An Ontology-Based Expert System for the Systematic Design of Humanoid Robots

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Samuel Rader

Tag der mündlichen Prüfung:	15.06.2020
1. Referent:	Prof. Dr.-Ing. Tamim Asfour
2. Referent:	Prof. Dr. Karsten Berns

Deutsche Zusammenfassung

Die Entwicklung humanoider Roboter stellt eine komplexe, herausfordernde und zeitaufwendige Aufgabe dar, die interdisziplinäres Expertenwissen erfordert. Roboterentwickler erlangen einen Großteil dieses Wissens durch Erfahrung. Eine große Herausforderung ist es jedoch, dieses Wissen an Nachfolger weiterzugeben. Obwohl Methoden des Wissensmanagements und der Produktentwicklung in vielen Bereichen wie der Automobilindustrie immer mehr verwendet werden, gibt es in der Robotik noch keinen systematischen Ansatz, der den Entwurf von Robotern durch das Zurückgreifen auf vorhandenes Expertenwissen vereinfacht.

Ziel dieser Arbeit ist es daher, Expertenwissen zur mechatronischen Entwicklung von Roboterkomponenten zu konservieren, um es zum Entwurf zukünftiger humanoider Roboter wiederzuverwenden. Dazu wird ein systematischer Ansatz vorgestellt, der Expertenwissen durch Methoden der künstlichen Intelligenz nutzbar macht. Der Ansatz lässt sich in drei aufeinanderfolgende Schritte unterteilen. Im ersten Schritt wird Wissen durch die Konstruktion und Analyse von Komponenten für humanoide Roboter gewonnen. Dieses Wissen wird im zweiten Schritt formalisiert und in Form einer ontologischen Wissensbasis gespeichert. Im letzten Schritt wird diese Wissensbasis zur Erstellung eines Expertensystems genutzt, das den Entwicklungsprozess zukünftiger humanoider Roboter unterstützt indem es Lösungsvorschläge für den Entwurf liefert.

Die ersten beiden Kernbeiträge dieser Arbeit, die *Formalisierung von Expertenwissen* und die *Entwicklung eines Expertensystems zur Unterstützung beim Entwurf humanoider Roboter* ergeben sich aus dem beschriebenen Ansatz. Der dritte Kernbeitrag beschreibt die *Evaluierung des Ansatzes am Beispiel von ARMAR-6*, einem kollaborativen humanoiden Roboter. Diese Kernbeiträge werden im Folgenden genauer beschrieben.

Formalisierung von Expertenwissen zum Entwurf humanoider Roboter

Der erste Beitrag dieser Arbeit ist ein neuer Formalisierungsprozess. Ziel dieses Prozesses ist es, Entwurfswissen so zu formalisieren, dass es als Wissensba-

sis eines Expertensystems dienen kann. Roboterentwickler gewinnen Wissen, indem sie Roboterkomponenten entwickeln und diese sowie verwandte Arbeiten analysieren. Basierend auf dieser Analyse wird eine Roboterkomponente in Form eines gerichteten azyklischen Graphen modelliert. Der Graph beschreibt den hierarchischen Aufbau der Roboterkomponente als Kombination von Teilsystemen, die wiederum selbst aus Teilsystemen bestehen. Eine Roboterkomponente wird auf diese Weise bis hin zu Platzhaltern für Katalogkomponenten hierarchisch modelliert. Verschiedene Lösungsprinzipien und Optionen werden durch zusätzliche Knoten dargestellt. Anschließend wird aus dem Graphen eine Ontologie abgeleitet, die den Aufbau der Roboterkomponente beschreibt. Wissen zu existierenden Katalogkomponenten wird durch eine weitere Ontologie repräsentiert, die neben physikalischen Daten des Herstellers auch hierarchische Verbindungen sowie Kompatibilitäten zwischen den Komponenten berücksichtigt. Ein parametrisierter Regelsatz beschreibt die Berechnung der Eigenschaften von Teillösungen in Abhängigkeit von den konkreten Katalogkomponentendaten. Außerdem definiert er, wann Anforderungen erfüllt sind.

Expertensystem zur Unterstützung beim Entwurf humanoider Roboter

Der zweite Beitrag dieser Arbeit ist ein Expertensystem, das den Entwurf humanoider Roboterkomponenten unterstützt. Die aus dem Formalisierungsprozess resultierenden Ontologien und Regeln bilden zusammen die Wissensbasis des Expertensystems, dessen weitere Komponenten eine Inferenzmaschine und eine Benutzerschnittstelle sind. Der Nutzer startet durch die Eingabe von Anforderungen einen mehrstufigen Schlussfolgerungsprozess, durch den auf Grundlage der Wissensbasis Entwurfslösungen in einem Bottom-Up-Ansatz generiert werden. Entsprechend des Graphen der Roboterkomponente werden zunächst Teillösungen gebildet, indem Katalogkomponenten und Lösungsprinzipien kombiniert werden. Danach werden die Eigenschaften der Teillösungen berechnet und mit den Anforderungen verglichen. Teillösungen, welche die Anforderungen nicht erfüllen, werden verworfen. Alle verbliebenen Teillösungen werden mit anderen Teillösungen kombiniert bis vollständige, anforderungsgerechte Entwurfslösungen entstehen. Die resultierenden Entwurfslösungen umfassen alle nötigen Katalogkomponenten, die gewählten Lösungsprinzipien und die berechneten Eigenschaften der Kombination. Sie werden dem Nutzer über die Benutzerschnittstelle visualisiert und können sowohl nach einzelnen Leistungsanforderungen als auch nach gewichteten Optimierungsfunktionen angeordnet werden.

