible to Evolvable Assembly ............... 1
   1.2 Principles of Evolvable Production Systems ..................... 2
   1.3 Properties of Evolvable Assembly Systems ........................ 3
   1.4 The Evolvable Assembly Paradigm ................................. 5
      1.4.1 The Evolvable Assembly System Ontology ...................... 6
      1.4.2 The Evolvable Assembly System Control ..................... 8
      1.4.3 Evolvable Assembly System Interfaces ...................... 12
      1.4.4 State of the Art ........................................ 12

2 Project Description ............................................. 16
   2.1 Project Description ............................................ 16
   2.2 Work Objective ............................................... 17

3 Definition of an Intermodular Receptacle 18
   3.1 EAS Interface Aspects ......................................... 18
   3.2 Intermodular Receptacle Definition ............................. 21
   3.3 Information Interface ......................................... 23
      3.3.1 Generic Properties ...................................... 24
      3.3.2 Skills .................................................. 26
      3.3.3 Auxiliary Concepts ..................................... 29
   3.4 Hardware Interface ........................................... 32
   3.5 Social Interface ................................................ 33
   3.6 Process-Oriented Ontology .................................... 35
   3.7 Discussion ..................................................... 37
      3.7.1 IMR Definition ........................................... 38
      3.7.2 Information Interface .................................... 39
      3.7.3 Social Interface ......................................... 39
Contents

A.4 Discussion ......................................................... 77

B Specifications .................................................. 78
   B.1 Auxiliary Concepts ........................................... 78
      B.1.1 Mathematical Tools ..................................... 78
      B.1.2 Non-Mathematical Tools .............................. 84

C Information Interface Instantiation ......................... 90
   C.0.3 General Properties ....................................... 90
   C.0.4 Auxiliary Concepts ...................................... 99
   C.1 Skills ....................................................... 106
      C.1.1 Grab .................................................... 106
      C.1.2 Interface .............................................. 110

Bibliography ..................................................... 111
# List of Figures

1.1 A scheme of the EAS paradigm ........................................ 5  
1.2 Process-Oriented Ontology Overview ............................... 7  
1.3 Multi-Agent control system architecture for EAS .................. 11  
1.4 Rexroth Desktop Factory ............................................. 13  
1.5 Feintool modutech .................................................... 13  
1.6 Rohwedder MicRohCell ............................................... 13  
1.7 An AAA example ..................................................... 13  
1.8 Plug and Produce assembly line ..................................... 15  

2.1 The Project Frame .................................................... 16  

3.1 Relationship between EAS Module, Equipment and IMR ........... 21  
3.2 Intermodular Receptacle Elements .................................. 21  
3.3 The Information Interface Structure ................................ 22  
3.4 Structure of the Social Interface ................................... 33  
3.5 The Merged Ontology Structure ..................................... 36  

4.1 Available Parts ...................................................... 45  
4.2 The Gumstix verdex with expansion boards ......................... 45  
4.3 The Schunk SWS-010 Master and Tool Head ....................... 46  
4.4 Computer Module Open .............................................. 48  
4.5 Computer Space Distribution ....................................... 48  
4.6 LT1076-5 Stepdown Converter Circuit ............................. 50  
4.7 Computer Module With Integrated Power Supply ................... 51  
4.8 Assembled Integrated Computer .................................... 51  
4.9 Module Hardware Design ............................................ 52  
4.10 Module Realisation .................................................. 54  
4.11 Tool Receptacle ..................................................... 55  
4.12 Module in Receptacle ............................................... 55
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.13</td>
<td>Preliminary Storage Facility</td>
<td>56</td>
</tr>
<tr>
<td>4.14</td>
<td>Tool Receptacle</td>
<td>57</td>
</tr>
<tr>
<td>4.15</td>
<td>Module in Receptacle</td>
<td>57</td>
</tr>
<tr>
<td>4.16</td>
<td>Hardware Interface</td>
<td>57</td>
</tr>
<tr>
<td>4.17</td>
<td>E10 Electrical Connector Pin Assignment</td>
<td>59</td>
</tr>
<tr>
<td>4.18</td>
<td>Software Interface Realisation</td>
<td>60</td>
</tr>
<tr>
<td>4.19</td>
<td>Software</td>
<td>62</td>
</tr>
<tr>
<td>4.20</td>
<td>Principal Axis Coordinate System</td>
<td>64</td>
</tr>
<tr>
<td>4.21</td>
<td>Module Work Envelope</td>
<td>65</td>
</tr>
<tr>
<td>4.22</td>
<td>Tool Center Point Coordinate System</td>
<td>65</td>
</tr>
<tr>
<td>4.23</td>
<td>Free Space Between Gripper Jaws</td>
<td>66</td>
</tr>
<tr>
<td>A.1</td>
<td>Handle - Disassembled State</td>
<td>72</td>
</tr>
<tr>
<td>A.2</td>
<td>Handle - Assembled State</td>
<td>72</td>
</tr>
<tr>
<td>A.3</td>
<td>Valve - Disassembled State</td>
<td>73</td>
</tr>
<tr>
<td>A.4</td>
<td>Valve - Assembled State</td>
<td>73</td>
</tr>
<tr>
<td>A.5</td>
<td>Handle Fixture - Empty</td>
<td>74</td>
</tr>
<tr>
<td>A.6</td>
<td>Handle Fixture - Handle Position</td>
<td>74</td>
</tr>
<tr>
<td>A.7</td>
<td>Preliminary Storage Facility</td>
<td>74</td>
</tr>
<tr>
<td>A.8</td>
<td>Handle Fixture Prototype - Empty</td>
<td>75</td>
</tr>
<tr>
<td>A.9</td>
<td>Handle Fixture Prototype - Handle Assembled</td>
<td>75</td>
</tr>
<tr>
<td>A.10</td>
<td>Valve Fixture - Empty</td>
<td>76</td>
</tr>
<tr>
<td>A.11</td>
<td>Valve CAD model</td>
<td>76</td>
</tr>
<tr>
<td>A.12</td>
<td>Valve Fixture Prototype - Empty</td>
<td>76</td>
</tr>
<tr>
<td>A.13</td>
<td>Valve Fixture Prototype - Valve Assembled</td>
<td>76</td>
</tr>
<tr>
<td>B.1</td>
<td>Description of Coordinate System Description</td>
<td>78</td>
</tr>
<tr>
<td>B.2</td>
<td>Description of a point in cartesian coordinates</td>
<td>79</td>
</tr>
<tr>
<td>B.3</td>
<td>Description of a point in cylindrical coordinates</td>
<td>79</td>
</tr>
<tr>
<td>B.4</td>
<td>Description of a point in spherical coordinates</td>
<td>80</td>
</tr>
<tr>
<td>B.5</td>
<td>Description of cuboid implementation</td>
<td>80</td>
</tr>
<tr>
<td>B.6</td>
<td>Description of cylinder implementation</td>
<td>81</td>
</tr>
<tr>
<td>B.7</td>
<td>Description of sphere implementation</td>
<td>81</td>
</tr>
<tr>
<td>B.8</td>
<td>Description of the Coordinate System Transformation</td>
<td>83</td>
</tr>
<tr>
<td>B.9</td>
<td>Description of the Canonical Coordinate Transformation</td>
<td>84</td>
</tr>
<tr>
<td>B.10</td>
<td>Implementation of the Mathematical Function</td>
<td>85</td>
</tr>
<tr>
<td>Number</td>
<td>Figure Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>B.11</td>
<td>Implementation of the Maintenance Data</td>
<td>85</td>
</tr>
<tr>
<td>B.12</td>
<td>Implementation of the Operation Data</td>
<td>86</td>
</tr>
<tr>
<td>B.13</td>
<td>Implementation of the Finite State Machine</td>
<td>87</td>
</tr>
<tr>
<td>B.14</td>
<td>Implementation of Time Information</td>
<td>87</td>
</tr>
<tr>
<td>B.15</td>
<td>Implementation of Electrical Parameters</td>
<td>87</td>
</tr>
<tr>
<td>B.16</td>
<td>Implementation of Hardware Interface</td>
<td>88</td>
</tr>
<tr>
<td>B.17</td>
<td>Implementation of Module Typology</td>
<td>89</td>
</tr>
<tr>
<td>C.1</td>
<td>Interface Coordinate System</td>
<td>100</td>
</tr>
<tr>
<td>C.2</td>
<td>Module Envelope</td>
<td>103</td>
</tr>
</tbody>
</table>
List of Tables

4.1 E10 Electrical Connector Pin Assignment .......................... 59
C.1 Generic Properties - Device Identification - ID .................. 91
C.2 Generic Properties - Device Identification - Module Type ........ 91
C.3 Generic Properties - Device Identification - Module Name .......... 91
C.4 Generic Properties - Device Identification - Serial Number ........ 91
C.5 Generic Properties - Device Identification - Part Number .......... 92
C.6 Generic Properties - Mechanical Interface - CAD Information ........ 93
C.7 Generic Properties - Mechanical Interface - Hardware Interface Slave . 93
C.8 Generic Properties - Mechanical Interface - Module Base CS ....... 94
C.9 Generic Properties - Mechanical Interface - Base CS Origin Point .... 94
C.10 Generic Properties - Environmental Interface - Humidity .......... 95
C.11 Generic Properties - Environmental Interface - Temperature ....... 95
C.12 Generic Properties - Electrical Interface - Electrical Interface Parameters 96
C.13 Generic Properties - Pneumatic Interface .......................... 96
C.14 Generic Properties - Hydraulic Interface .......................... 97
C.15 Generic Properties - Physical Properties - Center of Gravity Point ... 97
C.16 Generic Properties - Physical Properties - Module Mass ........... 98
C.17 Generic Properties - Physical Properties - Moment of Inertia ...... 98
C.18 Auxiliary Concepts - Interface Coordinate System ................ 99
C.19 Auxiliary Concepts - Origin Point ................................. 100
C.20 Auxiliary Concepts - Base Coordinate System Transformation ....... 101
C.21 Auxiliary Concepts - Base CS Origin Point .......................... 102
C.22 Auxiliary Concepts - EnvelopeP1 (Point) ........................... 102
C.23 Auxiliary Concepts - Electrical Interface Parameters ............... 104
C.24 Auxiliary Concepts - Module Center of Gravity (Point) ............. 104
C.25 Auxiliary Concepts - Principal Axis Coordinate System (Coordinate System Description) .............................. 105
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.26</td>
<td>Auxiliary Concepts - Base Coordinate System Transformation</td>
<td>105</td>
</tr>
<tr>
<td>C.27</td>
<td>Skills - Grab - Grab</td>
<td>106</td>
</tr>
<tr>
<td>C.28</td>
<td>Skills - Grab - Force-Displacement Characteristics</td>
<td>107</td>
</tr>
<tr>
<td>C.29</td>
<td>Skills - Grab - Tool Center Point</td>
<td>107</td>
</tr>
<tr>
<td>C.30</td>
<td>Skills - Grab - TCP Coordinate System (Coordinate System Description)</td>
<td>108</td>
</tr>
<tr>
<td>C.31</td>
<td>Skills - Grab - TCP Coordinate System Transformation</td>
<td>108</td>
</tr>
<tr>
<td>C.32</td>
<td>Skills - Grab - Free Space Volume</td>
<td>109</td>
</tr>
<tr>
<td>C.33</td>
<td>Skills - Interface - MasterCS to SlaveCS</td>
<td>110</td>
</tr>
<tr>
<td>C.34</td>
<td>Skills - Interface - Hardware Interface Slave</td>
<td>110</td>
</tr>
</tbody>
</table>
1 Evolvable Assembly Systems

1.1 From Classic to Flexible to Evolvable Assembly

Classic automated assembly systems are designed to assemble one kind of product. As the demands of the markets rose over the years of the 20th century, producers have been forced to react to the increasing speed of change. The development of a new customized production system with the every new product is today more a threat to many companies than a chance, regarding the costs for equipment, staff and the pricing pressure from the market. This leads producers to reutilisation and modification of existing systems and a more generic design. A milestone along the way has been the introduction of flexible assembly systems (FAS) [1]. Generally, a FAS is an assembly system in which there is some amount of flexibility that allows the system to react in the case of changes, whether predicted or unpredicted [2]. With FAS, a company can use the production line for a family of parts, can produce variable sizes of production lots and can do this more economically than classic assembly systems [1]. Research and practical experience have shown that the term flexibility needs to be looked at very specifically as many different levels and types of flexibility have grown in the 40-year history of FMS. I.e. production systems may be flexible in terms of products, processes, volumes or materials. Johansson [3] listed 25 types of flexibility. In doing so he distinguished between two major groups: Static flexibility and dynamic flexibility. In case of static flexibility the system has to be changed offline, implying that the production has to be interrupted. On the contrary, dynamic flexibility means that the system can either adapt to the new conditions itself due to inherent self-knowledge or the adaption is being triggered by a human user. According to [2], the best application of a FAS is found in the production of small sets of products like those in mass customisation. Under certain circumstances, flexibility can also have negative aspects: flexible systems have lower efficiency, they require over-design, generic components, extra interfaces and changeover times [4]. More specific information on FMS can be found in the following literature: [1, 5, 6, 7].
The flexibility approach allows to do diverse tasks with the same installation. This is not optimal in terms of adaptability. To make flexible assembly systems more flexible would reduce the symptoms, but not change the cause: The paradigm. This means, that, in order to take a major step to an optimum in terms of system adaptability, a new way of thinking is necessary. Onori [8] proposes: In terms of applying a reengineering perspective, what is required is not a solution which tries to accomplish all of the envisaged assembly needs within a closed unit (FAS) but, rather, a solution which allows a continuous evolution of the assembly system, being based on several reconfigurable, task-specific, process-oriented elements (system modules). This new approach is known as Evolvable Assembly Systems (EAS).

1.2 Principles of Evolvable Production Systems

The idea of Evolvable Production Systems (EPS) is based on two basic principles. Those imply a major shift of paradigm for the system design.

**Principle of no constraints:** The most innovative product design can only be achieved if no production and assembly process constraints are posed. The ensuing, fully independent, process selection procedure may then result in an optimal production system methodology [9].

To realize a production system following this principle, the highest priority must lie on its functionality [10]. Meanwhile product-related issues may not be forgotten. A way to accomplish this is to split the process into formalised, distinct parts [9]. All dependency on existing assembly system principles is broken [10]. The layout then consists of a broad range of small, process-oriented units. This also requires an analysis of the equipment, which has to be broken down into those smaller, process-oriented components. This may be realized by creating an ontology and a knowledge model of the process [10]. Knowing the requirements of the process, a system consisting of small targeted components may be attainable. On the other side the designer can access a standard set of production components with specific process specifications. This means he knows which technical solutions are available. The technical solution of the production system is therefore dynamically reconfigurable and highly adaptable, which means evolvable.

**Principle of evolution:** Systems in dynamic conditions need to evolve. They need to have an inherent capability of evolution to address the new or changing set of re-
1 Evolvable Assembly Systems

Those requirements can be imposed by a product change, by market or technical demands. The higher the systems' ability to adapt to changing requirements is, the faster the system is able to react to those changes.

1.3 Properties of Evolvable Assembly Systems

An Evolvable Assembly System may dynamically adapt to new products and production scenarios [10]. In order to achieve this, assembly solutions must be designed to integrate any form or type of equipment.

Requirements of different natures are imposed onto the production system by the product design. Designers have to take the available production infrastructure into account which constrains the final product design. On the other hand, the production infrastructure should be as broad as possible and offer as many possibilities as possible. Furthermore the information exchange between both domains should be as good as possible. To assure a certain level a standard way to describe the necessities should be found.

A way to create such a standard is the development of an ontology. An ontology is a catalog of the types of things that are assumed to exist in a domain of interest D from the perspective of a person who uses a language for the purpose of talking about [11]. The term ontology is directly linked to the term knowledge base. A knowledge base is not only a database, it is far more extensive. According to [12] a knowledge base is an amount of facts about the world. Each fact is expressed in a representation language. One can add facts, ask for facts and draw conclusions assisted by an interference mechanism. Such a mechanism is highly suitable for the knowledge representation in EAS.

The in chapter 1.2 mentioned breakdown of the assembly process into distinct sub-processes has to be considered in the paradigm. The by Onori [8] proposed way is based on modularity. Modularity is a concept that has proved useful in a large number of fields that deal with complex systems [13]. A synthesis of many definitions sprinkled throughout the design literature has been compiled by Clark in [13]:

A complex system can be managed by dividing it up into smaller pieces and looking at each one separately. When the complexity of one of the elements crosses a certain threshold, that complexity can be isolated by defining a separate abstraction that has a simple interface. The abstraction hides the
complexity of the element; the interface indicates how the element interacts with the larger system.

In EAS, each module is a task-specific element [14]. The underlying concept is process-oriented modularity [15]. This means that (product and production system) designers do not look for specific equipment in the first place, but for available processes. Predefined interfaces on many levels (mechanical, electrical, control, information, ...) support the EAS concept and simplify system construction. At the same time adaptation and reconfiguration efforts should be minimized; a "plug and play" solution should be aspired at.

In a real system, when one part is added, the additional functionality is not an obvious addition of the added parts functionality [14]. This effect is called emergence. Systems have emergent properties (of varying nature) that are not found in their parts. You cannot predict the properties of a complete system by taking it to pieces and analysing its parts. Similarly, the smaller the constituents of the system, the easier it is to define, structure and coordinate the skills being brought in and out of it [14].