Evaluierung am Beispiel von Komponenten für ARMAR-6

Der dritte Beitrag ist die Evaluierung des Ansatzes am Beispiel von Komponenten für ARMAR-6, einem kollaborativen humanoiden Roboter für industrielle Umgebungen. Dies beinhaltet die Entwicklung von mechatronischen Komponenten für ARMAR-6, insbesondere hochintegrierten Sensor-Aktor-Controller-Einheiten (SAC-Einheiten) für Robotergelenke. Die SAC-Einheiten sind neben den Roboterhänden die ARMAR-6-Komponenten mit dem höchsten Integrationsgrad, was ihre Entwicklung besonders herausfordernd macht. Daher dienen sowohl die SAC-Einheiten als auch die Roboterhände als Fallstudien, um den in dieser Arbeit vorgestellten Ansatz zu evaluieren. In beiden Fallstudien wird das Wissen zum Entwurf der Roboterkomponente formalisiert und in ein Expertensysteme integriert. Es wird demonstriert, dass das Expertensystem existierende Roboterkomponenten reproduzieren, optimieren und skalieren kann. Außerdem wird gezeigt, dass das Expertensystem Lösungen für Neuentwicklungen generieren kann, indem es Lösungsprinzipien und Katalogkomponenten neu kombiniert.

Acknowledgment

This thesis is the result of my work as a research scientist at the High Performance Humanoid Technologies Lab (H^{2}T) of the Institute for Anthropomatics and Robotics (IAR), Karlsruhe Institute of Technology (KIT).

First of all, I would like to thank my thesis supervisor and the head of H^{2}T Prof. Dr.-Ing. Tamim Asfour. In numerous conversations he not only gave me valuable feedback on my scientific work, but also believed in my ability to develop a humanoid robot, ARMAR-6. In this way he enabled me to fulfil a childhood dream. I would also like to thank Prof. Dr. Karsten Berns for co-supervising this thesis. During my visit in Kaiserslautern he spent much time discussing my topic with me and gave me valuable feedback.

My thanks also go to my colleagues at H^{2}T. In particular, I would like to thank Lukas Kaul, Pascal Weiner, Dmitriy Shingarey, Hans Haubert and Peter Kretzler who worked with me on ARMAR-6. I will always have fond memories of this time and will always be proud of what we achieved together. With Pascal Weiner and Julia Starke I not only worked on many robotic and prosthetic hands, but they were also my office colleagues and we spent many evenings together. Like Simon Ottenhaus, Isabel Ehrenberger and many other colleagues at H^{2}T, I also regard them as friends.

Besides my colleagues, I would also like to thank the many students I have supervised. In particular Oliver Karrenbauer, Hennes Fischbach and Daniel Förster provided valuable contributions to my scientific work.

Finally I would like to thank my family, especially my partner Sonja as well as my parents Gigi and Michael. Thank you so much for always supporting me.

Karlsruhe, July 2020 *Samuel Rader*

Contents

1. Introduction

The development of *humanoid robots* is a time-consuming, complex and challenging task. In order to create versatile humanoid robots, it is not only necessary to implement intelligent behavior, but also to design high-performance hardware components.

One of the greatest challenges in the design of humanoid robots is to build technical systems that have both human-like capabilities and human-like appearance. Due to this goal, the development of highly integrated robot components is necessary. Various, mostly mechatronic, subcomponents have to be combined in a very limited space, each with its own interfaces and installation constraints. The consequence of the resulting dependencies is that selection and arrangement of the subcomponents cannot be performed separately if an optimal result is to be achieved. Another challenge are the many, often conflicting requirements, such as high mechanical performance at low weight, which require the robot developers to find suitable trade-offs. As a result, the mechatronic design of humanoid robot components requires both *procedural knowledge* on systematic design and *domain-specific expert knowledge*.

Procedural knowledge for the systematic development of technical systems is provided through product development methodology and systems engineering. The provided instructions, tools and models are typically used in large companies or for complex projects in interdisciplinary domains with many different people involved, such as space research. However, without the necessary domain-specific expert knowledge, such systematic procedures cannot be used (Pahl et al., 2007).

Domain-specific expert knowledge is largely gained through experience and bound to individuals, the experts. Often, these experts have difficulties in passing on detailed knowledge to successors without gaps. As a result, there is a risk that knowledge is lost partially or completely. In the worst case, new developments have to be started from scratch. Research in the field of artificial intelligence investigates the question of how domain-specific expert knowledge can be represented and used through reasoning. To this end, *expert systems* have been developed since the 1960s. These programs are based on thought processes and specific domain knowledge of experts and serve to support users in solving problems (Görz, 1993).

Although methods of knowledge management and product development are becoming increasingly common in the automotive industry and other areas, there is still no systematic approach that supports the design of highly integrated humanoid robot components by using existing expert knowledge.

1.1. Problem Statement

This thesis addresses the following research question:

> *How can expert knowledge on the design of humanoid robot components be preserved in order to support future developments?*

To this end, a systematic approach has to be developed that formalizes expert knowledge gained in the design process of humanoid robots and makes it usable through an expert system. To start the expert system, the user defines technical requirements, which are decomposed and assigned to elements of the system model. Based on this, potential solution principles and subcomponents are evaluated and combined so that the expert system can generate valid overall solutions that are presented to the user.

The expert system should support a large part of the robot design process. In the case of NASA Systems Engineering (Kapurch, 2010), this encompasses technical requirement definition, logical decomposition and design solution definition (Figure 1.1). As a result, robot design should be significantly simplified for both novices and experts. Since humanoid robots represent highly integrated mechatronic systems whose components are arranged in a limited construction space, the approach must allow a high level of detail up to existing catalog components. Thus, it is not only necessary that the approach is based on existing catalog components, but that it also considers their installation requirements, dependencies and other constraints.

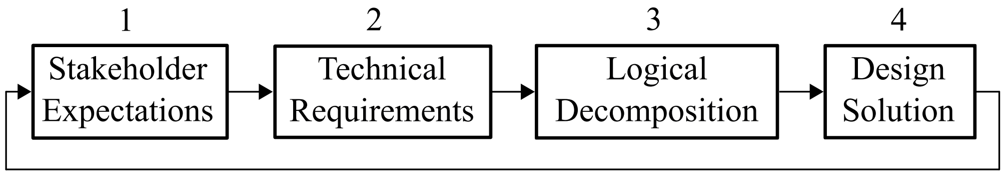


Figure 1.1.: Steps of the design phase based on Kapurch (2010).