The following properties have been mentioned so far:

- Process-Oriented Modules
- Emergent behaviour
- Product and system design with the help of a knowledge base
- High adaptability
- High reconfigurability

Putting all those properties together results in a fully "reconfigurable" system platform that exhibits an emergent behaviour which introduces new or refined levels of functionality [16]. From the compilation of the previously mentioned properties in a system emerges a new, more complex type of general property, which will from now on be called evolvable.
1.4 The Evolvable Assembly Paradigm

The EAS paradigm has been compiled into a scheme as can be seen in Fig. 1.1. It may be used to describe the approach proposed by Rossi and Maffei [17]:

The starting point is the product. Its assembly creates requirements. When an assembly process is proposed, it has to meet the requirements imposed by the product. The actual description of the process is performed by activities. Activities relate skills to each other. A skill is generally the ability of a machine or a worker to carry out a well specified task in a certain time. The result has to fulfil earlier defined requirements. In EAS, equipment is represented by its skills. Accordingly, a system designer is supposed to look for available skills (in a database) to realise the assembly system. The choice of skills leads the designer implicitly to applicable equipment. It can then be used to design the assembly system.

Equipment meeting the EAS standards is called *EAS module* (or just *module*). If the requested modules are somehow available, i.e. have been designed before and can be
bought or are already in stock (or at least developed), the emplacement\textsuperscript{1} process can take place. This includes system configuration, integration of hardware and the build-up of the assembly system. This can be theoretically regarded like putting LEGO bricks together, some minor adaption work might have to be carried out.

If a requested module is not available, a strategic decision has to be made. One option is to develop the module to make it available. Then the emplacement process can take place. The other option is to redesign either the process and/or the product in order to create a feasible process. The decision should be made upon consideration of economical, technological, financial and other reasonable aspects. The underlying information to decide rationally will come from the knowledge model - the ontology.

This approach is completely different from classic approaches: For Niebel [5] the available equipment is the starting point to develop production systems. Allegri [1] lists which components a flexible production system should include.

1.4.1 The Evolvable Assembly System Ontology

One vital piece of the EAS puzzle is the process-oriented ontology (See 1.4). Ontology is a concept widely used today in Knowledge Engineering and Computer Science in any application that involves knowledge management and information management [16].

Taxonomy is the practice and science of classification [18]. It refers to classification according to presumed natural relationships among types and their subtypes [18].

A very comprehensive definition of the term of ontology can be found in [19]:

In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application. In the context of database systems, ontology can be viewed

\textsuperscript{1}Does not refer to the EUPASS emplacement.
as a level of abstraction of data models, analogous to hierarchical and relational models, but intended for modeling knowledge about individuals, their attributes, and their relationships to other individuals. Ontologies are typically specified in languages that allow abstraction away from data structures and implementation strategies; in practice, the languages of ontologies are closer in expressive power to first-order logic than languages used to model databases. For this reason, ontologies are said to be at the "semantic" level, whereas database schema are models of data at the "logical" or "physical" level.

In the course of this project, a process-oriented ontology has been proposed. It is based on an ontology that emerged from EUPASS. It includes the following top-level terms: "Class of Products", Process, Activity, Skill, General Equipment, Assembly System and EAS Module. The interrelation between those basic terms can be seen in Fig. 1.2. A taxonomy of processes has been included, which distinguishes between many classes of processes. Also a differentiation of basic and complex skills has been included. A basic skill is a operation a single module is able to perform, while a complex skill is a composition of two or more basic skills. One module can perform one skill. Also taxonomies of General Equipment and EAS Modules have been included.

![Figure 1.2: Process-Oriented Ontology Overview](image)

The ontology has been used to show that it is possible create a system design accord-
ing to the EAS paradigm [17]. This proved that it is possible to create a system by the choice of processes which lead the designer to the components. The inherent evolvability should allow an adaption of the proposed system to a second product within 15 minutes by the exchange of modular system components [17].

But however, the ontology is not yet complete and needs to be reshaped. The concept of the Advanced Enabling Interface [10] (or a comparable interface concept) has not been included yet. The link between the concepts of EAS Module, Equipment and Skills is missing. Because of the missing concepts and links it is not yet possible to create a JAVA class representation of the ontology.

### 1.4.2 The Evolvable Assembly System Control

Taking the before described properties of EAS into consideration, the control solution must fulfil certain demands. According to [16], those are:

- Support the integration of modular components that might include their own controllers with different levels of intelligence.
- Support product changes
- Support fluctuations in demand
- Support of the addition and removal of components during normal production
- Provide support for the operative phase

The above mentioned demand require a very high ability of the system to adapt to changes. Barata [20] sees the major challenge in the control solution in how to guarantee proper coordination and execution in a system in which both its components and working conditions can be dynamically changed. Furthermore, Barata [20] claims that this challenge needs a completely new approach and this is why in the context of EPS a solution based on concepts inspired from Complexity Theory and Artificial Life is being developed.

**Complexity Theory** looks for simple causes leading to complex behaviours [21]. A complex system is any system featuring a large number of interacting components (agents, processes, etc.) whose aggregate activity is nonlinear (not derivable from the
summations of the activity of individual components) and typically exhibits hierarchical self-organization under selective pressures [22]. Complex Systems are spatially and/or temporally extended non-linear systems with many strongly coupled degrees of freedom [20].

**Artificial Life** is an interdisciplinary study of life and life-like processes that uses a synthetic methodology [23]. Examples are the mathematical description and implementation of neurons into artificial networks, usually using methods of computer science ("Neural Networks") or the implementation of swarm strategies into robots. Methods of Artificial Life are usually bottom-up approaches: The basic idea is to find the most simple element of a biologic system and to create a complex system out of those simple elements.

**Self-Organization** is the evolution of a system into an organized form in the absence of external pressures. Reason to implement self-organization in EAS are to minimize and facilitate user interaction, i.e. to hide complexity and increase system autonomy [20]. A major challenge in manufacturing applications is to let the system self-organize and at the same time, determine its behavior. EAS may require a kind of leader, a broker or (possibly human) decision maker. The control influence of this authority may be punctual in time and scope, e.g. at important strategic points.

An **Agent** (as referring to a component of software and/or hardware) [24] is capable of acting exactly in order to accomplish tasks on behalf of its user [25]. The goal of multiagent systems’ research is to find methods that allow us to build complex systems composed of autonomous agents who, while operating on local knowledge and possessing only limited abilities, are nonetheless capable of enacting the desired global behaviors [26].

**Autonomic Computing** is an initiative started by IBM in 2001. Its ultimate aim is to create computer systems capable of self-management, to overcome the rapidly growing complexity of computing systems management, and to reduce the barrier that that complexity poses to further growth [27]. In a self-managing Autonomic System, the human operator takes on a new role: He does not control the system directly. Instead, he

---

2Brain: Neuron, Swarm: Find a set of simple rules to create complex behaviour
3A very comprehensive and thrilling way to learn more about complexity, artificial life and multi agent systems is to read Michael Crichton’s Prey.
defines general policies and rules that serve as an input for the self-management process. For this process, IBM has defined the following four functional areas:

- **Self-Configuration**: Automatic configuration of components
- **Self-Healing**: Automatic discovery, and correction of faults
- **Self-Optimization**: Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements
- **Self-Protection**: Proactive identification and protection from arbitrary attacks.

Barata [20] identifies Autonomic Computing as a fundamental concept for evolvable systems. Although Autonomic Computing is designed for pure software systems, the ideas can be projected onto a modular assembly system. First developments in order to implement Self-*-capabilities into EAS have been published [28, 29, 30]. Self-configuration is important for autonomy of the module and can be realised by a distributed control solution [30]. Hardware and software faults can in many cases be detected automatically and in some cases be overcome using adequate methods [28]. For self-optimization a broad spectrum of methods is available [31]. Those methods can be applied on low level (e.g., automatic optimum parameter calculation of a robot) or on high level (e.g., automatic optimum assembly cell design). Self-Protection is important to protect the equipment investment cost, to assure proper system functionality and to protect human beings from hazards. It includes measures to protect the software execution, like e.g., firewalls, or hardware protection (like e.g., collision detection).

The concepts of Complexity Theory, Emergence, Artificial Life, Self-Organization and Autonomic Computing are necessary for the comprehension of EAS. Multi-Agent Systems are a way to deal with the occurring properties of EAS. In [20] a Multi-Agent based control approach for EAS has been proposed (Fig. 1.3). It has been implemented into the assembly cell solution NovaFlex. Each module or manufacturing component is considered to be an agent, which then becomes an abstraction of the component including its functionalities and enhanced interaction skills. A basic assumption of this approach is that a system is a composition of manufacturing components that somehow are aggregated under coalitions (Manufacturing Resource Agent (MRA)) and cooperate to solve the problem they were designed to. A coalition is here an aggregated group of agentified assembly modules (MRA), interacting in order to generate aggregated functionalities.
that in some cases are more complex than the simple addition of their individual capabilities [30]. A coalition can represent complex assembly modules, such as stations as they are compositions of agentified assembly modules [30]. Each component equipped with an agent has an inherent description of its abilities or functionalities, called skills. This way each module is being characterized and an information exchange between the modules becomes feasible. The concept includes four different kind of agents:

- **Agent Machine Interface (AMI):** An agent that connects the MRA to its physical controller by offering to the MRA the functionalities existing in the physical component.

- **Manufacturing Resource Agent (MRA):** An agent which allows a manufacturing component to participate in a society of agents.

- **Broker:** An agent that coordinates the shop floors coalitions.

- **Coalition Leader Agent:** An agent that leads the coalition and coordinates the other agents actions.
The realisation of the Novaflex cell, shows that the agent paradigm to a module based assembly system is adequate. This system shows already a high evolvability though it has been built from mostly legacy components.

A multi-agent control approach for manufacturing systems has been introduced by [32]. The proposed system also includes concepts of modularity. In combination with the human-machine-interface software solution a high reconfigurability could be achieved.

An extensive description of the here presented control approach and the used terms can be found in [33].

1.4.3 Evolvable Assembly System Interfaces

An interface is a preestablished way to resolve potential conflicts between interacting parts of a design. It is like a treaty between two or more subelements. To minimize conflict, the terms of these treaties - the detailed interface specifications - need to be set in advance and known to the affected parties [13].

As stated in 1.2, the interface indicates how the element interacts with the larger system. EAS consist of distinct modules, which are as independent as possible, with distributed intelligence. Furthermore the control concept includes methods that require self-knowledge. Therefore the modules do not only need mechanical and electrical interfaces definition, but also a way for the module software to share and access self-knowledge. The computer-based ways of information exchange also require a standard form of interface definition. The summary of those interfaces can be aggregated to an abstract interface [10, 34]. Proposals for such interfaces exist in the literature, like the Advanced Enabling Interface [10] or the emplacement [35].

1.4.4 State of the Art

Rexroth’s Desktop Factory (Fig. 1.4) is a modular assembly cell [36]. The respective system distinguishes between process modules and plug-in modules (standardized dimensions). Process modules provide mostly standardized assembly processes (e.g. pick and place, dosing), while plug-in modules provide other processes (like e.g. feeding). The cell is based on a standardized structural frame design with receptacles for 4 process modules and 3 plug-in modules. Each process module includes a workpiece holder
Figure 1.4: Rexroth Desktop Factory

Figure 1.5: Feintool modutecc

Figure 1.6: Rohwedder MicRohCell

Figure 1.7: An AAA example
transfer unit with two lanes. A pneumatic system allows to switch lanes and to put workpiece holders (WTs) onto defined positions where processes may take place. The flexibility is very high due to the conveyor system which allows parallel processes and not necessarily requires a process order, as one WT may "overtake" others. The modules are quite independent, as each module is equipped with its own control cabinet. The system is at least in mechanical terms very adaptable. There is no information on the control system concept (except that it is centralised) and hence reconfiguration times.

Feintool's **modutec** is also a modular assembly platform (Fig. 1.5) [37]. Like the Desktop Factory it consists of a frame with receptacles. The modules include control cabinets. The control solution is centralised, but also modular due to standard interfaces. The manufacturer claims to achieve an equipment reusability of 70% and a reduction of change-over times of 50%.

Rohwedder’s **MicRohCell** is a micro assembly cell (Fig. 1.6). It has no conveyor system. The basic idea behind it is to have a working space, which can be used by several types of process modules. The standardized interfaces allow changeover times in the dimension of minutes. Due to the modularity the cell may evolve together with available processes. This is very suitable for micro assembly technology, a process-driven domain.

The first system where modularity has been combined with a multi-agent control system has been presented with the **Agile Assembly Architecture** (AAA) [38]. The proposed minifactory is a system where every modular component has its own computer power and is represented by an agent in the control system (Fig. 1.7). The scalability and reconfigurability, due to the modularity and the multi-agent system have been ground-breaking in 1997. It shows however the big advantages of multi-agent architectures in assembly systems and the suitability for modular systems.

Another assembly solution that features properties of EAS are **Plug and Produce** assembly lines by ETA SA, a Swatch Group company specialised onto the production of clockworks and their parts. Three types (manual, semi-automatic and automatic) of plug-in units (modules in EAS nomenclature), equipped with multi-level standard interfaces, can be attached to the line transport system. Each plug-in unit is supposed to be as autonomous as possible. Parts of the control system are included into the plug-in units. The architecture allows to change the layout with a low time and manpower
A line is currently in use in the ETA plant in Sion and shows the suitability of a highly modular system for small to medium production lots and a high demand of product variety (Fig. 1.8).
2 Project Description

2.1 Project Description

This thesis work is carried out in the frame of an international project. The cooperation partners are:

- Electrolux Italy
- KTH Stockholm, Evolvable Production Systems Group
- New University of Lisbon (Uninova), Group of Robotics and Intelligent Manufacturing

The project frame is shown in Fig. 2.1. The main research efforts within EAS are carried out within two European projects. $A^3$ is a research project carried out by automobile companies. EUPASS is a joint effort of microtechnology and white goods
industry companies. Within $A^3$ control systems and methodology are developed, while within EUPASS systems, control and methodology are being developed. This work is being carried out within the Electrolux project. It is a side-project of EUPASS and focused on the development of the EAS interface and if legacy equipment can comply to the EAS requirements.

Electrolux posed the challenge of solving an assembly test case (See chapter A.1 for details). The solution is supposed to be achieved by converting a legacy cell into a system that has EAS features. The base system will be the former ABB/TUFF cell (see [17] for details), which is located in Stockholm.

The solution is also supposed to show modular properties. Modularity is directly linked to interfacing. Therefore a major focus within lies on the further development of the Intermodular Receptacle.

Beside major mechanical modifications of the cell, a decentralised control solution is to be aspired at. This part of the work will be carried out by Uninova.

### 2.2 Work Objective

One objective of this thesis is to further develop the concept of the Intermodular Receptacle. This includes a general definition of the term, the connections to previously carried out work on EAS, the general role of the IMR in EAS and a proposal of a generic Intermodular Receptacle template.

The second objective is the realisation of an EAS module out of legacy components. This does not include programming work. Moreover is the goal to provide the necessary hardware and structured information to allow an implementation of an as-autonomous-as-possible module into the project’s demonstration cell.

Furthermore a participation in the Elektrolux project and especially in the development of the demonstration cell and its connected issues is desirable.
3 Definition of an Intermodular Receptacle

3.1 EAS Interface Aspects

Every piece of equipment has certain properties. Some of them enable the equipment to perform or to assist at technical operations leading to the process goal. On the other side every piece of equipment has functionality constraints on several levels. Those can be of environmental nature, e.g. ambient temperature constraints, or technical process requirements, like the repeatability of a linear axis. Moreover aspects like lifetime or maintenance cycles have to be taken into account. The design of assembly systems requires the awareness of both the abilities as well as the constraints. Automating the system design process requires therefore a structured knowledge base including the equipments properties, comprehensive for operators or designers as well as for software services. If the technical properties were static and no runtime access to the properties necessary it would make sense to save everything in one large database. But such a centralised database contradicts the evolvability paradigm.

Taking the aspects of evolvability into account, where should those knowledge bases be? The concept of autonomous modules, with high reconfigurability and the ability to control emergent behaviour on the horizon imply knowledge bases on module level, not higher and not lower. The link between the knowledge bases and equipment is a prerequisite. So why should knowledge be merged and put onto higher level? This would result in less autonomous modules, as other system components than the module itself require updates when it is introduced to the society of modules. Knowledge bases on levels lower than module level require more effort to reconfigure, as the granularity becomes finer. Therefore it is also not to aspired at.

---

1 This refers to the emplacement process introduced in the process-oriented ontology (See 1.4.1 for details)
To allow an automated or at least operator assisted automatic design, it is important to assure the completeness of the information (for the regarded task). Additionally the information has to be processed automatically and therefore to exist in a predefined form. A way to achieve this is the definition of templates. This way both mentioned aspects can be taken care of. Additionally, templates structure information [39]. The request of information from the knowledge base becomes distinct, when both sides are aware of the structure. An example is a library: It is something different if a customer asks for a book about cars, or if he asks for "automobile" book 2 in shelf 3a. In the first case neither the question is distinct, nor will be the answer. In the second case by awareness of the structure by both sides, they have created an interface for information exchange. EAS Modules will also be equipped with such an information interface.

The module choice, at buildtime and at runtime, depends on the module’s ability to perform skills and constraints imposed by general properties. A thoughtful choice requires a set of informations, that allow one to see the possible options and, hopefully, lead to a logical choice. This means designers and operators need to have access to module information prior to putting them on stream.

Aiming at adaptability requires that as many components as possible fit together. This imposes standard hardware interfaces, covering all relevant mechatronic aspects. Minimizing the amount of used interface types leads to a higher plugability and therefore adaptability on system design. In order to allow system designers to create new modules, by modifying legacy equipment, or by developing new ones, clear interface specifications are necessary. This way the designers know how to develop a module fitting into the standard. The specifications should also describe approaches and show examples.

Decentralised control implies intermodular communication. In EAS, an arbitrary number of more or less intelligent entities is supposed to perform a process. Most likely one of those entities is not able to perform the process alone. This means the entities have to organise their actions in some way and this shall happen autonomously: Self-Organisation is required. In self-organizing systems, pattern formation occurs through interactions internal to the system, without intervention by external directing influences [40]. The rules specifying interactions among the system’s components are executed using only
local information, without reference to the global pattern. This means for EAS that for every entity in the system a set of rules needs to be specified. When the entities act, they do it according to those rules. The action can and should be based on a local knowledge, when external information is required, the rules indicate to contact the respective entities to obtain the information.

An EAS production system has no predefined process chain. Moreover the system adapts its behaviour dynamically to the current situation. Social Interaction is a dynamic, changing sequence of social actions between individuals who modify their actions by their interactive partners. The module interaction in this case has quite similar requirements. Modules shall interact like persons in a society. Technical systems are usually supposed to be as effective as possible, therefore the modules should speak a common language in order to enable communication and to avoid misunderstandings. The ability to participate in a society implies the skill to perform social actions. They can be defined as actions that take into account the actions and reactions of other individuals and are modified on those events. Successful social interaction furthermore requires the ability to share information between the modules. The ability of a module to participate in a society of modules can be described as social interface. The multi-agent control paradigm supports the idea of interaction within a society of individual agents [30]. A deeper insight into social interaction in technical systems realised by multi-agent technology can be found in [41].

The summary of the interfaces should represent a single point of control for the module. The complexity of the module is hidden for interactors, input and output are realised through the interfaces. It is not the interior of the module that is interesting, but its behaviour. In this kind of perception a module may very well be considered as a black box.
### 3.2 Intermodular Receptacle Definition

Standard interfaces are a fundamental requirement for modularity. EAS modules need to conform to a certain standard, otherwise the requirements of EAS system cannot be met. The summary of module interfaces determines if it is able to interact with the larger system in the designated way and is called *Intermodular Receptacle*.

![Figure 3.1: Relationship between EAS Module, Equipment and IMR](image)

The *Intermodular Receptacle* is a complex interface, that enables a piece of equipment to conform to the EAS standards (Fig. 3.1). It includes the summary of physical interfaces (Hardware Interface), a standardized, structured knowledge base (Information Interface) and enables the module abilities of perception, action, goal directed behavior, information sharing in a common language and collective problem solving through participation in a society of entities (Social Interface) (Fig. 3.2).

![Figure 3.2: Intermodular Receptacle Elements](image)
Figure 3.3: The Information Interface Structure
3 Definition of an Intermodular Receptacle

3.3 Information Interface

A structured, local representation of knowledge inside the modules is required. The information should be exchangeable and implementable into earlier proposed knowledge representations, like the process-oriented ontology (Chapter 1.4.1). This requirement is identified in [10]. As described before the knowledge representation should form a template that prestructures and demands specific information.

The included knowledge should provide the necessary information to realise a module with evolvable functionality. This includes the demand of autonomy. It can be realised with the implementation of the prior described Self-∗-capabilities as described in chapter 1.4.2. The internal knowledge base content is required to provide the necessary information for the description of processes. Self-configuration, self-optimization, self-protection and self-healing, a.s.o. require manifold information. The module knowledge base is supposed to provide this required information in order to allow the functionality to feature the module.

In order to obtain an solution, that allows to exchange information with the process-oriented ontology, a new ontology was created with Protégé. Its advantages to other ontology editors are the graphical visualisation and editing possibilities [42]. The GraphViz plugin allows to browse classes and global properties, while the jambalaya plugin enables nested graph views with editing. Another useful feature is the possibility to merge ontologies. If necessary, the ontology can be exported in XML or OWL formats.

Fig. 3.3 shows an overview of the included concepts in form of a taxonomy. The ontology consists of three main branches: Generic Properties, Skills and Auxiliary Tools. The Generic Properties include information independent of module class. The concepts are not directly linked to a process and are common information that every module is supposed to carry. The Generic Properties can be described as the least common denominator of all modules in the sense of information. Skills on the other hand include informations that are necessary to perform a process. Skills are split into Simple Skills and Complex Skills. Simple Skills denote distinct (sub-)processes usually directly linked to a piece of equipment and cannot be further split into subprocesses. Complex Skills are composed of two or more skills. The third branch, Auxiliary Tools, contains elements that help to describe the concepts in the other two branches.
While the Generic Properties and the Auxiliary Tools branches contain both over 20 concepts, a skill remains one concept. Therefore the respective branches have the correspondent size. Although Auxiliary Tools have been used for the description of both, Generic Properties and Skills, but the Generic Properties contain simply are lot more information. This is intended. Not all fields in the Generic Properties templates are compulsory to fill out, but it is supposed to contain every kind of information that might be useful in the lifetime of the module, while a Skill is supposed to contain the necessary information for just one type of process.

A detailed description of the included elements can be found in the appendix.

### 3.3.1 Generic Properties

The Generic Properties have been subdivided into a classes, that denote their function. To describe every element in detail would not be appropriate at this point, therefore an overview may be given by a brief description:

**Environmental Interface** is the information necessary to perform checks if the ambient conditions are suitable for the module to work properly. At the moment allowable temperature and humidity ranges have been included. The concepts of temperature and humidity do not denote a property of the module, but the range of environmental conditions they are designed to work in. If the provided sensor information is outside the range, the module is able to refuse work. Future implementations could also include electric and magnetic field issues or basically any environmental conditions.

**Physical Properties** describe the basic mechanical properties of the module. Those include the mass, the principal axis coordinate system of the module, the moments of inertia and the center of gravity. This information is relevant for the calculation of dynamic parameters. For example a robot module is able to calculate new dynamic parameters autonomously due to the information exchange with a tool module.

**Mechanical Interface** provides information about interaction with other modules and the environment. With the Module Base Coordinate System every module is
equipped with at least one coordinate system. A description of the *envelope* of the module has also been included. Thus collision detection or space distribution in automated system design become feasible. Furthermore the link to a *CAD model* has been included. This way more sophisticated geometric model than the envelope can be used for further calculations or interaction with human operators. Moreover *Hardware Interface Slave*, an instantiation of *Hardware Interface* has been included. It is supposed to create the link between the hardware and the information interface. It provides a coordinate system in the module and declares the interface type.

**Electrical Interface** stores the parameters of electrical power supply and allows to check if voltage, current or type of power supply are available or suitable. It may be chosen between three types of power supply: AC, DC and Three Phase Current, by instantiating the respective *Electrical Parameters* class. A functionality that the module checks the electrical parameters or even negotiates them with its host becomes feasible. If the electrical parameters are outside the allowed range, the module should deny the power supply. This can be realised by a switch in the module controller.

**Pneumatic Interface and Hydraulic Interface** describe the parameters of compressed fluid supply. If the provided fluid supply is outside the allowed range, the module will be able to refuse work and to tell the problem to the system.

**Device Identification** gives the module a unique name in the system and provides information about the module class and its manufacturer at the same time. This information may be applied for communication, maintenance issues or identification.

**Monitoring and Maintenance** provides all information that concerns the modules current status and its past. The *Uptime* indicates how long the modules’ computer system has been online, while the *Idle Time* shows how long the module’s actions have not been requested. The *Maintenance Log* provides the past concerning necessary repair and maintenance actions (uses auxiliary tool *Maintenance Data*). Software disfunctionality and the module’s ability to overcome those errors is being represented by the *Number of Self-Healing Operations*. *Mean time to Repair* and *Mean time between maintenance* exhibit the frequency of necessary human operator operations. The *Number of Unhandled Events* displays when the module
unable to perform requested operations. The *Operations Log* is a collection of instantiations of the whole knowledge base and is thus a thorough documentation of module activities (uses auxiliary tool *Operation Data*).

### 3.3.2 Skills

Every module in EAS systems is supposed to perform processes. The systems’ processes related in time and space add up to the assembly process. The description of those processes inside the information interface can be found inside skill class. A module may have multiple skills, but must have at least one defined skill. Providing an interface for module connection will also be considered as a skill. The generic properties forms are identical for every module. The skills are very module-specific. E.g. the ‘grab’-skill of a gripper needs very different information compared to the ‘feed’-skill of a bowlfeeder. Defining a new form for every piece of equipment is not practicable in many ways. Therefore the skill description has to be just as specific as necessary, but as generic as possible. A skill can be directly linked to the assembly process, e.g. pick and place. But also an indirect assembly process connection is possible. A service robot with a skill that maintains other modules during operation is an example. Parallel execution of skill chains is possible and absolutely necessary in complex assembly systems. Every skill, except of the first to be executed, has a predecessor. This way process chains can be realised. If two skills have the same predecessor they must be executed in parallel. Multiple chains of processes are possible. The coordination of those is a control system issue and must not be included into the ontology.

It is important to distinguish between the skill template and the skill itself. The ontology includes the template. When a module offers the service of performing a process, it offers a skill. The template describes parameters that are necessary for the execution of the skill.

**Simple Skill:** The description of a single process a module is able to perform. It is linked directly to a module. A module must offer the system at least one skill, otherwise it cannot be considered as a module.

**Complex Skill:** The composition of two or more skills. Complex skills relate skills, simple or complex, in time and space.

As examples two skill templates have been created. The elements may be briefly described:
The Grab Skill

What information is necessary to accomplish an automated use of a gripper? This question shall be answered with the grab skill. It mostly consists of information that can be retrieved from data sheets, but it also contains important information that concern control issues, like a finite state machine that provides the system status or the tool center point that is necessary for robot trajectory planning. The here presented skill is applicable for most available gripper types, independent if two, three or four fingers and independent of the mechanical realisation.

Skill Name (String): An unique skill instantiation name making it clearly identifiable for human and non-human interactors.

Force-Displacement Characteristics (Instance of Mathematical Function): The mathematical description of the force between the gripper jaws in dependency of the gripper displacement and, if applicable, other parameters (e.g. system pressure for pneumatic gripper).

Gripper Status (Instance of Finite State Machine): The instantiation of the finite state machine provides information about the current status of the gripper and establishes a link between the module software (the agent) and the information interface. Sensor information can be used to indicate the status of the gripper. Possible statuses could be for example "open", "close" or "controlled position" (sensor based closed-loop control of displacement).

Jaw CAD Information (String): A link to gripper jaw CAD files. The 3D model is at the moment not supposed to be a base for collision detection or other complex calculations, but is for technical personel to check which jaws are installed without having to see the hardware.

Maximum Displacement (Float): The gripper displacement difference of a point in open and in closed state without grabbing a workpiece.

Repeatability (Float): The expectable variation of displacement values. This value is usually datasheet information.

Response Time Opening (Float): The time in s the gripper needs to open after it has been triggered.
Response Time Closing (Float): The time in s the gripper needs to close after it has been triggered.

Free Space Volume (Instance of Volume): Due to the primitive geometrical volumes used to describe the work envelope, a collision detection in the space between the open gripper jaws is not possible. It is then considered as massive material. This volume describes this free space and thus allows automatic trajectory calculation for move and grab processes. By defining multiple volumes in this slot a more complex free space can be defined. This is applicable for grab processes using the outer surfaces of the jaws.

Tool Center Point (TCP) (Instance of Point): It marks an appropriate virtual point between the gripper jaws that represents an important parameter for robot trajectory planning.

TCP Coordinate System (Instance of Coordinate System Transformation): A coordinate system starting in the tool center point. It is useful for the free space starting volume description.

Grab Argument (Instance of Argument): This slot denotes the requested functionality. At moment it is possible to chose between openGripper and closeGripper. Future implementations could also include a specific indication of a displacement or a specific force.

Displacement (Instance of Point): For every gripper the current position can be represented by a suitable point on the gripping surface. The coordinates of the respective point are calculated according to sensor information. If the gripper is not equipped with sensors the slot is left empty.

The Information Interface instantiation is for the case of a gripper valid for one gripper/jaw combination. If the jaws are changed, the instantiation needs to be changed.

The Interface Skill

Information which interfaces a module provides shall be provided with this skill. By now it is kept as simple as possible to allow the necessary functionality and not to make the first test phase more complicated than necessary. In the future geometrical information, accuracy issues, locking forces, etc. might be included. This is mostly information that
can retrieved easily by the manufacturer and can be implemented without bigger efforts. It includes a single element:

**Hardware Interface Master (Instance of Hardware Interface):** This declares the hardware interface type and a coordinate system in the interface. In connected state master and slave coordinate systems are supposed to be identical in position and orientation.

The service behind the Interface Skill should check if electrical power supply parameters, compressed air or fluid pressures or simply payload, a.s.o. are inside the allowed range. A feature that the service can decide about activation of the physically connected module is technically feasible. This could mean, that electrical power ports or compressed air will be activated after collecting the necessary information. Otherwise the access could be denied and possible hazards avoided this way.

### 3.3.3 Auxiliary Concepts

One of big advantages of the object-oriented ontology approach is that objects might be reused. This fact comes very much in handy in this case. A problem of spoken and written language is that between sender expression and receiver perception is always a difference. In a technical system this may cause emergence, but most likely of an uncontrollable kind. A precise definition of those elements is necessary. These specifications are important for programmers to know and to follow them. E.g. cartesian coordinate systems can be left-handed and right-handed and there are many possibilities to transform a coordinate system into another one.

The elements of the auxiliary concepts have been splitted into a mathematical and a non-mathematical part. The mathematical part includes the following elements:

**Coordinate System Description:** To define a coordinate system in the knowledge base this class has to instantiated. It includes the transformation specifications to another coordinate system, a description string of the coordinate system purpose and the origin point. The purpose of the *Origin Point* is the definition of the coordinate system type.

**Coordinate System Transformation:** In order to describe the relationship of one coordinate system to another one, the *Coordinate System Transformation* has been
3 Definition of an Intermodular Receptacle

included. If necessary, first the source coordinate system is being transformed into cartesian coordinates (by the Canonical Coordinate Transformation), then the actual translation, described a translation vector and euler angles may take place.

**Canonical Coordinate Transformation** The definition of multiple coordinate system types implies that a way to transform the coordinates from one coordinate system into another must be defined. Therefore canonical coordinate transformations must be defined in the system. Cartesian coordinates will be regarded as the reference type coordinates. For every coordinate system type the transformation description to cartesian coordinates and from cartesian coordinates to the respective coordinate system type has to be defined. The corresponding vectors have been described.

**Mathematical Function:** The description of mathematical coherences makes it necessary to find a way of description. It should practicable in the first place. The planned JAVA implementation lead to a solution that uses the manifold features of the object-oriented programming language. The mathematical function includes a string, where the mathematical function may be described with the help of JAVA instructions.

**Point:** With cartesian, cylindrical and spherical points three types of points have been implemented. Every type of point may be used solely in the respective coordinate system type.

**Volume:** To make the description three-dimensional structures easier, three types of volumes have been designated. With Cuboid, Sphere and Cylinder models simple models for e.g. collision detection may be created.

In the non-mathematical part the following parts have been included:

**Maintenance Data:** A template for an automated or manual maintenance operation on the module, including a text description, an ID of the operator, a time stamp and information about the repaired parts and the time spent on the operation.

**Operation Data:** To log the operations of the module, instantiations of the performed skills, a time stamp and an instantiation of the finite state machine have been included.
**Finite State Machine:** Module interaction, especially communication, make status information necessary. Therefore a finite state machine will be developed, which describes the actual status of the module. So far, just a status string has been implemented.

**Time Information:** The complexity of the intermodular communication makes time information fundamental in order to reconstruct dialogues. Therefore time and date are being saved within instantiations of this class.

**Electrical Parameters:** With the parameter description of AC, DC and three-phase current three types of power supply have been accounted for. They allow to specify voltage and current ranges, as well a suggested voltage value.

**Hardware Interface:** Every module needs to be equipped with one of the systems’ standard hardware interfaces. Mating modules can be identified with an instantiation of this class, due to the *Interface Type* instantiation. A mathematical description of the interface connection is delivered by the *Interface Coordinate System* an instantiation of *Coordinate System Description*.

**Interface Type:** This class provides an identification and a description string to clearly identify mating hardware interfaces. It is used in the *Interface*-skill to identify the interface master and in the *Mechanical Interface* class to identify the interface slave.

**Argument:** This abstract class allows to specify the requested functionality inside the skill, by choice of a subclass. At the moment the classes *openGripper* and *closeGripper* are included and have no content. Arguments inside the class are possible and allow to request parameters if they are required. For example a specific displacement could be requested inside the argument class if necessary.