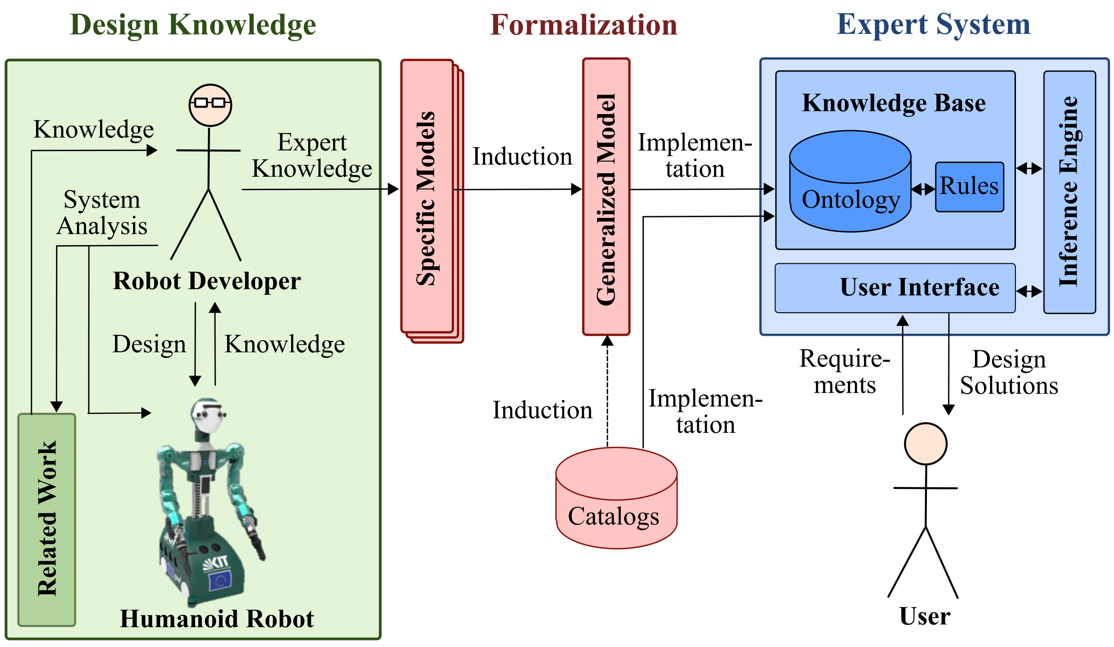


Figure 1.2.: Overview of the approach presented in this thesis, which allows to formalize design knowledge about humanoid robot components and make it usable as part of an expert system that supports future developments.

1.2. Contributions

The thesis presents a novel approach that supports the systematic design of humanoid robots (Figure 1.2). *Design knowledge* gained from the development of humanoid robots and system analysis is *formalized* and stored in an ontological knowledge base. This knowledge base is part of an *expert system* that generates design solutions for robot components based on user requirements. The described approach is evaluated using the example of components for the humanoid robot ARMAR-6. The resulting contributions of this thesis can be structured into three parts, which can be summarized as follows.

Formalization of Expert Knowledge on Humanoid Robot Design: The first contribution of this thesis is a novel formalization process that describes how design knowledge on humanoid robot components can be stored in an ontological knowledge base that allows for automated reasoning. It starts with the acquisition of design knowledge. Robot developers gain this knowledge by designing humanoid robot components and analyzing related work. Based on system analyses, specific models are built for each robot component that serves as knowledge source. From these specific models a generalized model of the robot component type is induced, which is represented by a directed acyclic graph. The graph describes the hierarchical structure of the robot component as a combination of subsystems, which themselves consist of subsystems. Thus, a robot component is modeled hierarchically from the overall system down to placeholders for catalog components. Different solution principles are represented by additional nodes and paths. An ontology is then derived from the graph, which describes the structure of the robot component. Knowledge about existing catalog components is stored in another ontology, which considers not only physical data provided by the manufacturer but also hierarchical relationships and compatibilities between the components. A parameterized rule set describes the calculation of the properties of partial solutions depending on the concrete data of the catalog components. It also defines when requirements are satisfied.

Expert System to Support the Design of Humanoid Robot Components: The second contribution of this thesis is an expert system that supports the design of humanoid robot components based on the formalized expert knowledge. The ontologies and the rule set form the knowledge base of the expert system. Further components of the expert system are an inference engine and a user interface. By entering requirements, the system starts a multi-stage reasoning process in which design solutions are generated in a bottom-up approach based on the knowledge base. According to the graph describing the robot component, first partial solutions are created by combining existing catalog components and solution principles. The properties of the partial solutions are then calculated and compared to the requirements and constraints. Partial solutions that do not meet the requirements or constraints are discarded. All remaining partial solutions are combined with other partial solutions in a bottom-up approach until complete, valid design solutions for a robot component type are created. The resulting design solutions include all necessary catalog components, the selected solution principles and the calculated properties of the combination. They are visualized via the user interface and can be evaluated according to both single performance requirements and weighted optimization functions. Compared to related work in the field of automated robotic and mechatronic design, this approach has a high level of detail. Catalog components are automatically selected and arranged, without the usual simplifications such as the use of modules or the use of only a single parameterized solution.

Evaluation Using the Example of the Humanoid Robot ARMAR-6: The third contribution of the thesis is the evaluation of the approach using the example of components for ARMAR-6, a collaborative humanoid robot for industrial environments (Figure 1.3). This includes the design of mechatronic components for

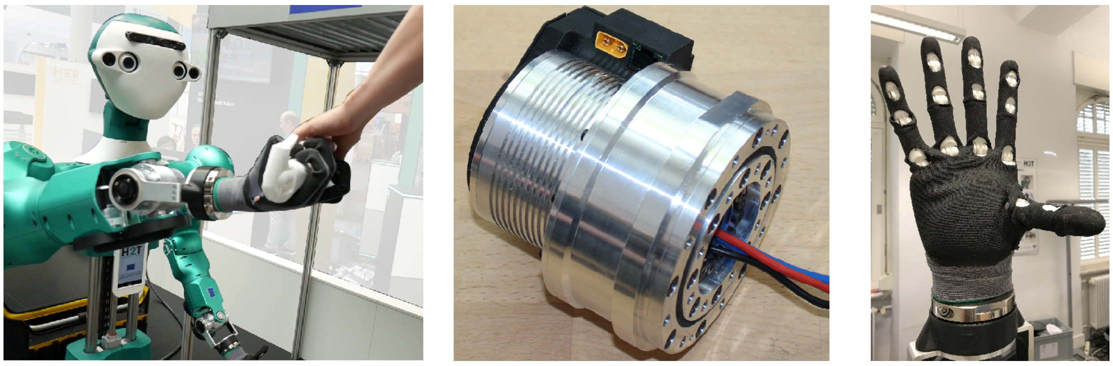


Figure 1.3.: Components of the humanoid robot ARMAR-6 (left) are used to evaluate the approach: SAC units for robot joints (middle) and robotic hands (right). *Source (from left to right):* (Asfour et al., 2019b) © 2019 IEEE, (Rader et al., 2017) © 2017 IEEE, (Asfour et al., 2018) © 2015 IEEE.

ARMAR-6, most notably sensor-actuator-controller (SAC) units for robot joints. In comparison to related work, the SAC units are characterized by a high degree of integration of subcomponents and functionalities. As this also makes them the mechatronic components of ARMAR-6 with the highest degree of integration, they are used as a case study to evaluate the approach presented in this thesis. To demonstrate that the approach can be used for a wide range of robot components, robotic and prosthetic hands serve as a second case study. In both case studies, the design knowledge is formalized and integrated into an expert system. The evaluation shows that the expert systems can reproduce, optimize and scale existing robot components. Furthermore, it is demonstrated that the expert systems can also be used for the generation of novel designs by combining solution principles and catalog components in a novel way.

1.3. Structure of the Thesis

This thesis is divided into six chapters.

Chapter 2 presents *fundamentals* of (1) systematic design methodology as well as (2) knowledge representation and reasoning, two research fields that are combined by the approach presented in this thesis. In particular, classical product development, systems engineering, expert systems and ontologies as well as other knowledge representation techniques are introduced. In addition, *related work* on automated design of robotic and mechatronic systems as well as rotary sensor-actuator units for robot joints is presented and compared.

Chapter 3 introduces the first and second step of the approach, (1) the *design knowledge acquisition* and (2) its *formalization*. It is discussed how expert knowledge can be gained by designing humanoid robot components and by analyzing related work. Furthermore, it is described which knowledge has to be gained by system analysis to perform the formalization process. The formalization process is then explained step by step. It is described how design knowledge is used to create specific models, on the basis of which a generalized model for a robot component type is developed. Subsequently, it is explained how this generalized model can be represented by ontologies and rules to create a modular ontological knowledge base for expert systems. In this context it is also motivated why ontologies are a suitable form of knowledge representation.

Chapter 4 presents the third and last step of the approach, the development of an *expert system* that supports the design of humanoid robot components. Since the ontological knowledge base is already described in chapter 3, the two other components of the expert system architecture are described in this section: The inference engine and the user interface. This includes a description of a novel multi-stage reasoning procedure, a runtime analysis and a description of the rating functions to evaluate design solutions.

Chapter 5 presents the *evaluation* of the approach using the example of components for the humanoid robot ARMAR-6. First, the overall design of ARMAR-6 is presented. Thereafter, the two robot components with the highest degree of integration, the SAC units and robotic hands, serve as case studies to evaluate the approach presented in this thesis. For both case studies it is shown how design knowledge can be formalized to develop an expert system. To demonstrate the capabilities of these expert systems, existing units are reproduced, optimized and scaled as well as novel designs are generated.

Chapter 6 serves as a *conclusion* to this thesis. It summarizes the contributions, discusses its limitations and presents possibilities for future work.

2. Fundamentals and Related Work

This section presents fundamentals of (1) systematic design methodology as well as (2) knowledge representation and reasoning, two research fields which are combined by the approach presented in this work. Firstly, guidelines for the systematic design of technical systems, classical product development and systems engineering are described. Secondly, fundamentals from knowledge representation and reasoning are explained, a subfield of artificial intelligence. In particular, these include expert systems and knowledge representation techniques such as ontologies.

Thereafter, related work in the field of automated design of mechatronic and robotic systems is presented and categorized. Here, the focus is on expert systems and other knowledge-based systems that support detailed design. Furthermore, related work on sensor-actuator units for robot joints is presented to show the comparatively high degree of integration of the SAC units, which serve as a case study for the approach presented in this thesis.

2.1. Product Development and Systems Engineering

Product development and systems engineering both serve to provide guidelines, tools and methods to systematically support the design process. In the following, the fundamentals from both areas are described, as well as morphological analysis, a popular creativity method adapted for this work.

2.1.1. Types of Design Tasks

Depending on the novelty of a product, Pahl et al. (2007) distinguish between three different types of design tasks.

- **Original design:** New tasks and problems are solved either by new combinations of known solution principles or by inventing new technology.

- **Adaptive design:** The solution principles remain unchanged. Only the embodiment is adapted to new requirements and constraints.
- **Variant design:** Size and arrangement of the subcomponents are varied within already designed product structures. Variant designs are typical for product series and modular systems

Most design tasks are adaptive or variant designs, which however are not necessarily less challenging than original designs (Pahl et al., 2007).

2.1.2. Product Development Process

Designers of technical systems have a significant responsibility in terms of technical and commercial properties of a product. In order to assist them in an efficient product development process, various systematic design procedures have been developed. A distinction can be made between *design sciences*, which use methods for the analysis of technical systems, and *design methodologies*, which describe a concrete course of action (Pahl et al., 2007).

Main Phases of the Planning and Design Process

One of the most prominent design methodologies is presented by Pahl et al. (2007). Their activity plan shares similarities with the German VDI Guidelines 2221 and 2222 and provides a description of the work flow during the design process, focusing on mechanical engineering. Pahl et al. (2007) divide the planning and design process into four main phases:

I **Planning and Task Clarification:** In this phase, the task is clarified in detail by collecting information about the requirements to be met. The result is a requirements list that serves as a *specification* of the design.