By defining the arguments, further functionality can be implemented into the Grab Skill. E.g. if a gripper would be equipped with a force sensor, a specific argument could be defined. It could include in this case an operating point for the force. With suitable hardware a closed-loop control for the gripping could be realised. The same scheme could reused to include further functionalities.
3.4 Hardware Interface

The purpose of the *Hardware Interface* is to establish a temporary physical connection between a module and a host. The hardware interface has two mating parts, one supposed to be a part of the module host (interface master), the other one is a part of the module itself (interface slave).

The hardware interface is supposed to fulfil the following requirements:

- In connected state, interface master and slave have a defined position and orientation to each other.
- During connection the interface is a static system with no degrees of freedom.
- Defined dimensions of both interface sides with appropriate tolerances.
- Determined type, position and orientation of connection elements.
- Specified force application points and therefore controlled transfer of mechanical energy.
- Specifications of the static and dynamic stress the module is able to bear.
- Position and type descriptions of electrical, pneumatic and hydraulic connectors.

The requirements have been compiled with the help of [13], [43] and [44].

The hardware realisation of the interfaces should constrain mechanical connection to just one way, the way it has been designed to. This can be realised by appropriate mechanical design. E.g. Poka Yoke is a system of mistake proofing that eliminates defects by simple measures [45]. The locking mechanism can be automated or manual. It should be as easy as possible to lock and unlock the module.

In respect to flexibility and the demand of recombination of modules, the number of interface types should be kept as low as possible. Making one standard connector for all modules is not practical, just like a new connector for every module. This means a trade-off between recombination possibilities and connection constraints needs to be found. The market already offers many physical connector solutions, that fulfil the demands of EAS systems. The eventual choice of the physical connectors is very dependent
of situation, e.g. the product, the production volume, the budget, a.s.o.. Therefore it requires a complex strategic decision, which actual connectors might be applied. The EAS demands are narrowing down the options and are leading to a flexible solution that allows a high, but logic, module interchangeability.

3.5 Social Interface

Figure 3.4: Structure of the Social Interface

Can a module that is controlled from outside be considered autonomous? Most likely not. An autonomous module should make its own decisions and be able to set them into action. Automated decision making requires artificial intelligence and therefore modules need to be equipped with a computer. Another issue of automated decision making is that information is necessary to be able to make logic decisions. The module needs to know what processes are going on and if necessary request more information. This implies that the modules communicate and that they do that in a common language. Especially the processes the module itself performs require to be well perceived. Furthermore the module should be able to access the knowledge base through the information
interface. The control software should be able to connect to module equipment and, if applicable, to its controller. As the system is composed of many entities it needs to be assured, that the composed system is acting in the designated way. Its behavior is supposed to be directed onto a prior defined goal [24, 46].

The basic, generic requirements of the social interface, independent of the control architecture have been worked out with the help of literature about basics of decentralised control paradigms [24, 25, 26, 47, 48] and literature about EAS [10, 20, 30, 34, 46]. The following items have been worked out:

- Communication with other modules in a common language
- Ability to access and update the module knowledge base
- Ability to establish a connection to the equipment controller or to equipment directly
- Ability to make decisions and to set them into action
- Ability to participate in a society of modules in order to enable collective problem solving [24, 46]
- Ability to perceive processes of matter
- Ability to provide process information to other modules
- Goal directed behavior [24, 46]

Those tasks can be carried out by a software agent. It is required however that the agent is linked to the information interface in order to be able to access information and to update the knowledge base if necessary. Another required link the one to the equipment, the agent is supposed to control. [46] describes an agent that is linked to a piece of equipment by adding an intermediate layer called "Agent Machine Interface". The purpose of that layer is to enable communication between the agent and the equipment low-level controller (belonging to "equipment"). It translates instructions that the agent gives to the equipment controller, as well as it translates controller feedback for the agent. Communication between the module agents is also required to be enabled. The structure for the example of a multi-agent system has been compiled in Fig. 3.4.
3 Definition of an Intermodular Receptacle

The social interface offers services to other entities, called skills in EAS nomenclature. The description of the skill in the Information Interface can be regarded as a form that is used to request the execution of the service. Inside the Social Interface, the request is processed in a suitable manner. The Social Interface is able to decide to execute services, upon request and own decision, and to give instructions to the low-level equipment controller. This way the social interface is able to trigger functionality.

The structure that enables communications between the equipment and the agent, that enables the agent to access and to update the knowledge base and that provides communication in a standard language among the agents is the social interface. Physical connections are necessary to fulfil the demands, but are not covered within the social interface. Moreover it is a piece of software. It enables the module to participate in a society of agents, in order to enable collective problem solving. That includes the control of the module action and thus its skills. A protocol for the updates of the knowledge base needs to be defined or chosen, in order to avoid conflicts when more than one entity wants to change content at the same time.

The here described social interface is based on software agents. There are also other approaches like service-based architectures that might work as well as multi-agent systems [49]. In respect to previously carried out work on multi-agent systems within the project a multi-agent based approach will be considered.

3.6 Process-Oriented Ontology

The process-oriented ontology by Rossi and Maffei [17] is not computer-comprehensive. Some links are missing and therefore a port into a programming language is not possible yet. Furthermore the concept of the Intermodular Receptacle has to be included. Therefore a new process-oriented ontology that features the missing properties has been created\(^2\). An overview of the structure can be seen in Fig. 3.5.

The starting point of the ontology is the product design. A product design generates technical requirements. Other requirements, like e.g. marketing, economic or strategic aspects will not be considered so far. To produce a product, a set of transformations

\(^2\)This part of the work was carried out in collaboration with Luís Ribeiro, Uninova Lisbon
3 Definition of an Intermodular Receptacle

Figure 3.5: The Merged Ontology Structure
of input elements into our product is required: a process. It has to meet the technical requirements generated by our product. The description of the process and respectively subprocesses will be performed by skills. Those can be simple or complex. Latter are compositions of simple and complex skills. A description by skills allows to describe a process and its distinct subprocesses. It can be regarded as a logical and structured connection of technical transformations. EAS Modules in the proposed ontology will be equipment plus an Intermodular Receptacle.

As stated previously, skills and generic properties are described within the Intermodular Receptacle. Instead of asking directly for equipment, a description of the required skill will be sent to the knowledge base and a list of the available skills will be the reply. Furthermore a check, if the module’s generic properties of the eventual module are suitable, might be necessary. The arrangement of the skills is not a part of the ontology, but a technical issue, that still needs to be solved.

3.7 Discussion

Prior to this work, concepts of EAS compliant interfaces were described in the literature: The Advanced Enabling Interface [10, 34] and the Emplacement [35]. Information about their development status was not available. The process-oriented ontology by Rossi and Maffei [17] showed links between the EAS concepts, but did not consider an interface. An approach for the control of EAS systems was available in the literature [30].

This was the starting point for a new compilation of requirements and properties for an EAS interface (See chapter 3.1). After having the interface requirements worked out, the definition of the Intermodular Receptacle including its components was formulated. Afterwards, the new development of the information interface was presented and its structure and components detailed. The information interface includes the concept the skill and provides a form that allows other modules to request a certain functionality of the EAS module. Furthermore a new developed form for a broad range of module information that can be used for many functions and features of modules has been implemented with the Generic Properties. As not available before, the specific requirements of the physical connectors for Evolvable Assembly Systems (Hardware Interface) have been compiled with the help of literature. The available control approach was taken
as base for the new definition of the control system requirements. A completely new process-oriented ontology was created, that includes the main EAS concepts, especially the concepts of Intermodular Receptacle and Skill and their relationships.

### 3.7.1 IMR Definition

The term Intermodular Receptacle has been introduced to create a distinction to the "emplacement" and to the "Advanced Enabling Interface". According to [35] the emplacement is a knowledge base inside the module, while in [10] it is an abstract interface. The "Advanced Enabling Interface" (AEI), described in [10] and [34], has common properties with the IMR. An EAS module is formed by enabling three kinds of plugability (control, mechanical, electrical) and the summary is the AEI. The control plugability includes control and social plugability, internal knowledge management and emergence support software. Although the structure is different, the elements included in the IMR and AEI are basically the same. A distinction between electrical and mechanical plugability would also require to include types of plugability like compressed air supply plugability or wireless plugability. The summary into one hardware interface simplifies the definition and is more generic. The definition does not need to be expanded or stretched if further functionality is supposed to be included. The IMR social interface is also more generic from the definition point of view. The Information Interface has similar properties to both "emplacement" descriptions. It has abstract properties, like the specifications or the template structure, on one hand, on the other hand the module knowledge base and the included informations are not abstract and are according to the concept described in [35].

The IMR definition clarifies the included elements and their relationship to the concepts of EAS modules and equipment.

In every technical system a conversion of energy, matter and information takes place [50]. Input and output over the system boundaries are realised by the interface, which also denote the system boundaries. Matter and energy are distinctly being transferred by the Hardware Interface. Information from the physical point of view is also constricted to the Hardware Interface. A part of the information content concerns module properties (Information Interface) and control information (Social Interface).
Both Information and Social Interface influence the exchanged information content. They could be combined in one sub-interface. Two separate parts have advantages. The module development process might require teams out of members with different backgrounds. To fill out the Information Interface form, does at this point not necessarily require a control background. This impression could arise when social and information interface are merged. Furthermore, the information interface access is not exclusive by the social interface. The JAVA representation may allow other interactors to access the ontology directly. Such an access could be made for example to save log files externally or to save the knowledge base content externally to obtain a database of all available modules. Moreover the information might also be accessed by human interactors.

3.7.2 Information Interface
A structured knowledge base with specifications was presented. It provides process information by describing skills and generic module information. Reusable, predefined elements simplify the information description. Manifold possibilities arise by a module carrying internal knowledge. Automatic calculation or acquisition of parameters, automatic trajectory planning, self-configuration, self-diagnosis, self-healing and many more possible features require internal module knowledge.

3.7.3 Social Interface
The social interface structure and its requirements were described. The social interface enables the powerful features of decentralised control systems. The social interface controls the equipment and is able to reconfigure it. This is a step towards autonomous units. By intermodular communication and access to the knowledge base the perspectives of self-reconfiguration and highly adaptive systems become feasible.

3.7.4 Hardware Interface
As the other two interface types, the physical interface must fulfil certain requirements. A set of requirements that lead to EAS compliance was compiled.
3.7.5 Process-Oriented Ontology

The new ontology models the EAS assembly process and links it to EAS modules and the IMR. The description of processes by skills, with inherent information, and the links between them, create direct connections between information interface, control and system design.

3.7.6 Properties and Features

An advantage of the skill description is that the request of functionality is issued with one prior defined form. Within the form a relative small amount of information is transferred. The complexity of control and communication remains hidden for the interactors. This way only high level commands are transferred among the modules and the complexity of the exchanged information decreases significantly. The low level commands are issued in suitable manner by the social interface to the equipment. This property can also be called self-configuration.

The module work envelope can be used for collision avoidance methods. Information about electrical, pneumatic and hydraulic parameters can be used to avoid equipment failure due to wrong parameters of the respective way of energy exchange. The module software should be protected of hostile attacks to ensure proper function. These functions can be regarded as module self-protection.

The mechanical and physical properties in the information interface can be together with sensor information used to optimize processes. If suitable algorithms are used, the assembly system and the modules may optimize themselves. They perform self-optimisation. It could be applied for example in a robot module. Before a tool is connected, the robot and the tool communicate in order for the robot to obtain physical and mechanical properties of the tool. With the information available, the robot as EAS module could be able to calculate optimum motor and control parameters or trajectories. This way process precision and speed could be increased significantly.

To truly maximise the potential profit emerging from evolvable system, device downtime should be reduced [28]. Therefore the ability of a certain device to perform self-diagnosis and take self-healing actions is of major relevance [28]. The Monitoring and
3 Definition of an Intermodular Receptacle

Maintenance class in the information interface has been included with perspective of inclusion of those features. Efforts are currently carried out to develop and implement this functionality inside EAS modules [28, 29].

3.7.7 Intermodular Receptacle vs. EAS Requirements

The definition states, that the IMR converts a piece of equipment into a EAS module. Is this true? A confirmation can be obtained by checking the requirements.

According to [13], modularity is based on the concepts of abstraction, information hiding and interfaces. A specific piece of equipment becomes a module, when IMR is added. This is abstraction. Mass, energy and information are only transferred through the defined channels by the three interfaces. The complexity, thus information, of the module remains hidden for interactors. Conclusion: The IMR supports modularity.

The whole EAS paradigm is process-oriented. The information interface for instance, provides information about available processes by skills. The agent in the social interface triggers the equipment to execute them. A product assembly is considered a complex chain of processes. The complete relationship is described within the process-orientation ontology (3.6). Conclusion: The IMR supports process-orientation.

Does the IMR support the idea of product and system design with the help of a knowledge base? The information interface provides information concerning run- and buildtime. The information interface knowledge base might need to be adapted when the global knowledge base is being developed. Nonetheless, the process information and furthermore the interface and the monitoring and maintenance information are valuable for the creation and regular refreshment of the global knowledge bases. Conclusion: The IMR supports product and system design with the help of a knowledge base.

Adaptability is the ability to adjust easily to a new environment or different conditions [51]. The control of the assembly system needs to have a very high adaptability in order to behave according to the dynamical conditions of the system. In [30] the compliance of a distributed control solution has been shown. The modularity allows to change the system physically with low effort if necessary. The grade of adaptability is defined by module granularity. The social interface is based on a distributed control solution and the matter of modularity has been discussed above. Conclusion: The IMR supports
Reconfigurability can be described as the ability to rearrange available system components to perform new, but pre-defined operations [14]. The information interface provides those predefined operations as skills and their templates to the society of entities and human interactors. They can be arbitrary arranged. Through the social interface the execution of the process is controlled. The execution can be controlled from outside the assembly system or by the system itself. Conclusion: The IMR supports reconfigurability.

The Intermodular Receptacle supports all concepts that are necessary to obtain an evolvable system. An arbitrary piece of equipment requires to be equipped with an IMR receptacle to become EAS compliant. Now the prerequisites for the creation of EAS modules exist and can be applied.
4 Development of an EAS Module

4.1 Boundary Conditions

In the last chapter the component, that makes equipment an EAS module was identified: The Intermodular Receptacle. It has been described, the next logic step is the application.

A cell layout, that may solve the two test cases was presented earlier [17]. In order to progress with the Electrolux project, a newly developed module should be applicable inside the cell. Another boundary condition concerns the equipment: Within the Electrolux project, legacy equipment shall be converted in order to become EAS compliant. A limitation of this work is, that only the hardware shall be provided. The software programming and implementation will be carried out by Uninova Lisbon.

The goal is to show how an arbitrary piece of equipment can be modified and made EAS compliant. Furthermore the application of the Intermodular Receptacle shall be demonstrated.

4.2 Equipment

As this is the first attempt to apply the Intermodular Receptacle, it is desirable to recognize the basic problems. With the complexity of the equipment, the application of the IMR also becomes more complex, as many very equipment-specific problems might arise. The proposed layout requires several modules to be realised: four feeder modules, a robot module and several gripper modules [17].

Within the cell an ABB IRB140 robot [52] and an ABB S4C+ controller [53] are available. They are together a very complex system. That the control system can be modified
in order to obtain an agent-based control has been shown before [46]. But, however, a modification in order to make the robot a modular component from the hardware point of view would exceed by far the frame of this thesis.

The feeders would not be too complex in the context of this work to modify. In the case of the drum feeding solution it would be necessary to develop the module from scratch, as the solution is very specific and no comparable hardware is available in the Production Engineering Department workshop at KTH. The bowl feeders are relatively simple equipment, especially compared to the robot. Two legacy bowl feeders were available in the workshop.

Several Schunk two finger parallel grippers (PGF series) and the Schunk SWS system (See chapter 4.2.2 for details) were available. The parts are shown in Fig. 4.1. Furthermore several adapter plates, that allow a mechanical connection between the SWS tool exchange system and the PGF grippers, were found in the cell. This is considered legacy equipment. Like the bowl feeders, the grippers were not equipped with a computer. The creation of a gripper module had several advantages compared to the other potential module types. The hardware was already modular in mechanical terms and therefore not much modification effort is to expect. Moreover, many gripper modules are necessary for the assembly cell and they can not be simulated (in mechanical terms) easily, as feeders can. The need for a gripper module is higher than for the feeder modules. Some of the potential features, like usage of the mechanical property information by the robot or automated module exchange can not or just with significantly higher effort be included.

After weighing the arguments, the logic choice is to create a gripper module.

The available grippers are not equipped with computers. Therefore a way to apply computer power to a gripper module must be found. If possible, this solution should have the potential to be reused in other modules or module types. The applied computer should not constrain the functionality of the module, but enable new functionality.
4 Development of an EAS Module

Figure 4.1: Available Parts

Figure 4.2: The Gumstix verdex with expansion boards
4.2.1 The Gumstix Verdex Computer

Applicable control architecture for Evolvable Assembly Systems [30] requires computer power inside the modules in order to realise the decentralisation of the control. Particularly with regard to the conversion of a legacy gripper into an EAS module, the computer should be as small and as lightweight as possible. At the same time, the computer should have the necessary processing power to host the control, have networking capability and sufficient storage memory. Furthermore a link to the controllable equipment should be possible to establish.

The gumstix verdex XL6P (Fig. 4.2) is a single board computer with the dimensions of $80 \times 20 \times 5.5 \text{mm}$. To enable Ethernet communication and the use of a Compact Flash memory card an additional circuit board can be connected (netCF-vx). The XL6P has a Marvell PXA270 600Mhz processor with ARM architecture, 128Mb RAM and 32Mb of flash memory. The computer comes with preinstalled Linux. The Robostix allows to connect via serial connection to a computer to access the operating system via console. The Robostix is also equipped with several I/O pins. All requirements posed onto the computer can be coped with the gumstix verdex. Therefore it will be used to equip the gripper with "intelligence".

4.2.2 The Schunk SWS Tool Exchange System

Figure 4.3: The Schunk SWS-010 Master and Tool Head

Already available in the KTH Production Engineering workshop was the The Schunk SWS-010 Tool exchange system. It can be therefore considered to be legacy equipment.
An end-effector with two mating parts normally called a Master Side and Tool Side that have been designed to lock or couple together automatically and are able to pass utilities such as electrical signals, pneumatic, water, etc. is called automatic tool changer. The Schunk SWS-010 is such a system. It offers six compressed air feed-throughs. The optional E10 electrical feed-through has ten pins for electrical power supply and/or electronic signals. The manufacturer suggests a payload limit of 16kg. The locking force is 1100N, while the positional repeatability is 0.01mm. The picture shows the SWS-10 master on the left and a tool head on the right (Fig. 4.3).

4.3 Module Realisation

4.3.1 Computer Module Design

The gumstix circuit boards are not cased. The integration into the gripper module requires a suitable case in order to protect the computer hardware and to be able to create an appropriate mechanical connection to the gripper module. A certain collision resistance and good heat transfer properties of the case are desirable. If possible, a power supply should be integrated inside the case. This way the computer itself becomes a modular component.

Gumstix offers plastic cases for the verdex and netCF-vx combination. The material is thin, the case therefore light, but gives a flimsy impression. The plastic insulates heat instead of distributing it. The case dimensions are tolerated in millimeters and, depending on the used case, the circuit boards are either loose or jammed. Furthermore it requires an external power supply. The summmary of the mentioned arguments leads to the conclusion that the gumstix cases are not suitable.

A reasonable alternative is the Fischer Elektronik AKG series. A case consists of two identical aluminium profiles and two aluminium front plates. The connection is realised by four M3 screws. The used material, AlMgSiO 5F22, is suitable for mechanical cutting, as the material is a wrought, thus hard, alloy and produces short chips. The material heat transfer properties ($\lambda_{\text{Aluminium}} = 221 \frac{W}{m \cdot K}$) as well as the mechanical properties are considered as more than sufficient.
4 Development of an EAS Module

The two circuit board fit into the case after some mechanical processing. This can be seen in Fig. 4.4. The netCF-vx circuit board is clamped in the bottom part. In Fig. 4.5 the space distribution within the case is demonstrated. The case (a) encloses the two circuit boards (b). The netCF-vx is equipped with a receptacle (c) for a Compact Flash card (d).

4.3.2 Power Supply

The remaining space inside the case can be used to integrate a power supply. The power supply solution should be small, as the space inside the case is very limited. Also, to reduce heat, the efficiency should be as high as possible. Furthermore the number of external components should also be low, as they require space. Preliminary testing showed that the circuit board combination requires an input voltage between 3.3V and 5V and a current between 300mA and 800mA.

As available in most industrial systems (and the former ABB/TUFF cell), a power supply input voltage of 24V was chosen. Linear and switching regulators are available as integrated circuits. Both are standard parts and can be bought for prices under 10€. Two electrically appropriate chips are the LM7805 linear regulator and the LT1076-5 switching regulator. The chips have a fixed output voltage $U_{out}$ of 5V, the system input voltage $U_{in}$ of 24V would be inside the intended range and both ICs are able to provide
an output current $I_{out}$ of at least 1A. Both chips require a comparable amount of auxiliary components to realise the circuit boards.

The heat dissipation is dependent of the electrical parameters. The operating junction temperature $T_j$ in $^\circ C$ is given by [54]:

$$ T_j = T_a + P_d \theta_{ja} $$

with $T_a$: Ambient temperature in $^\circ C$; $P_d$: Power dissipated by device in W; $\theta_{ja}$: Thermal resistance from junction to ambient air in $^\circ C/W$

For a linear regulator $P_d$ the following equation is applicable [54]:

$$ P_{d,LR} = (U_{in} - U_{out})I_{out} $$

Assumption of the electrical parameters (worst case assumption) with $U_{in} = 24V; U_{out} = 5V; I_{out} = 0.8A$ and insertion of the values results in:

$$ P_{d,LM7805} = (24V - 5V)0.8A = 15.2W $$

The resulting junction temperature for an ambient temperature $T_a = 20^\circ C$ and the datasheet information [55] for $\theta_{ja,LM7805} = 65^\circ C/W$ is:

$$ T_{j,LM7805} = 20^\circ C + 15.2W \cdot 65^\circ C/W = 1008^\circ C $$

Linear Technology, the manufacturer of the LT1076-5, provides an equation for the calculation of the power dissipation [56]:

$$ P_{d,LT1076-5} = U_{in}((7mA + 5mA \cdot \delta + 2I_{out} \cdot t_{sw} \cdot f) + \delta(I_{out} + 0.1\Omega) \cdot (I_{out})^2) $$

with Duty Cycle $\delta = \frac{U_{out} + 0.5V}{U_{in} - 2V}$, effective overlap time $t_{sw} = (60ns + (10ns/A))(I_{out})$ and switching frequency $f = 100kHz$.

The power dissipation for the same electrical parameters as above is then:

$$ P_{d,LT1076-5} = 0.708W $$

The junction temperature is then with $\theta_{ja,LT1076-5} = 65^\circ C/W$ and ambient tempera-
ture $T_a = 20^\circ C$:

$$T_{j,LT1076-5} = 20^\circ C + 0.708W \cdot 65^\circ C/W = 66.02^\circ C$$

According to the calculations the linear regulator requires serious heatsink measures, while the switching regulator can be used without any heatsink. With the high component density in the computer module, the temperature development of the LM7805 is unacceptable. Due to comparability in the other important aspects the switching regulator is in this case the chip of choice.

The circuit board has been designed in accordance with the manufacturer’s application notes [56]. The following elements have been designated:

$C_1 = 220\mu F$: A local input bypass capacitor is required for buck converters because the input current is a square wave with fast rise and fall times. This capacitor is chosen by ripple current rating. The capacitor must be large enough to avoid overheating created by its equivalent series resistance. The capacity has been chosen because of the resistance of frequencies above 10kHz.

$C_2 = 470\mu F$: The output capacitor is necessary to filter output ripple voltages. Dimension because of frequency resistance.

$C_3 = 0.033\mu F$: This is a part of the oscillating feedback circuit. The IC is able to sense changes of the input and output parameters and adjusts its switching behavior.
appropriately. Dimension according to manufacturer’s specifications.

\( D_1 \): A MBR340 Schottky diode is used to generate a current path for \( L_1 \) current when the LT1076-5 switch turns off. Must be dimensioned to the maximum output current.

\( L_1 = 100 \mu H \): The inductor acts as both an energy storage element and a smoothing filter. The dimension is a manufacturer proposal.

\( R_1 = 1.5k\Omega \): Part of the oscillating feedback circuit. Dimension according to manufacturer’s specifications.

The power supply has a minimum input voltage \( U_{in,min} \) of 8V and a maximum input voltage \( U_{in,max} \) of 30V. The maximum output current \( I_{out,max} \) is 2A. The output voltage \( U_{out} \) is 5V ± 0.5% and independent of input voltage.

### 4.3.3 Computer Module Assembly

![Figure 4.7: Computer Module With Integrated Power Supply](image)

![Figure 4.8: Assembled Computer](image)

Fig. 4.7 shows the module without the top part, but with integrated power supply. In order to fit into the case a twin circuit board realisation was necessary. Input and output capacitors are located next to the integrated circuit in order to minimize input voltage spikes for voltages below 12V [56]. On Circuit Board 1 (CB1) the LT1076-5, \( C_1, C_2 \) are implemented. Furthermore, the input and output connector cables are connected here. Circuit Board 2 (CB2) includes \( D_1, L_1, R_1 \) and \( C_3 \). CB1 will be placed between the RJ45 receptacle and the Compact Flash slot, while CB2 will be on the other side of
the Compact Flash Card slot. Insulation is provided by plastic foil between the CB1 and the RJ45 connector and on top of the CF card slot. To model the mechanical properties of the power supply, two cuboid blocks have been included into the CAD model at the proper positions.

The assembled integrated computer module can be seen in Fig. 4.8. It represents an embedded computer system, with a 600Mhz CPU, 128Mb RAM, 32Mb of flash memory, a Compact Flash card slot, 100Mbit Ethernet and integrated power supply. The choice of the BNC connector for power supply was made because of the availability and the mechanical robustness. Autodesk Inventor 2008 calculates the weight of the computer at 112g, while measurements with a mechanical balance result in 115±5g. Due to the comparability of the values the model is considered as appropriate for the calculation of further mechanical properties.

### 4.3.4 Gripper Module Design

The availability of several Schunk 2 finger parallel grippers (PGF series) and of the Schunk SWS system was the starting point of the gripper module design (Fig. 4.1). Furthermore several adapter plates, that allow a mechanical connection between the SWS and the PGF grippers, were available. This is considered legacy equipment.
The design has been created with the components presented in Fig. 4.9. The Schunk SWS-10 adapter is connected to an adapter plate. By two aluminium elements a space between a second adapter plate is constructed. Further four M6 countersunk screws connect the plate-sandwich to the gripper and assure proper mechanical position between SWS and gripper. The free space between the adapter plates is used for the placement of the computer black box. It is fastened with three M6 countersunk screws. Eight of the ten electrical feed-through pins are used for ethernet, the other two are designated for the computer’s electrical power supply. Two tubes are connecting SWS air feed-throughs with the respective gripper connectors. The four storage pins serve as a geometrically defined storage option. They are placed in the four unused air feed-through holes. In the CAD model two example jaws have been put onto the gripper.

In order to avoid complicated measurements, the position of the SWS relative to the gripper should be as defined as thorough as possible. Translatory displacements can be measured and are usually corrected in the control system, especially for the use with robots [57]. Rotary displacement values are usually more difficult to obtain. The parallelity of the adapter plates is for all available plates under 0.01 and therefore expectable displacement far under the tolerance values of the robot [52]. To be on the safe side, the module TCP should be measured and information included into the knowledge base. However, all screws should be fastened with a dynamometric screwdriver with a torque of 4.6Nm to avoid unsymmetrical tension that might lead to displacement.

The storage pins are ISO 10462 M5 screws with a length of 50mm and sawed off heads. They are an economical alternative to 150 times more expensive Schunk pins. Each pin is locked with a M5 nut. In [58] either the use of a special adapter plate, with threaded holes for pins, or the use of air feed-throughs for storage pins is advised. The usage of those adapter plates would make the module more expensive and the design would become more complicated and heavier. Therefore the air feed-through solution was chosen. If more than two air feed-throughs would be necessary, four threaded holes could be drilled into the upper adapter plate.

4.3.5 Gripper Module Realisation

The realisation of the module is presented in Fig. 4.10. The SWS-010 adapter is an older version than the one in the CAD model, but it is 100% compatible to the SWS-010 head.
The interfaces are compatible. Also the hole pattern and the connectors are identical. The electrical feed-through has a different shape, but the pin pattern remained with the version change.

For the ethernet connection a Cat5e-cable has been cut and soldered into the pins. The unshielded cable parts have been covered with antistatic foil in order to provide electromagnetic shielding: Four layer SCC1000 static shielding bag [59] material has been wrapped three times around the open cables. The power supply cable with BNC connector has likewise been covered.

The computer module was implemented as described in chapter 4.3.3. The gripper is a Schunk PGF-64 from 1994.

4.3.6 Tool Warehouse

The product part diversity makes the use several gripper modules necessary. The tool exchanger simplifies this process. But nonetheless the modules need to be stored in a predefined position, in order for the robot to pick it up and release it.

Schunk offers together with its tool exchangers storage solutions [58]. For the case of the SWS-010, the manufacturer suggests a three pin solution that imposes the use
of a special adapter plate or a four pin solution, that uses four or the six compressed air feed-throughs and does not require any further hardware than the pins. For both solutions the same receptacle, mounted onto a standard aluminium profile structure is offered.

Due to the available machinery in the Production Engineering Department of KTH it was possible to create an in-house solution. To be reasonable, the production costs should be significantly lower than the Schunk solution. And, if necessary, it should be possible to produce further receptacles with low effort.

The KTH Production Engineering workshop has a three-axis CNC mill. This has been considered during mechanical design. The starting point of the design was a rectangular shaped aluminium profile with dimensions of $50 \times 25\text{mm}$. The angle between the here interesting feed-throughs is $50^\circ$. The feed-through holes point onto the center point of the central hole. The module position in storage state is being defined by wedge-shaped clearances that, with the help of gravity forces, center the pins. The collectivity of the clearances defines position and orientation of the whole module. Fig. 4.11 shows one of two necessary tool receptacle parts. Fig. 4.12 shows how the module is supposed to be stored by the receptacle. The connection to the host structure is realised by two 8.5mm holes.

A preliminary tool warehouse has been created of a aluminium profile system using 45$x45\text{mm}$ profiles (Fig. 4.13). It may be not described in detail here. It has been connected to cell interior base plate and the receptacles have been mounted using two M8
screws for each receptacle.

The ABB robot, equipped with tool exchange master, has been used to test the receptacle. In Fig. 4.14 a coordinate system is described. It has the purpose to describe translational and angular displacements.

Tests how much displacement the storage facility can compensate have been accomplished. To obtain the center point, the module was put into the receptacle and then picked up by the robot using the tool exchanger. The robot was calibrated prior to the test and set into manual mode. The module was displaced in translatory in x- and y-direction and rotated in $\varphi$-direction. From various positions the module was released by the tool exchanger from a height of 20mm above the receptacle surface. From one point, the module was dropped 20 times. If from one point at least one drop-off was unsuccessful, the point was considered unreliable. The result is presented in Fig. 4.15.

The vector $\vec{d}$ denotes the translatory displacement in respect to the geometrical center point of the receptacle. Its absolute value ($|d| = \sqrt{x^2 + y^2}$) may not exceed 7mm, or the module will most probably not fall into position. The angular displacement is described by $\varphi$. How much angular displacement the storage facility is able to bear is determined by the translatory displacement. For $\vec{d} = 0$ the maximum reliable $\varphi$ value (that was tested) was $10^\circ$. For $|\vec{d}| = 7.5mm$, values for $\varphi$ up to $7.5^\circ$ worked. Fig. 4.15 shows this
4.4 Hardware Interface

Fig. 4.16 shows the Schunk SWS-010-E10 gripper change system. It represents the module hardware interface. In linked state, the connector faces, colored in blue, are pressed against each other. Orientation and position are defined by two centering pins,
The locking mechanism ensures that the required forces are being provided. It consists of three balls (green) caged in inside the master side. A mechanism triggered by a pneumatic cylinder presses the balls out. They apply an axial force onto the interface slave. This way the necessary locking force is applied.

- The centering pins with the connector faces define position and orientation in locked state.
- The connected system has no degrees of freedom.
- Dimensions and tolerances have been determined by the manufacturer [58].
- The manufacturer also determined type, position and orientation of connection elements [58].
- Radial Forces are transferred form-fit through the centering pins, while axial forces are transferred through the connector faces. The transfer of forces is therefore determined.
- Specifications of the bearable static and dynamic stress are given in [58].
- The interface has no hydraulic connectors. The Pneumatic connector positions, type and specifications and the dimensions and specifications of the electrical E10 connector are described in [58]. The pin purposes are described in Fig. 4.17 and Tab. 4.4. The SWS-010 provides six feed-throughs, of which four are used on the adapter side for the storage pins. The remaining feedthroughs are used for compressed air.

The storage pins are also a part of the hardware interface. The physical interface between the module and tool warehouse, as well as its suitability as hardware interface will be discussed in detail later. The here proposed hardware interface fulfils the requirements described in chapter 3.4.