II **Conceptual Design:** In the conceptual design phase, *principle solutions* are determined based on the requirements list. To this end, essential problems are identified and function structures are established. The designer searches for suitable *solution principles* (also called working principles) and combines them into concept variants. These variants are then first filtered for satisfying the requirements and second evaluated on the basis of technical and economic criteria. It is possible that the result of this phase is not only one but several principle solutions[1] and that the decision about which principle solution is best can only be made on a more detailed level.

[1]Please note the difference between solution principles and principle solutions. Principle solutions are a combination of solution principles.

III **Embodiment Design:** Starting from principle solutions, the designer determines the construction structure, the definitive overall layout of the system. For this purpose, layouts are refined and improved successively. The embodiment phase can be divided into two subphases, (1) the *development* and (2) the *definition* of the design structure. The result of the development subphase are the *preliminary layouts*. It is often necessary to create several preliminary layouts in order to obtain more information about the advantages and disadvantages of the individual variants. The development of a preliminary layout includes preliminary form design, material selection and calculation. Based on this more detailed information, it is possible to refine the evaluation with regard to technical and economic criteria and to select the best preliminary layouts. In order to obtain the *definitive layout*, weak points are eliminated, often by combining different variants. The definitive layout allows checking for functionalities, spatial compatibility, costs, strength and other requirements.

IV **Detail Design:** The arrangement, dimensions, forms, materials, costs, production possibilities and other properties of all individual parts of the system are finally determined. The result of this final phase is the solution as well as the corresponding *product documentation* that includes drawings and other production documents.

Table 2.1.: General procedure for development and design according to VDI 2221. The phases (I-IV) correspond to the main phases used by Pahl et al. (2007).

Stage	Description	Results	Phases
1	Clarify and define the task	Specification	I
2	Determine functions and their structure	Function structures	II
3	Search for solution principles and their combinations	Principle solutions	II
4	Divide into realizable modules	Module structures	II, III
5	Develop layout of key modules	Preliminary layouts	III
6	Complete overall layout	Definitive layout	III
7	Prepare product and operating instructions	Product documentation	IV

VDI 2221 Guideline

The VDI guideline 2221 describes a systematic approach to the development and design of technical systems and products (VDI, 1993). Starting from a given task, the general procedure encompasses seven stages, from which seven corresponding work results are derived (Table 2.1). The seven stages can be assigned to four main phases (I-IV). For mechanical systems, they correspond to the main phases used by Pahl et al. (2007). Besides the division of the conceptual and embodiment design into several stages with corresponding work results, the biggest difference lies in the explicitly mentioned modularization (stage 4 and 5). In this context, modularization does not necessarily mean that the system can only consist of independent hardware modules. Rather, in stage 4, a division into groups or parts which are decisive for the later design should take place. In stage 5, these decisive modules are designed. They serve as a starting-point for the design of the definitive layout (stage 6).

2.1.3. NASA Systems Engineering

Systems engineering (SE) is an interdisciplinary and methodical approach to support the design, realization and technical management of complex technical systems. In terms of interdisciplinarity and scope, it goes further than classical product development. System means a collection of various elements which are required for the realization of the product. Elements are therefore not only hardware and software, but also people, institutions, policies and documents (Kapurch, 2010).

NASA Systems Engineering describes 17 common technical processes and divides them into three sets (Kapurch, 2010):

- **System Design Processes (1-4):** Starting from stakeholder expectations, technical requirements are generated and converted into a design solution. These processes are applied to every product of the system structure, starting from the top level and ending at the bottom level. The bottom level is reached when parts can be built, purchased or reused.
- **Product Realization Processes (5-9):** Starting from the bottom level, the product realization processes are applied to all products of the system structure. This includes the implementation and integration of the individual parts, but also evaluation processes and the product transition to the next hierarchy level.

- **Technical Management Processes (10-17):** These processes are the link between the management and the technical team. Technical plans for the project are created, communication interfaces are managed and the progress of plans or requirements is assessed.

Since the procedure for the development of technical systems is particularly relevant for this work, the system design processes are described in detail in the following.

System Design Processes

System design is divided into four processes (Kapurch, 2010):

1. **Stakeholder Expectations Definition:** This is the initial process within SE. Its main purpose is the identification of the stakeholders and how they want to use the product to be designed. Typical outputs are top-level requirements and expectations as well as descriptions of the system from an operational perspective.
2. **Technical Requirements Definition:** This process transforms the stakeholder expectations into a set of technical requirements that represent a complete description of the problem. The technical requirements are used for defining a design solution in the product breakdown structure model. Another output are technical measures to assess product effectiveness and customer satisfaction. The technical requirements definition is a recursive and iterative process.
3. **Logical Decomposition:** Taking technical requirements and measures as input, detailed functional requirements are created in the logical decomposition process. In the process, functional analysis is used to create a system architecture. Requirements are decomposed and allocated to the lowest level selected. In addition to the system architecture model, the end product requirements are an output of this process.
4. **Design Solution Definition:** This process is used to transform the technical requirements and logical decomposition models into a design solution. First, alternative solutions are formed and analyzed by detailed trading studies. Based on these results, a preferred alternative is selected and fully defined to obtain a final design solution that satisfies all requirements. Results of the design solution definition are specifications and plans for the product realization processes.

Compared to classical product development described by VDI (1993) and Pahl et al. (2007), the system design processes of NASA Systems Engineering focus more on the conceptual phase. This can be explained by the fact that the interdisciplinarity and complexity of typical Systems Engineering projects requires a more elaborate description of the problem. Of particular importance are requirements that have to be defined, decomposed and allocated.

Requirements: Types, Decomposition and Allocation

Kapurch (2010) distinguishes between different technical requirements, of which the most important types are:

- **Functional Requirements:** They define *what functions* have to be fulfilled in order to achieve the goals during the product's entire lifetime.
- **Performance Requirements:** They define *how well* these functions have to be performed. If possible, these quantitative requirements should be defined as threshold values and baseline levels of the desired performance.
- **Interface Requirements:** They define the external interfaces between the system boundaries and its environment.

During the *technical requirements definition process*, requirements are hierarchically structured. First, high-level requirements are decomposed into functional and performance requirements and assigned across the system. These requirements are then further decomposed and allocated to subsystems and system elements until a complete set of requirements is achieved. The derivation and allocation of technical requirements is part of an iterative design loop and closely related to the system architecture model defined in the *logical decomposition process*. It is completed when the functional and performance analysis of the *design solution definition process* confirms that sufficient depth has been achieved. This results in recursive links between the four system design processes.

SysML

In the following, the *Systems Modeling Language (SysML)* is described, based on the book *"SysML Distilled: A Brief Guide to the Systems Modeling Language"* (Delligatti, 2014). SysML is a graphical modeling language for systems engineering applications. It is based on a subset of the *Unified Modelling Language (UML)* and extends it with systems engineering capabilities. The grammar and notations for SysML are defined in a standard specification published by object

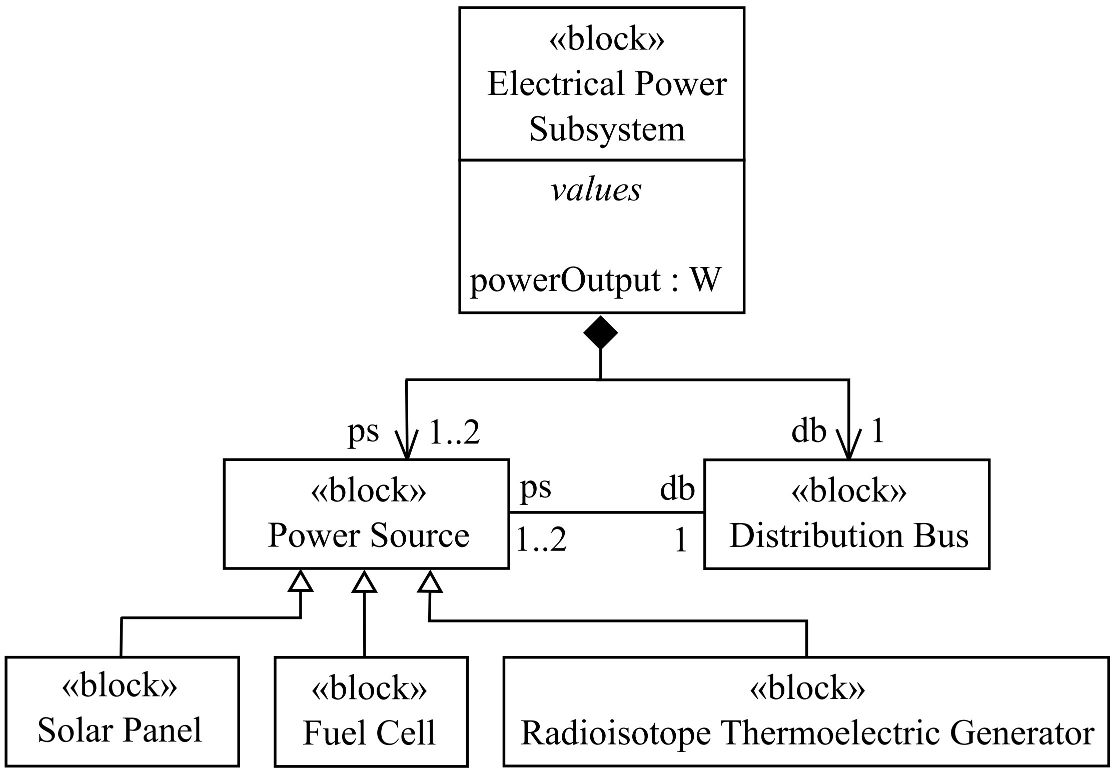


Figure 2.1.: SysML Block Definition Diagram (BDD) of the electrical power subsystem of a satellite, based on Delligatti (2014).

management group (OMG). SysML provides the means to communicate and visualize important aspects of a system design such as structure, behavior and requirements. In particular, SysML features different diagrams types.

Block definition diagrams (BDD) are the most common type of SysML diagrams and used to visualize system hierarchy and classification trees. An example for a block diagram that visualizes the electrical power subsystem of a satellite is given in Figure 2.1. Block diagrams consist of entities and relationships between them.

Blocks, the most important elements in BDD, can model any type of entity in the system. They are visualized by rectangles and can display optional compartments such as part properties, reference properties, value properties, constraint properties or operations. Figure 2.1 displays the blocks *Electrical Power System*, *Power Source*, *Distribution Bus*, *Solar Panel*, *Fuel Cell* and *Radioisotope Thermoelectric Generator*. The block *Electrical Power Subsystem* has a value property with the name *powerOutput* and the type *W* (Watt).

Relationships between the blocks are visualized by lines. A solid line with a solid diamond expresses a special kind of relationship, a **composite association**: An instance of a composite is made up of instances of its parts. Multiplicities of blocks in these relationships are optionally expressed by small numbers near the corresponding blocks. For example, an instance of *Electrical Power Subsystem* (composite) consists of one or two instances of *Power Source* (part) and exactly one instance of *Distribution Bus* (part). Another important relationship between two elements are **generalizations**: They convey inheritance between a more generalized element, the supertype, and a more specialized element, the subtype. Generalizations are visualized by a solid line with a hollow triangle as arrowhead on the end of the supertype. In the presented example, *Solar Panel*, *Fuel Cell* and *Radioisotope Thermoelectric Generator* are subtypes of the supertype *Power Source*.

For a more detailed description of SysML please refer to Delligatti (2014).

2.1.4. Morphological Analysis

Morphological analysis is a heuristic idea generation method, introduced by Fritz Zwicky (1948). Core of the morphological analysis is the creation of a *morphological box*, also known as *morphological matrix* or *Zwicky box*. The basic idea is to use a classification scheme to develop overall solutions through systematic combination. The procedure for creating and using a morphological box is as follows:

1. The problem is precisely formulated.
2. The relevant parameters for the problem are determined. The parameters should be independent of each other.
3. For every parameter, all possible characteristics are determined and entered in the row of a matrix: the morphological box.
4. Overall solutions are built by combining the parameters systematically: Exactly one characteristic is selected for each parameter, i.e. one entry in each row of the matrix. Zwicky and Wilson (1967) propose to build, analyze and evaluate every solution contained in the morphological box.
5. Finally, the best solutions are selected and realized. Zwicky and Wilson (1967) suggest an additional morphological analysis for the practical application.