### 4.5 Social Interface

The hardware that may host an agent, and thus enable a social ability, is implemented within the computer module. The necessary software to obtain the IMR is shown in
4 Development of an EAS Module

Figure 4.17: E10 Electrical Connector Pin Assignment

<table>
<thead>
<tr>
<th>Pin</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ethernet 10/100 BaseT TX+ (white/green)(TIA 568A)</td>
</tr>
<tr>
<td>2</td>
<td>Ethernet 10/100 BaseT TX- (green)(TIA 568A)</td>
</tr>
<tr>
<td>3</td>
<td>Ethernet 10/100 BaseT RX+ (white/orange)(TIA 568A)</td>
</tr>
<tr>
<td>4</td>
<td>Ethernet 10/100 BaseT (blue)(TIA 568A)</td>
</tr>
<tr>
<td>5</td>
<td>Ethernet 10/100 BaseT (white/blue)(TIA 568A)</td>
</tr>
<tr>
<td>6</td>
<td>Ethernet 10/100 BaseT RX- (orange)(TIA 568A)</td>
</tr>
<tr>
<td>7</td>
<td>Ethernet 10/100 BaseT (white/brown)(TIA 568A)</td>
</tr>
<tr>
<td>8</td>
<td>Ethernet 10/100 BaseT (brown)(TIA 568A)</td>
</tr>
<tr>
<td>9</td>
<td>DC Electrical Power Supply (8-30V)</td>
</tr>
<tr>
<td>10</td>
<td>Electrical Power Supply Ground</td>
</tr>
</tbody>
</table>

Table 4.1: E10 Electrical Connector Pin Assignment
Fig. 4.18. It uses a customized Linux distribution as operating system. On the computer runs Jam VM 1.5.1, a JAVA virtual machine, which conforms to the JVM specification version 2 [60]. It is extremely small, with a stripped executable of only 200K. However, unlike other small VMs it is designed to support the full specification. It is therefore suitable for running JADE (Java Agent DEvelopment framework). JADE is a completely distributed middleware system with a flexible infrastructure allowing easy extension with add-on modules. The framework facilitates the development of complete agent-based applications by means of a run-time environment implementing the life-cycle support features required by agents, the core logic of agents themselves, and a rich suite of graphical tools. As JADE is written completely in Java, it benefits from the huge set of language features and third-party libraries on offer, and thus offers a rich set of programming abstractions allowing developers to construct JADE multi-agent systems with relatively minimal expertise in agent theory [61]. Java VMs are available for a large bandwidth of computer architectures and operating systems. Programs written in JAVA can be executed on every computer, that runs a Java VM. This is a huge advantage for the modular paradigm, as software can be developed once and then be run, after customisation effort, on every computer hosting a compliant Java VM. In other cases, the software would be necessary to be developed from scratch.

A Linux kernel has been compiled with, amongst other things, Jam VM. The resulting file system image has been copied onto a gumstix XL6P/netCF-vx combination. Then
JADE 3.5 could be executed from a prepared compact flash on the gumstix computer system. Therewith the computer module is suitable for running a JADE-based multi-agent system\(^1\).

Two other software elements are required to establish a complete social interface: an agent and an AMI. None of the elements have been customized for this project and this module yet. Agent and AMI have to conform the prior described requirements (See 3.5 for details). The module knowledge base should be, in order to enable the information interface, based on the module. It is purposeful to run the knowledge base within the same virtual machine as the agent and AMI.

The development of the Social Interface for this case and its components is not finished yet. Therefore the realisation might change. However, the structure and the requirements of the Social Interface must be met by the control software.

### 4.6 Information Interface

With the creation of the ontology, an extensive form has been compiled. It has been filled for the presented hardware representation. The complete instantiation, with a description of how the values have been obtained, can be found in the appendix C. In this chapter some of the more complex slots may be described and demonstrated.

The instantiation is only valid for the installed gripper jaws. A modification or new instantiation is necessary if other jaws are installed, as e.g. the tool center point, work envelope, the free space volume or the mechanical parameters become different with the jaw exchange.

#### 4.6.1 Generic Properties

Essential elements necessary to describe mechanical concepts are coordinate systems. To define a coordinate systems, the type (cartesian, cylindrical, a.s.o.), the position of the origin point and the orientations of the unit vectors are required. If one coordinate system is set, others may be described with respect to it, by definition of coordinate

\(^1\)This long, probably annoying but finally successful battle has been fought out by Pedro Mendes at Uninova Lisbon.
system transformations. For each EAS module a coordinate system must be defined in order to provide compulsory physical information.

The parameters of at least one coordinate system within the module information interface needs to be described by geometrical attributes. A logical point to assign the coordinate system to is the tool change system. The tool change adapter will be used for numerous modules. Therefore the description might be reused.

Fig. 4.19 shows how a geometrically defined coordinate system may be described. $A_1$ is the center axis of the concentric hole. $A_2$ is the central axis defined by a centering pin, as shown in the picture. $E_1$ is the plane defined by the interface contact surface (red). The origin point of the coordinate system is the intersection point of $E_1$ and $A_1$, the x-axis crosses the intersection point of $E_1$ and $A_2$ and the y-axis is truly parallel to $E_1$. The axis directions are like shown in the picture. The description in the ontology has been realised with "Schunk SWA-010-E10 Coordinate System", the instantiation of interface coordinate system. If the corresponding geometrical attributes are used for the coordinate system description of the master side, the coordinate system transformation from master to slave coordinate system becomes trivial.
The geometrically defined coordinate system is important because of the determination of physical parameters, that require to be defined even when the module is not connected. An example is the module work envelope. It might be applied in build time, e.g. for space approximations, as well in runtime for e.g. collision detection. Otherwise one (by geometrical attributes) defined coordinate system inside the whole cell would be sufficient. The other coordinate systems could then be referenced directly to the "world" coordinate system.

Additionally to the interface coordinate system, for every module a module base coordinate system is to be defined. If available, it should be referenced to another coordinate system, preferably the interface coordinate system. Otherwise it can also be defined by geometrical attributes (e.g. for the cell). For the here presented module, the base coordinate system and the interface coordinate system are equal.

To obtain values of physical properties is often a complicated task. Most values can be either calculated or measured. While calculations often require approximations and the obtained values are therefore not very exact. Measurements require special equipment and are therefore expensive. Modern CAD software, like Autodesk Inventor 2008, is able to assist in obtaining those values. If the effort to make a detailed, accurate model is undertaken, the software is able to calculate mass, center of gravity, moment of inertia and principal axis coordinate systems. Furthermore dimension measurement and retrieval of point coordinates are standard features of CAD software. The model eligibility was tested by weighing the module and comparing the values. The CAD model had a virtual weight of 1103 g, while the mechanical balance measurement resulted in a mass of 1070 ± 5 g. The variance is under 3% and the CAD model is therefore considered as suitable for further calculations of physical parameters.

The knowledge base contains descriptions for the moment of inertia and its principal axis coordinate system. This information might be used for automatic calculation of motion equations or robot parameters. Beside the three values a cartesian coordinate system with translation description to its reference system (Module Base CS) has been included. The origin point is the module’s center of gravity. The relationship is demonstrated in Fig. 4.20.

Four points are necessary to define a cuboid. The remaining points can then be
4 Development of an EAS Module

Figure 4.20: Principal Axis Coordinate System

calculated and therefore the mathematical set that describes the cuboid. The point coordinates could also be retrieved from the CAD model. Fig. 4.21 shows the result.

The concepts of "Monitoring and Maintenance" are runtime information and therefore have to be updated by the either the module software itself, or, for maintenance or repair actions, a graphical user interface is supposed to be provided. Therefore this part of the template has been left blank.

4.6.2 Skills

A complete instantiation includes the Grab-Skill instantiation. A comprehensive description of instances can be found in C.1.

The tool center point denotes a point at an appropriate position, that marks the tool grasp position for e.g. a robot. The Grab-Skill includes also a coordinate system in the TCP. The coordinates of the Tool Center Point should be measured on the real module, preferably using standard methods like described in [62].

The TCP coordinate system can be used to describe other geometrical structures. Its
Figure 4.21: Module Work Envelope

Figure 4.22: Tool Center Point Coordinate System
position and orientation can simplify the description of e.g. the free space between the gripper jaws. The TCP coordinate system implementation is shown in Fig. 4.22, the free space between the gripper jaws is presented in Fig. 4.23.

Figure 4.23: Free Space Between Gripper Jaws

The force-displacement characteristics have been included. The characteristics given in the gripper datasheet have been approximated by a linear function. The function depends in this case on pneumatic system pressure and displacement. It programmed in JAVA and inserted into the skill form inside the ontology.

Although not part of the module, the interface skill has been instantiated. The interface skill for the moment includes a sole slot in order to enable interface identification.

4.7 Discussion

The starting point of the work described in this chapter was the availability of the Intermodular Receptacle description. With the available legacy equipment an attempt to create an EAS module featuring the Intermodular Receptacle was started.
After choosing a pneumatic gripper as base for the development, the requirement of a computer solution for the particular module was identified. Therefore a new integrated computer module, based on the gumstix verdex miniature computer was developed.

Using the available equipment and the new computer module, the new gripper module was designed using state-of-the-art CAD software. Afterwards the gripper module was built, using a high amount of legacy components.

The properties of the gripper module Hardware Interface were compared to the requirements and were determined to be appropriate. The Information Interface knowledge base was successfully instantiated for the case of the gripper module. The process was thoroughly documented.

The development of the Social Interface realisation is still in progress and is carried out by Uninova Lisbon. The approach, the software components and the current status of the development were described.

4.7.1 Equipment

The new proposed computer unit combines state-of-the-art processing power (in its size class) with high packing density. Compact Flash cards may provide way more than sufficient memory than necessary. The included network adapter enables reliable and fast communication. The included power supply works for a high range of voltages.

As many as possible legacy components were demanded for the module hardware realisation. Beside the computer module just the spacers had to be produced with an effort of half an hour and that out of a standard aluminium profile. The rest of the parts is legacy equipment. No legacy part had to be customized or adjusted. The presented solution is simple and cheap, but nonetheless has all necessary functionality.

The utilisation of legacy components is not always applicable and they might not always be available. In this case a module may be developed rather than adopting a legacy unit. If larger volumes of the computer module are demanded, an integrated circuit of the electronic (computer + power supply) components may be developed. Most of the applied legacy components are (still) purchasable. The application of bought compo-
A new in-house module storage solution was presented. It is applicable for the used tool exchange system. Due to the use of standard parts and an automated production process it is in this case cheaper than the Schunk solution. The results of the testing show a high reliability that exceeds by far the assumable robot positioning tolerances.

4.7.2 EAS Module

The physical parts fulfil the prior stated requirements of the hardware interface. In order to comply, the remaining description of the electrical and pneumatical connectors have been provided. The hardware interface is considered as conform with the IMR and therefore the EAS demands.

With the integration of the computer a hardware foundation for the social interface in the knowledge base has been established. Furthermore a functional software framework for a multi-agent control solution has been installed. The next steps require development of the agents itself and interfaces between soft- and hardware.

The module knowledge base has been instantiated successfully and described thoroughly. The next step is to create a JAVA representation out of it. Then, with suitable algorithms, the content and the structure can be tested.

The conversion of legacy components also has its drawbacks. The main function of the gripper, the opening and closing of the jaws, requires a pneumatic switch. It could be integrated onto the gripper module and an interface between the computer module and the switch could be established (AMI). But then a switch would be required on every pneumatic tool module. The module equipment and development costs would increase. Furthermore the dynamic properties would get worse with increased weight and this is not to aspired at. An external switch on the other hand reduces the autonomy of the module. If that external switch would be controlled by a central PLC (Programmable Logic Controller) that controls also other switches, the final system will be centralised, although the use of multi-agent control. A good solution would be the development of an intelligent switch module, that switches only the air-supply of the robot tool. If it fails, only a subsystem breaks down, while with the breakdown of a central PLC, the
whole system denies work.

Nonetheless, the in this chapter introduced gripper module is on the best way to become EAS compliant. The hardware interface is compliant, the information interface exists, but needs to be ported. The information interface is strongly connected to the social interface. And that is the last piece of the puzzle, that is required to call the here presented piece of equipment an EAS module.
5 Conclusion and Future Prospects

5.1 Conclusion

A complex interface, tailor-made for the requirements of EAS has been defined, specified, described and instantiated. The relationship to other EAS concepts has been clarified supported by a new process-oriented ontology. Interfaces are fundamental for modular systems and particularly for the manifold demands of Evolvable Assembly Systems. The Intermodular Receptacle complies to EAS demands and sets a base for the development of EAS modules.

A legacy piece of equipment has been converted to be EAS compliant. Two of three Intermodular Receptacle components have been instantiated successfully (Hardware Interface, Information Interface), the third one (Social Interface) is based on a known paradigm and development is in process. Even if the module development is not finished yet, a first conclusion can be drawn: Legacy equipment can comply.

5.2 Future Prospects

When the EAS module is finished, it will first have to be tested. If successful, further modules can be developed on the IMR base in order to solve the test case. The resulting cell will then most likely be evolvable from many point of views, as its components are.

Within the Information Interface many informations that can be used to obtain parameters have been included. System parameters and control are closely linked. A major part of future developments will be necessary in this field in order to get reconfigurability and adaptability.
From the experience gathered by the Electrolux project a methodology for the creation of systems and a global knowledge base that supports system development may be further developed.

With a running system the concepts of emergence [14], self-organisation [20], self-diagnosis [28], self-healing [28], relationships to complexity theory and other research domains may be further investigated and proved.

A perspective not actually from in the assembly domain, but from robotics is shown by [63]. Based on intelligent, standard interfaced robot modules, evolutionary algorithms are used successfully to obtain an optimum robot structure for a given task. The adaptability and reconfigurability must be maximised as this technology is supposed to be used on a place, where system reengineering is tricky: The moon.
A Further Developments

A.1 Test Case

For the evaluation two assemblies will be taken as a test case. The goal is to realise a changeover time of less than 15 minutes. Both assemblies are subsystems of Electrolux products and are being assembled manually at the moment.

A.1.1 The Delta Handle

The Delta Handle assembly group consists of a plastic handle, a latch, a spring and a pin as can be seen in Fig. A.1 and Fig. A.2. The shapes of the latch and the spring make them hard to orientate and to handle automatically. The hole basis fit between the pin and the handle is very tight and in combination with production tolerances an automated solution using conventional automation technology is very complex.
A.1.2 The Valve Group

The valve group consists of the valve itself, two tubes with different diameters and two types of clamp rings. Two clamp rings are necessary for each tube (Fig. A.3, Fig. A.4). There are two major challenges. One is to grab and put the tubes automatically onto the valve. The elastic rubber material changes its shape with application of small forces in relation to usual pneumatic gripping forces. Furthermore, the tubes bend due to their elasticity. That makes the application of the clamp rings the second challenge.

A.2 Handle Fixture

The porthole handle assembly has been designed to be built manually. Therefore, the pieces have not been designed to be grabbed automatically and reference planes have also not been considered. Automation of the assembly process is therefore a complex task.

A handle fixture proposal is presented in Fig. A.5. The position of the handle is defined by three faces (colored in green) and a spring-based locking mechanism (colored in blue). The handle lies on a plate (colored in red). The orientation of the handle can be adjusted with washers under the plate. In Fig. A.6 the fixture with inserted handle is shown.

Due to the limited space an insertion of the latch into a designated pocket was not possible. Therefore, an alternative solution, based on a lever mechanism, was implemented.
When the latch is inserted, the lever is, if it is in upright position, pushed back against a mechanical stop. The latch is then being moved to its final position. The lever moves with the latch. When the lever is in upright position, it is being locked by a spring mechanism that pushes a ball against a hole on the lever sidearms. The final position of the latch can be adjusted by three set screws. When inserted, the latch is embedded between two sidearms.

Afterwards the spring is inserted. It has no really defined geometrical shape and that makes it hard to handle automatically, but this may not be discussed in detail here. The round head of the latch may be used for guidance into the final position.

The last step is insertion of the pin. The tight tolerances between the handle hole and
the pin make it necessary to round the tip or at least to phase one end a pin.

A prototype of the handle fixture has been produced in the KTH Production Engineering workshop. The base part has been produced with the use of GibbsCAM and the Mazak three-axis CNC machine. The other parts have been manufactured by the author. The results are illustrated in Fig. A.8 and Fig. A.9.

![Figure A.8: Handle Fixture Prototype](image1)  
**- Empty**  

![Figure A.9: Handle Fixture Prototype](image2)  
**- Handle Assembled**

### A.3 Valve Fixture

For automated assembly of the valve a fixture has been created. Fig. A.10 shows a CAD model of the fixture, while Fig. A.11 shows a model of the valve. The position definition is realised by two centering pins on the valve (red) and holes on fixture (red). The valve lies on the edge of two metal parts (blue). The corresponding plane on the fixture is colored in blue.

A prototype from aluminium has been built. The result is presented in Fig. A.12. The hole basis fits do not fix the valve properly. Tighter holes could not be drilled, as the necessary equipment was not available. Therefore a threaded hole has been drilled, in order to clamp the valve with a screw. The use of GibbsCAM and the Mazak three-axis CNC machine. The other parts have been manufactured by the author. The results are illustrated in Fig. A.12 and Fig. A.13.
A Further Developments

Figure A.10: Valve Fixture - Empty

Figure A.11: Valve CAD model

Figure A.12: Valve Fixture Prototype - Empty

Figure A.13: Valve Fixture Prototype - Valve Assembled
A.4 Discussion

Furthermore a fixture for both assemblies was designed and prototyped. Both of them were not yet tested in an automated environment. The handle fixture is very promising, due to its adjustability possible deviations of dimensions may be compensated.

The handle fixture is a complicated solution. The approach was chosen because of the lack of alternatives. A simpler solution could not be found. A customisation of the product into the direction of assembly system promises to result in a simpler and thus cheaper and better controllable solution.

The valve fixture is suitable for testing purposes, but the hole basis fit deviations make the use of the screw necessary and prohibit industrial use in this state.
B Specifications

B.1 Auxiliary Concepts

In order to describe the properties of a module some auxiliary tools are necessary. Those will be divided into mathematical and non-mathematical.

B.1.1 Mathematical Tools

Coordinate Systems

![Diagram of Coordinate System Description](image)

Figure B.1: Description of Coordinate System Description

The definition of physical properties requires the description of coordinate systems. As a first proposal cartesian, cylindrical and spherical coordinate systems were defined. Those allow to describe most of the technical issues in assembly systems. If necessary, the definition can always be expanded and further types of coordinate systems, like for example ellipsoid coordinate systems, can be included. Fig. B.1 describes the structure of Coordinate System Description. Further aspects of coordinate system description can in found in chapter 4.6.1.
**Cartesian Coordinate System:** For the description of a point in cartesian coordinates, a three-dimensional vector \( \vec{r}_{ca} = (x_{ca}, y_{ca}, z_{ca})^T \) is necessary. The implementation of such a vector inside the ontology is shown in Fig. B.2. When in future descriptions the term "cartesian coordinate system" is being used, it refers to a right-handed three-dimensional coordinate system Fig. B.2 in cartesian notation.

![Cartesian Coordinate System Diagram](image)

Figure B.2: Description of a point in cartesian coordinates

**Cylindrical Coordinate System:** To be able to define a point in cylindrical coordinates a three-dimensional vector \( \vec{r}_{cy} = (\rho_{cy}, \varphi_{cy}, z_{cy})^T \), according to ISO31-11 will be used. Fig. B.3 shows the meaning of the vectors’ components and the implementation into the ontology.

![Cylindrical Coordinate System Diagram](image)

Figure B.3: Description of a point in cylindrical coordinates
**B Specifications**

**Spherical Point:** The definition of a point in spherical coordinates requires two angles and a radius. Therefore a three-dimensional vector \( \vec{r}_{sc} = \left( \rho_{sc}, \varphi_{sc}, \theta_{sc} \right)^T \) is applicable. The meaning and the implementation are shown in Fig. B.4.

![Figure B.4: Description of a point in spherical coordinates](image)

**Volumes**

**Cuboid:** If several geometrical constraints are predefined a cuboid can be explained with four points. The coordinates of the points can be given in any type of coordinate system. The remaining points, edge vectors and the side planes can be calculated implicitly, but this may be not discussed in detail here. The ontology implementation and a graphic explanation of the definition can be seen in Fig. B.5.

![Figure B.5: Description of cuboid implementation](image)

**Cylinder:** The definition of a cylinder requires predefined geometrical constraints and three points. The type of coordinate of coordinate system can be any in the system.
given. A graphic explanation of the definition and the actual implementation can be
found in Fig. B.6.

![Figure B.6: Description of cylinder implementation](image)

**Sphere:** Beside the geometrical constraints, for the definition of a sphere a center point
and the radius are necessary. The coordinate system may be any described. See Fig.
B.7 for a graphic description and implementation details.

![Figure B.7: Description of sphere implementation](image)

Clues how to implement the volumes can be found in ISO 10303 [64] and ISO 13584
[65].

**Coordinate System Transformation** For the transformation of cartesian coordinate
systems translation, rotation and scale will be taken into account. It can be described
with a mathematical equation:

\[ \vec{x}' = \lambda c_t \left( R\vec{x} + \vec{b} \right), \vec{b} = \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} \]

\[ R = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & \sin(\psi) \\ 0 & -\sin(\psi) & \cos(\psi) \end{pmatrix} \begin{pmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \]

The intersection of the \(xy\) and the \(x'y'\) coordinate planes is called the line of nodes \((N)\).

- \(\varphi\) is the angle between the \(x\)-axis and the line of nodes \(N\).
- \(\psi\) is the angle between the \(z\)-axis and the \(z'\)-axis.
- \(\theta\) is the angle between the line of nodes and the \(x'\)-axis.

\(\vec{b}\) adds the translation, \(R\) is a description of the rotation in Euler angles, according to the \(zxz'\)-convention [66]. \(\lambda_{ct}\) is a scale factor.

To allow a transformation independent of the coordinate system types, the canonical coordinate transformations to cartesian and back have been implemented. The actual implementation is shown in Fig. B.8.

**Canonical Coordinate Transformation:**

The definition of multiple coordinate system types implies that a way to transform the coordinates from one coordinate system into another must be defined. Therefore canonical coordinate transformations must be defined in the system. Cartesian coordinates will be regarded as the reference type coordinates. For every coordinate system type the transformation description to cartesian coordinates and from cartesian coordinates to the respective coordinate system type has to be defined.

The transformation into cartesian coordinates can be described with corresponding vectors:

- **Cartesian Coordinates:**
  \[
  \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}
  \]
### B Specifications

#### Coordinate System Transformation

- **Cylindrical Coordinates:**

  \[
  \begin{pmatrix}
  x \\
  y \\
  z
  \end{pmatrix}
  =
  \begin{pmatrix}
  \rho \cos \varphi_{cy} \\
  \rho \sin \varphi_{cy} \\
  z_{cy}
  \end{pmatrix}
  \]

- **Spherical Coordinates:**

  \[
  \begin{pmatrix}
  x \\
  y \\
  z
  \end{pmatrix}
  =
  \begin{pmatrix}
  \rho \sin \theta_{sc} \cos \varphi_{sc} \\
  \rho \sin \theta_{sc} \sin \varphi_{sc} \\
  \rho \cos \theta_{sc}
  \end{pmatrix}
  \]

Transformations from cartesian coordinates into the respective coordinate systems are more complicated:

- **Cartesian Coordinates:**

  \[
  \begin{pmatrix}
  x_1 \\
  y_1 \\
  z_1
  \end{pmatrix}
  =
  \begin{pmatrix}
  x_2 \\
  y_2 \\
  z_2
  \end{pmatrix}
  \]

- **Cylindrical Coordinates\(^1\):**

  \[
  \begin{pmatrix}
  \rho_{cy} \\
  \varphi_{cy} \\
  z_{cy}
  \end{pmatrix}
  =
  \begin{pmatrix}
  \sqrt{x^2 + y^2} \\
  \arccos \frac{x}{r}, y \geq 0 \\
  2\pi - \arccos \frac{x}{r}, y < 0
  \end{pmatrix}
  \]

\(^1\)Some programming languages, i.e. JAVA, provide the atan2-function, which allows to avoid the case differentiation.
• Spherical Coordinates:

\[
\begin{pmatrix}
\rho_{sp} \\
\varphi_{sp} \\
\theta_{sp}
\end{pmatrix}
= \begin{pmatrix}
\sqrt{x^2 + y^2 + z^2} \\
\arctan \sqrt{\frac{x^2 + y^2}{z}} \\
\arctan \frac{y}{x}
\end{pmatrix}
\]

The actual implementation can be found in Fig. B.9. The mathematical functions have not been implemented yet.

Figure B.9: Description of the Canonical Coordinate Transformation

**Mathematical Function**  In order to describe mathematical coherences, an implementation including a string is part of the ontology (Fig. B.10). In this string a function can be included using JAVA code.

### B.1.2 Non-Mathematical Tools

**Maintenance Data**  Data about scheduled and unscheduled maintenance operations are saved in form of an instantiation of this class. So far, a text description of the maintenance operation, a maintenance technician ID, the necessary time to maintain the module, a time stamp and multiple strings describing the repaired or exchanged parts (Fig. B.11).

**Operation Data**  In order to keep track of the performed module operations the class *Operation Data* was included (Fig. B.12). Featuring a time stamp, a description of the
Figure B.10: Implementation of the Mathematical Function

Figure B.11: Implementation of the Maintenance Data
performed skill, by saving the skill instance, and the instance of the finite state machine the necessary information a complete logging of the module activities becomes possible.

Figure B.12: Implementation of the Operation Data

**Finite State Machine** The module interaction, especially the communication, makes it necessary to know what other modules are doing. Therefore a finite state machine will be developed, which describes the actual status of the module. So far, just a status string, representing the output, was implemented Fig. B.13.

**Time Information** To realise time stamps the class *Time Information* was implemented. It features information about date and time. An assembly system should provide a central time server or all modules should use the same one. The description in the ontology can be seen in Fig. B.14.

**Electrical Parameters** For the description of *Electrical Parameters* an abstract class was created. It features allowed voltage and current ranges, as well as the suggested electrical voltage. By instantiating one the concrete subclasses its possible to chose between DC, AC and three-phase current. The realisation inside the ontology is presented in Fig. B.15.
Figure B.13: Implementation of the Finite State Machine

Figure B.14: Implementation of Time Information

Figure B.15: Implementation of Electrical Parameters
**Hardware Interface**  Comparing two interface sides with software requires an appropriate identifier. The interface skill for the master side as well as the information of the slave side, stored in the Generic Properties, are equipped with an instantiation of this class. Despite an identification string it features the interface coordinate system, defined by geometrical properties on the slave side. On the master side the coordinate system is supposed to be referenced onto module coordinate system, directly or indirectly. The coordinate system transformation from master to slave side is supposed to be trivial. The implementation is shown in Fig. B.16.

![Figure B.16: Implementation of Hardware Interface](image)

**Module Typology**  In order to be able to get a rough estimation of the module abilities, it may be allocated to one of the concrete classes within the abstract Module Typology. The list, consisting of just three elements, may be expanded when necessary. The inclusion is demonstrated in Fig. B.17.

![Template Slots](image)
Figure B.17: Implementation of Module Typology
C Information Interface

Instantiation

Goal of this section is to lead future step by step through the IMR instantiation process, to show how the elements are meant to be used and how upcoming problems should be solved. The theoretical description already exists, but during the realisation one encounters often unforeseen problems. Hopefully as many as possible can be eliminated. This how-to will be more a structured text than a manual for Protégé. A GUI for the future implementation will most likely be created and therefore a Protégé manual would not have a very long persistence.

When referring to classes, methods or variable types like `String`, `Integer` or `Float`, JAVA structures with their properties are meant. A detailed description can be found in [67].

The basic physical units specified for the ontology are m, s, g, V, A, N, bar, °C. Angles are expressed in radian measure. If a slot description does not contain a unit requirement, the basic units shall be applied. This means if not mentioned explicitly to do it differently, the float value of "1µm" is supposed to be "0.000001".

C.0.3 General Properties

Device Identification

Here every information that concerns identification is stored.
### C Information Interface Instantiation

<table>
<thead>
<tr>
<th>ID</th>
<th>String</th>
<th>Gripper1</th>
</tr>
</thead>
</table>

An unique identification code representing the module in the system is required here. The ID may be randomly chosen, but must be unique in the system.

<table>
<thead>
<tr>
<th>Table C.1: Generic Properties - Device Identification - ID</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Instance of Module Typology</th>
<th>&quot;ToolInstance&quot;</th>
</tr>
</thead>
</table>

A small set of module types (so far: Robot, Feeder, Tool) is included. To choose a type, an instantiation of one of the types is required to be assigned to this slot. For each type one instance should be available in the system (e.g. "ToolInstance", "RobotInstance", "FeederInstance").

<table>
<thead>
<tr>
<th>Table C.2: Generic Properties - Device Identification - Module Type</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module Name</th>
<th>String</th>
<th>&quot;Schunk PFG-F 65 Gripper&quot;</th>
</tr>
</thead>
</table>

The module name is supposed to give a human interactor a distinct, not necessarily unique, indication of manufacturer, model and functionality.

<table>
<thead>
<tr>
<th>Table C.3: Generic Properties - Device Identification - Module Name</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Serial Number</th>
<th>String</th>
<th>&quot;n/a&quot;</th>
</tr>
</thead>
</table>

The manufacturer’s serial number for the module. A serial number is an unique number assigned for identification. One serial number marks one object. In this case a serial number could not be retrieved, therefore this field is marked with "n/a". The serial number is supposed to allow manufacturers in case of maintenance or repair to recognize a module.

| Table C.4: Generic Properties - Device Identification - Serial Number |
The part number the manufacturer gave the part. A part number is a unique identifier of a part used in a particular industry. One part number might be used for multiple objects. If not available, the placeholder value "n/a" should be inserted. The particular value was taken from a sticker on the gripper. The serial number is supposed to allow manufacturers in case of maintenance or repair to recognize the product category.

Table C.5: Generic Properties - Device Identification - Part Number

<table>
<thead>
<tr>
<th>Part Number</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;300250&quot;</td>
</tr>
</tbody>
</table>
### Mechanical Interface

#### CAD Information

<table>
<thead>
<tr>
<th>String</th>
<th>&quot;/data/CAD/module.step&quot;</th>
</tr>
</thead>
</table>

A path to a ISO 10303-203:2005 step 3D model file. The file should be stored on the module file system and not externally. Access compliant to the filesystem hierarchy standard [68] is supposed to be enabled. It is not compulsory to fill out this slot, but the integration of 3D models promises manifold prospects e.g. in respect to automated or computerassisted system design, programming or configuration.

Table C.6: Generic Properties - Mechanical Interface - CAD Information

#### Hardware Interface Slave

<table>
<thead>
<tr>
<th>Instance of Hardware Interface</th>
<th>&quot;Schunk SWA-010-E10&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW Interface Information (String)</td>
<td>&quot;Schunk SWS-010-E10&quot;</td>
</tr>
<tr>
<td>Interface Coordinate System (Instance of Coordinate System Description)</td>
<td>&quot;Schunk SWS-010-E10 Coordinate System&quot;</td>
</tr>
</tbody>
</table>

*HW Interface Information* is a distinct identifier that is supposed have the same value in the respective instantiation in the Interface-Skill. The *Interface Coordinate System* is represents a link between hardware attributes and mathematical description. An instantiation from the respective Interface-Skill may not be reused here, because of the different coordinate systems.

Table C.7: Generic Properties - Mechanical Interface - Hardware Interface Slave
This instance describes the coordinate system of the mechanical interface. Without connection, the coordinate system is linked to geometrical attributes of the module (Fig. C.1). The Interface-skill provides the Coordinate System Description of the respective interface master and the Base Coordinate System Transformation that is inserted here on connection and deleted on disconnection. By definition of the Origin Point the type of the coordinate system is chosen. The instantiation of Coordinate System Description requires always a new instantiation of Origin Point. More info regarding this topic can be found in 4.6.1.

Table C.8: Generic Properties - Mechanical Interface - Module Base CS

<table>
<thead>
<tr>
<th>Instance of Cuboid (Volume)</th>
<th>&quot;Module Envelope&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_{Cu1}) (Point)</td>
<td>&quot;EnvelopeP1&quot;</td>
</tr>
<tr>
<td>(P_{Cu2}) (Point)</td>
<td>&quot;EnvelopeP2&quot;</td>
</tr>
<tr>
<td>(P_{Cu3}) (Point)</td>
<td>&quot;EnvelopeP3&quot;</td>
</tr>
<tr>
<td>(P_{Cu4}) (Point)</td>
<td>&quot;EnvelopeP4&quot;</td>
</tr>
</tbody>
</table>

The module envelope is a virtual hull that allows a rough space estimation. A cuboid promised the best estimation. The result is presented in Fig. C.2. To define a cuboid at least four points are necessary (with implicit constraints). They are all defined in the Module Base CS. As example EnvelopeP1 is presented in detail. All Points should have the same reference system, although they not necessarily have to. The point values have been aquired from the CAD model. In cases of doubt, the envelope should rather be a little bit bigger than necessary, therefore more space than used in the realisation was accounted for the cables. The other values are exact plane-to-plane measurements.

Table C.9: Generic Properties - Mechanical Interface - Base CS Origin Point
Environmental Interface

<table>
<thead>
<tr>
<th>Humidity</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>n/a</td>
</tr>
<tr>
<td>Maximum Humidity (String)</td>
<td>&quot;80&quot;</td>
</tr>
<tr>
<td>Minimum Humidity (String)</td>
<td>&quot;0&quot;</td>
</tr>
</tbody>
</table>

After checking datasheet information of the major components, the most sensitive part turned out to be the gumstix. Therefore values from the gumstix datasheet have been inserted. They refer to the relative humidity in percent.

Table C.10: Generic Properties - Environmental Interface - Humidity

<table>
<thead>
<tr>
<th>Temperature</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>n/a</td>
</tr>
<tr>
<td>Maximum Ambient Air Temperature (String)</td>
<td>&quot;5&quot;</td>
</tr>
<tr>
<td>Minimum Ambient Air Temperature (String)</td>
<td>&quot;40&quot;</td>
</tr>
</tbody>
</table>

Not for all components a temperature range could be found in the datasheets. Therefore reasonable values have been inserted. The most sensitive component to high temperatures should be module, especially the power supply. Although it did not heat up during testing, temperatures above 40°C should be avoided. Furthermore condensed water may cause short circuits, therefore temperatures below 5°C are not allowed.

Table C.11: Generic Properties - Environmental Interface - Temperature
Electrical Interface

Table C.12: Generic Properties - Electrical Interface - Electrical Interface Parameters

### Electrical Interface Parameters

| Instance of Electrical Interface Parameters | "Module Electrical Parameters"
---|---

Three types of electrical currents have been considered in the information interface. To choose the type, an instance of the abstract class *Electrical Interface Parameters* is necessary. By the choice of one of the concrete succeeding classes, the power supply knows if DC, AC or TPC current is used to power the module. Inside those classes the parameters are declared in detail.