Although morphological analysis has already been used in various fields, it is especially popular for engineering design. This can be explained by the large number of variables that have to be taken into account for technical problems

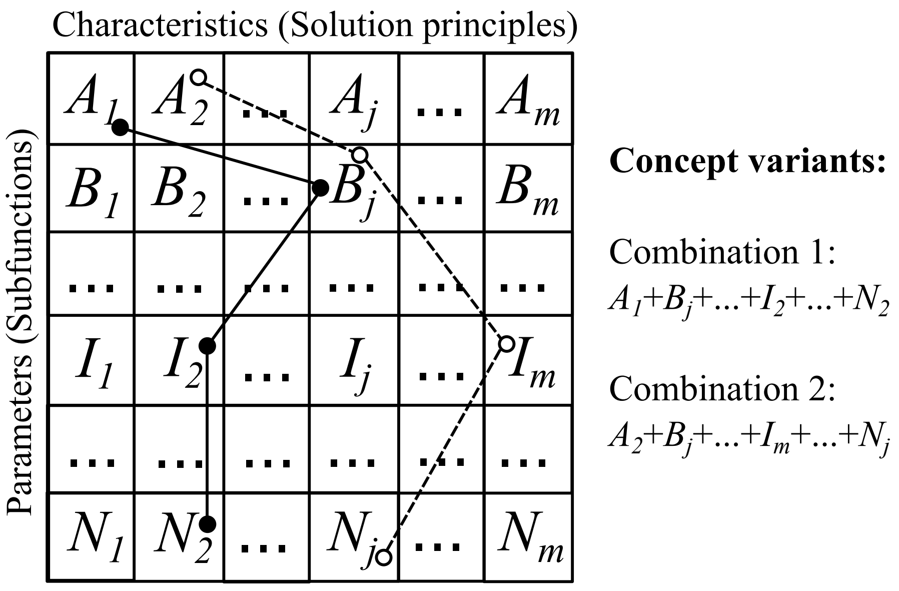

Characteristics (Solution principles)
Parameters (Subfunctions)
A_1 A_2 ... A_j ... A_m
B_1 B_2 ... B_j ... B_m
... ...
I_1 I_2 ... I_j ... I_m
... ...
N_1 N_2 ... N_j ... N_m
Concept variants:
Combination 1:
$A_1+B_j+...+I_2+...+N_2$
Combination 2:
$A_2+B_j+...+I_m+...+N_j$

Figure 2.2.: Example for a morphological box that allows to combine solution principles for different subfunctions to find concept variants.

(Álvarez and Ritchey, 2015). Especially in the *conceptual design phase* of the design process (subsection 2.1.2), morphological analysis has proven to be particularly useful (Pahl et al., 2007). As shown in Figure 2.2, the combination of solution principles into concept variants can be conducted with the help of a morphological box. Here *subfunctions* correspond to *parameters* and *solution principles* correspond to *characteristics*. However, morphological analysis can also be used during the *embodiment design phase* to combine subsolutions, components or assemblies (Pahl et al., 2007).

A drawback of morphological analysis is that it is not always possible to manually analyze and evaluate all theoretically possible overall solutions contained in a morphological box. For n parameters and m_i possible characteristics for parameter i, step 4 of the procedure results in $\prod_{i=1}^{n} m_i$ combinations that are all theoretically possible overall solutions. Therefore, in practical applications for problems with many parameters and characteristics, often only a few overall solutions are intuitively selected by the user (Ritchey, 1998). Another possibility is to check the compatibility between characteristics of different parameters and to automate the configuration and elimination process, thus conducting a computerized morphological analysis (Eriksson and Ritchey, 2002).

2.2. Expert Systems

An expert system is a computer program that is developed based on thought processes and experience of experts in a specific domain to assist users in solving problems (Görz, 1993). Here, the goal is to reach or even surpass the quality of human experts in problem solving. Further typical characteristics are enormous quantities of domain-specific knowledge in minute detail and the use of heuristics to reduce the solution space (Tripathi, 2011).

Expert systems are located in the field of artificial intelligence (AI). They emerged in the late 1960s, after the first decade of AI research, when it became apparent that general search mechanisms were too weak for problem solving (Russell and Norvig, 2010). The main problems were poor performance for complex domains as well as limited domain knowledge (Russell and Norvig, 2010). A new approach to overcome these problems was presented by Buchanan et al. (1968), who consulted chemists in the development of DENDRAL, a program for inference of molecular structures based on mass spectrometer data. They succeeded in representing the chemists' expert knowledge in the form of rules. Thus, DENDRAL became the first successful knowledge-intensive system. Today, it is considered one of the first expert systems besides MYCIN (Shortliffe, 1974), a rule-based system for the diagnosis of blood infections. Both systems were developed at Stanford University by the "Heuristic Programming Project" group led by Edward Feigenbaum (Russell and Norvig, 2010).

2.2.1. Applications

Expert systems are used to find solutions to various problems in different industries. Clancey (1985) presents a taxonomy that allows for categorizing expert systems according to problem domains. To this end, Clancey (1985) proposes three main groups:

- Analysis problems
- Synthesis problems
- Problems combining analysis and synthesis

For example, the problem domain *diagnosis* is assigned to the main group of *analysis problems* whereas *design* is assigned to the main group of *synthesis problems* (Table 2.2).

Besides the problem domain it is also possible to differentiate between applications of expert systems by industry. The study of Wagner (2017) indicates

Table 2.2.: Taxonomy of problem domains of expert systems, based on Clancey (1985).