Pneumatic Interface

Table C.13: Generic Properties - Pneumatic Interface

<table>
<thead>
<tr>
<th>Pneumatic Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
</tr>
</tbody>
</table>

- *Maximum Pneumatic Pressure* (Float) | "8"
- *Minimum Pneumatic Pressure* (Float) | "3.5"
- *Suggested Pneumatic Pressure* (Float) | "6"

This is gripper datasheet information. The given pressures are in bar.
## Hydraulic Interface

<table>
<thead>
<tr>
<th>Hydraulic Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
</tr>
</tbody>
</table>

- **Maximum Hydraulic Pressure (Float)**: "" empty
- **Minimum Hydraulic Pressure (Float)**: "" empty
- **Suggested Hydraulic Pressure (Float)**: "" empty
- **Fluid Description (Float)**: "" empty

This is datasheet information. The given pressures are in bar. In this case there is no hydraulic interface. The *Fluid Description* is supposed to contain info about the allowed and not allowed fluids. It may also be a link to a datasheet.

Table C.14: Generic Properties - Hydraulic Interface

### Physical Properties

<table>
<thead>
<tr>
<th>Center of Gravity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
</tr>
</tbody>
</table>

- **Center of Gravity Point (Instance of Cartesian Point (Point))**: "Module Center of Gravity"

Obtaining the center of gravity requires some calculation or measurement effort. In this case the availability of a CAD model helps. Autodesk Inventor 2008 allows to define physical properties of components. The program is able to calculate physical properties, amongst them the center of gravity.

Table C.15: Generic Properties - Physical Properties - Center of Gravity Point
This value has been obtained by weighing the module. The value is given in gram. The respective value in the CAD model is 1103g. The small discrepancy indicates a high eligibility for further dynamic calculations.

Table C.16: Generic Properties - Physical Properties - Module Mass

<table>
<thead>
<tr>
<th>Class</th>
<th>2.214</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moment of Inertia - X</td>
<td></td>
</tr>
<tr>
<td>Moment of Inertia - Y</td>
<td>0.862</td>
</tr>
<tr>
<td>Moment of Inertia - Z</td>
<td>2.114</td>
</tr>
</tbody>
</table>

The values of the moment of inertia are quite complicated to obtain. Either collaborate measurement or time-consuming calculations are required. In this case the CAD software helped with the calculations. Due to the description in principal coordinates only three values instead of 9 are necessary to describe the moment of inertia.

Table C.17: Generic Properties - Physical Properties - Moment of Inertia
C.0.4 Auxiliary Concepts

<table>
<thead>
<tr>
<th>Interface Coordinate System</th>
<th>&quot;Schunk SWA-010-E10 Coordinate System&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate System Description (String)</td>
<td>&quot;Schunk SWA-010-E10 Coordinate System&quot;</td>
</tr>
<tr>
<td>Base Coordinate System Transformation (Instance of Coordinate System Transformation)</td>
<td>empty</td>
</tr>
<tr>
<td>Origin Point (Instance of Point)</td>
<td>&quot;SWA-010-E10 CS Origin Point&quot;</td>
</tr>
</tbody>
</table>

This instance describes the coordinate system of the mechanical interface. Without connection, the coordinate system is linked to geometrical attributes of the module (See 4.6.1 for details). The Interface-skill provides the Coordinate System Description of the respective interface master and the Base Coordinate System Transformation that is inserted here on connection and deleted on disconnection. By definition of the Origin Point the type of the coordinate system is chosen. The instantiation of Coordinate System Description requires always a new instantiation of Origin Point.

Table C.18: Auxiliary Concepts - Interface Coordinate System
C Information Interface Instantiation

Figure C.1: Interface Coordinate System

<table>
<thead>
<tr>
<th>Instance of Cartesian Point (Point)</th>
<th>&quot;SWA-010-E10 CS Origin Point&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Y_Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Z_Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Reference System (Instance of Coordinate System Description)</td>
<td>&quot;Schunk SWA-010-E10 Coordinate System&quot;</td>
</tr>
</tbody>
</table>

*Point* is an abstract class. Its three subclasses allow to chose between three different types of points. But each type of point just works in the respective coordinate system. E.g. defining a point in cylindrical coordinates in a cartesian coordinate system wouldn’t make much sense. This circumstance is used in the *Point* class in order to chose the type of the used coordinate system. The coordinates X_Ca, Y_Ca, Z_Ca in *Origin Point* must be "0". In this case the "Schunk SWA-010-E10 Coordinate System" has been defined as a cartesian coordinate system. The *Reference System* assigns the point to the "Schunk SWA-010-E10 Coordinate System".

Table C.19: Auxiliary Concepts - Origin Point
### Base Coordinate System Transformation

<table>
<thead>
<tr>
<th>Instance of Coordinate System Transformation</th>
<th>&quot;InterfaceCS_to_BaseCS&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Y_Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Z_Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Phi_Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Psi_Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Theta_Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Transformation Description (Instance of Canonical Coordinate Transformation)</td>
<td>&quot;Cartesian&quot;</td>
</tr>
<tr>
<td>Reference System (Instance of Coordinate System Description)</td>
<td>&quot;Base Coordinate System Transformation&quot;</td>
</tr>
</tbody>
</table>

In this case no transformation between the coordinate systems is necessary. The interface coordinate system works just fine as module base coordinate system. The Transformation Description is compulsory to choose. The target system is cartesian, therefore the Cartesian to Cartesian translation has been chosen.

Table C.20: Auxiliary Concepts - Base Coordinate System Transformation
### Origin Point

<table>
<thead>
<tr>
<th>Instance of Cartesian Point (Point)</th>
<th>&quot;Base CS Origin Point&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Y Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Z Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Reference System (Instance of Coordinate System Description)</td>
<td>&quot;Module Base CS&quot;</td>
</tr>
</tbody>
</table>

By this instantiation *Module Base CS* is defined as cartesian coordinate system. See C.0.4 for details.

<table>
<thead>
<tr>
<th>P Cu1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance of Cartesian Point (Point)</td>
<td>&quot;EnvelopeP1&quot;</td>
</tr>
<tr>
<td>X Ca (Float)</td>
<td>&quot;75.5&quot;</td>
</tr>
<tr>
<td>Y Ca (Float)</td>
<td>&quot;-70&quot;</td>
</tr>
<tr>
<td>Z Ca (Float)</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Reference System (Instance of Coordinate System Description)</td>
<td>&quot;Module Base CS&quot;</td>
</tr>
</tbody>
</table>

As example of how to define a *Point*, *EnvelopeP1* was chosen. As *Reference System*, the Module Base CS was chosen. The values derive from the envelope description.

Table C.21: Auxiliary Concepts - Base CS Origin Point

Table C.22: Auxiliary Concepts - EnvelopeP1 (Point)
Figure C.2: Module Envelope
C Information Interface Instantiation

### Electrical Interface Parameters

| Instance of DC Electrical Power Supply (Electrical Parameters) | "Module Electrical Parameters"
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Electrical Voltage (Float)</td>
<td>&quot;8&quot;</td>
</tr>
<tr>
<td>Maximum Electrical Voltage (Float)</td>
<td>&quot;30&quot;</td>
</tr>
<tr>
<td>Minimum Electrical Current (Float)</td>
<td>&quot;0.8&quot;</td>
</tr>
<tr>
<td>Maximum Electrical Current (Float)</td>
<td>&quot;2&quot;</td>
</tr>
<tr>
<td>Suggested Electrical Voltage (Float)</td>
<td>&quot;24&quot;</td>
</tr>
</tbody>
</table>

Here the parameters of the module power supply have been inserted (See 4.3.2 for details.) Voltages are defined in Volt, the current unit is Ampère. The system knows which type of current is used (DC, AC or three phase), by the instantiated class.

Table C.23: Auxiliary Concepts - Electrical Interface Parameters

### Module Center of Gravity

| Instance of Cartesian Point (Point) | "EnvelopeP1"
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X_Ca (Float)</td>
<td>&quot;-0.052&quot;</td>
</tr>
<tr>
<td>Y_Ca (Float)</td>
<td>&quot;-1.979&quot;</td>
</tr>
<tr>
<td>Z_Ca (Float)</td>
<td>&quot;68.293&quot;</td>
</tr>
</tbody>
</table>

Reference System (Instance of Coordinate System Description) | "Module Base CS"

The center of gravity has been obtained from the CAD model (See C.0.3 for details).

Table C.24: Auxiliary Concepts - Module Center of Gravity (Point)
Principal Axis Coordinate System

<table>
<thead>
<tr>
<th>Instance of Coordinate System Description</th>
<th>&quot;Principal Axis CS&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS Description (String)</td>
<td>&quot;Principal Axis CS referred to Module Base CS&quot;</td>
</tr>
<tr>
<td>Base Coordinate System Transformation</td>
<td>&quot;BaseCS_to_PrincipalCS&quot;</td>
</tr>
<tr>
<td>Origin Point (Instance of Cartesian Point (Point))</td>
<td>&quot;Principal Axis Origin Point&quot;</td>
</tr>
</tbody>
</table>

Basically there is nothing new here. A new instance for the Origin Point has been created. The Principal Axis Coordinate System has to be cartesian. More info in Tab. C.0.3.

Table C.25: Auxiliary Concepts - Principal Axis Coordinate System (Coordinate System Description)

<table>
<thead>
<tr>
<th>Base Coordinate System Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance of Coordinate System Transformation</td>
</tr>
<tr>
<td>B Ct - X (Float)</td>
</tr>
<tr>
<td>B Ct - Y (Float)</td>
</tr>
<tr>
<td>B Ct - Z (Float)</td>
</tr>
<tr>
<td>Phi_Ct (Float)</td>
</tr>
<tr>
<td>Psi_Ct (Float)</td>
</tr>
<tr>
<td>Theta_Ct (Float)</td>
</tr>
<tr>
<td>Transformation Description (Instance of Canonical Coordinate Transformation)</td>
</tr>
<tr>
<td>Reference System (Instance of Coordinate System Description)</td>
</tr>
</tbody>
</table>

The translation vector $\vec{b}_{ct}$ for the principal axis is always the center of gravity. The rotation values have once more been obtained from Inventor. The transformation description is cartesian as the two involved systems are cartesian. The rotation is tricky to describe: First a rotation around the $z$-axis is performed ($\varphi$), then around the new, temporary $x$-Axis, called "Line of Nodes" ($\psi$) and then finally around the new $z'$-axis ($\theta$).

Table C.26: Auxiliary Concepts - Base Coordinate System Transformation
C Information Interface Instantiation

C.1 Skills

C.1.1 Grab

<table>
<thead>
<tr>
<th>Grab</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload (Float)</td>
<td>&quot;1800&quot;</td>
</tr>
<tr>
<td>Response Time Closing (Float)</td>
<td>&quot;0.06&quot;</td>
</tr>
<tr>
<td>Response Time Opening (Float)</td>
<td>&quot;0.06&quot;</td>
</tr>
<tr>
<td>Repeating Accuracy (Float)</td>
<td>&quot;0.00002&quot;</td>
</tr>
<tr>
<td>Maximum Displacement (Float)</td>
<td>&quot;0.0115&quot;</td>
</tr>
<tr>
<td>Jaw CAD Information (String)</td>
<td>&quot;/CAD/JAWS/&quot;</td>
</tr>
<tr>
<td>Force-Displacement Characteristics (Instance of Mathematical Function)</td>
<td>&quot;forcedispl-function1&quot;</td>
</tr>
<tr>
<td>Free Space Volume (Instance of Cuboid (Volume))</td>
<td>&quot;FreeSpaceGrab1&quot;</td>
</tr>
<tr>
<td>Gripper Status (Instance of Finite State Machine)</td>
<td>empty</td>
</tr>
<tr>
<td>Tool Center Point (Instance of Cartesian Point (Point))</td>
<td>&quot;Grab1TCP&quot;</td>
</tr>
<tr>
<td>Tool Center Point Coordinate System (Instance of Coordinate System Description)</td>
<td>&quot;Grab1TCPSCS&quot;</td>
</tr>
<tr>
<td>Grab Argument (Instance of Argument)</td>
<td>&quot;closeGripper&quot;</td>
</tr>
<tr>
<td>Displacement (Instance of Point)</td>
<td>empty</td>
</tr>
</tbody>
</table>

Payload, Response Time Closing, Response Time Opening, Repeating Accuracy and Maximum Displacement are basic datasheet informations. Jaw CAD Information is a link to a directory on the module file system, see Tab. C.0.3 for details. The Gripper Status finite state machine has not been implemented into the ontology yet. The other instantiations may described in detail in separate tables. As described before, the Displacement is only applicable, when sensors are installed, therefore the slot is left empty. The Grab Argument denotes if the gripper shall open or close.

Table C.27: Skills - Grab - Grab
Just as an example, the code of a JAVA method has been implemented here. It returns a float value that represents the gripper force in N and needs system pressure and displacement as input values in order to calculate the force. The description of the mathematical function will be revised later. The linear function has been calculated from datasheet values.

Table C.28: Skills - Grab - Force-Displacement Characteristics

<table>
<thead>
<tr>
<th>Force-Displacement Characteristics</th>
<th>Instance of Mathematical Function</th>
<th>&quot;forcedispl-function1&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function (String)</td>
<td>&quot;public float force(float pressure, float displacement)float frc = -0.3 * displacement + 55*pressure;return frc;&quot;</td>
<td></td>
</tr>
</tbody>
</table>

The Tool Center Point (TCP) is a virtual reference point, that describes in an appropriate way the position of the tool. This is being used especially for the programming of robots. In this case a centered point between the jaws is a logic choice.

Table C.29: Skills - Grab - Tool Center Point

<table>
<thead>
<tr>
<th>Tool Center Point</th>
<th>Instance of Cartesian Point (Point)</th>
<th>&quot;Grab1TCP&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_Ca (Float)</td>
<td>&quot;0&quot;</td>
<td></td>
</tr>
<tr>
<td>Y_Ca (Float)</td>
<td>&quot;0&quot;</td>
<td></td>
</tr>
<tr>
<td>Z_Ca (Float)</td>
<td>&quot;147.65&quot;</td>
<td></td>
</tr>
<tr>
<td>Reference System (Instance of Coordinate System Description)</td>
<td>&quot;Module Base CS&quot;</td>
<td></td>
</tr>
</tbody>
</table>
**Tool Center Point Coordinate System**

| Instance of Coordinate System Description | "Grab1TCP CS"
| CS Description (String) | "Cartesian CS with TCP as origin"
| Base Coordinate System Transformation (Instance of Coordinate System Transformation) | "BaseCS_to_TCP CS"
| Origin Point (Instance of Cartesian Point (Point)) | "TCP CS Origin Point"

Definition of a Coordinate System in the TCP. No rotation. New instance of Point necessary.

Table C.30: Skills - Grab - TCP Coordinate System (Coordinate System Description)

**Base Coordinate System Transformation**

| Instance of Coordinate System Transformation | "BaseCS_to_TCP CS"
| B Ct - X (Float) | 0
| B Ct - Y (Float) | 0
| B Ct - Z (Float) | 147.65
| Phi_Ct (Float) | 0
| Psi_Ct (Float) | 0
| Theta_Ct (Float) | 0
| Transformation Description (Instance of Canonical Coordinate Transformation) | "Cartesian"
| Reference System (Instance of Coordinate System Description) | "Module Base CS"

When all rotation angles are zero, just the coordinates of the TCP need to be filled in here. Module Base CS is the reference system again.

Table C.31: Skills - Grab - TCP Coordinate System Transformation

108
This is a space that is supposed to be substracted from module envelope, in order to allow a more sophisticated collision detection. In this case a cuboid was supposed to be suitable to model the space between the gripper jaws. The width of the cuboid equals the module envelope width. As reference system the TCP coordinate system was chosen. The point instantiations will not be discussed in detail. The values are: \( P_1(75.5, 11.2, 12), P_2(75.5, -11.2, 12), P_3(75.5, -11.2, -12), P_4(-75.5, -11.2, -12) \).

Table C.32: Skills - Grab - Free Space Volume

<table>
<thead>
<tr>
<th>Free Space Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance of Cuboid (Volume)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>( P_{Cu1} ) (Point)</td>
</tr>
<tr>
<td>( P_{Cu2} ) (Point)</td>
</tr>
<tr>
<td>( P_{Cu3} ) (Point)</td>
</tr>
<tr>
<td>( P_{Cu4} ) (Point)</td>
</tr>
</tbody>
</table>
C.1.2 Interface

<table>
<thead>
<tr>
<th>MasterCS to SlaveCS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transformation</strong></td>
</tr>
</tbody>
</table>

| **B Ct - X (Float)** | 0 |
| **B Ct - Y (Float)** | 0 |
| **B Ct - Z (Float)** | 0 |
| **Phi_Ct (Float)**   | 0 |
| **Psi_Ct (Float)**   | 0 |
| **Theta_Ct (Float)** | 0 |

| **Transformation Description** (Instance of Canonical Coordinate Transformation) | "Cartesian" |
| **Reference System** (Instance of Coordinate System Description) | empty |

The coordinate system transformation from interface master to interface slave is trivial, when the coordinate systems are chosen appropriately. On the master side, the Interface CS should be referenced to another coordinate system to establish a link to global cell CS (if possible).

Table C.33: Skills - Interface - MasterCS to SlaveCS

<table>
<thead>
<tr>
<th>Hardware Interface Master</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance of Hardware Interface</strong></td>
</tr>
</tbody>
</table>

The counterpart to HW Interface Information and serves as a distinct identifier.

Table C.34: Skills - Interface - Hardware Interface Slave
Bibliography