Analysis problems	
● Classification	Categorizing based on observables
● Debugging	Prescribing remedies for malfunctions
● Diagnosis	Inferring system malfunctions from observables
● Interpretation	Inferring situation descriptions from sensor data
Synthesis problems	
● Configuration	Configuring collections of objects under constraints in relatively small search spaces
● Design	Configuring collections of objects under constraints in relatively large search spaces
● Planning	Designing actions
● Scheduling	Planning with strong time and/or space constraints
Problems combining analysis and synthesis	
● Command and control	Ordering and governing overall system control
● Instruction	Diagnosing, debugging and student behavior
● Monitoring	Comparing observations to expected outcomes
● Prediction	Inferring likely consequences of given situations
● Repair	Executing plans to administer prescribed remedies

that expert systems are strongest in the areas of medicine, manufacturing, accounting services, banking and financial services. But also in the automotive industry and more unusual areas such as aerospace, expert systems have their applications. The study demonstrates that expert systems are used in very different industries for very different problems.

2.2.2. Architecture

Expert systems consist of several components that are logically linked and fulfill different tasks (Styczynski et al., 2017). The two major components are the knowledge base and the inference engine. In current work, the user interface is usually referred to as the third component. The classical architecture, as shown in Figure 2.3, contains additional components. In other works on architectures of expert systems, these additional components are not always listed because they are combined with other components or are generally optional. For instance, the explanation facility can be seen as part of the user interface.

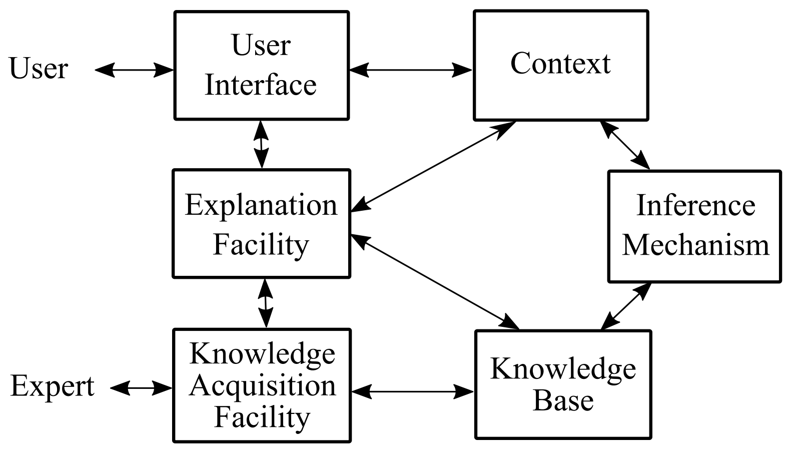


Figure 2.3.: Classical architecture of an expert system, based on Maher et al. (1984).

In the following, the classical components of expert systems illustrated in Figure 2.3 and the important human actors are described in more detail.

- **Knowledge Base:** The knowledge base contains the knowledge that is used to understand, formulate and ultimately solve problems. It can contain both facts and heuristic knowledge. The knowledge base is characterized by the chosen knowledge representation techniques that make the knowledge machine-readable. For instance, the knowledge base of a rule-based expert system contains a set of IF-THEN rules. Knowledge representation techniques are introduced in detail in subsection 2.2.3.
- **Inference Engine (Inference Mechanism):** The inference engine (also called inference mechanism) controls the processing of the program and provides a methodology for reasoning (Tripathi, 2011). Based on a user request, it infers conclusions from the knowledge base. Knowledge reasoning techniques depend on the chosen knowledge representation and are described in subsection 2.2.5.
- **Context:** The context is a collection of symbols and facts reflecting the program's current state and contains all information generated in the course of the current program run (Maher et al., 1984). It is modified by the inference engine.
- **User Interface:** The user interface facilitates communication between the user and the expert system. For this purpose, it converts knowledge from its internal representation to a user understandable form and vice versa (Tripathi, 2011). Specifically, the user interface allows the input of data based on which the inference engine performs reasoning. The inferred solutions are then displayed via the user interface.

- **Explanation Facility:** The explanation facility explains the expert system's actions and serves to share knowledge of the system with the user (Tripathi, 2011).
- **Knowledge Acquisition Facility:** The knowledge acquisition facility is a subsystem that helps to build the knowledge base. Problem-solving expertise has to be accumulated, transferred and transformed to a computer program (Tripathi, 2011).
- **Human Actors:** Users are the target group that the system is intended to serve. They can, but do not usually have to be experts to use the expert system. Experts are needed to make their expertise available. They provide domain-specific knowledge for the knowledge base. This knowledge can be entered either directly by the expert or by a knowledge engineer. In the development of many expert systems, the domain expert and the knowledge engineer are one and the same person (Wagner, 2017).

2.2.3. Knowledge Representation

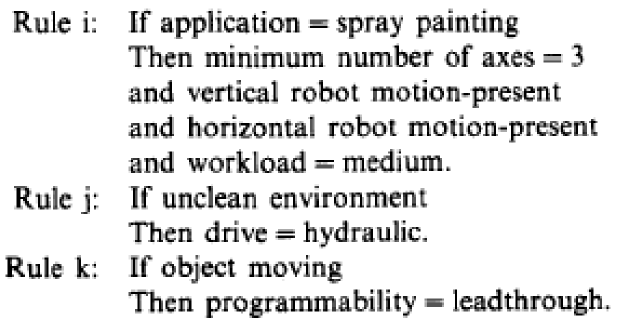
In order to make knowledge machine-readable, it must be formalized (Jakus et al., 2013). Davis et al. (1993) explain the concept of knowledge representation by the five roles it plays:

1. **Surrogate:** It represents its counterpart in the real world.
2. **Set of Ontological Commitment:** It represents only a part of the real thing.
3. **Fragmentary Theory of Intelligent Reasoning:** It is expressed by its conception of intelligent reasoning and the set of inferences the representation sanctions as well as those it recommends.
4. **Medium of Efficient Computation:** In order to use a representation, it must be possible to compute with it. As a consequence, computational efficiency is of importance.
5. **Medium of Human Expression:** It is a medium to communicate with machines and other humans.

With regard to the fifth role, natural language is the most widespread *medium of human expression*, but difficult to use for computer systems (Jakus et al., 2013). This is why in most cases, machine-readable formalisms are used to represent knowledge in expert systems. The most commonly mentioned formalisms are rules, semantic networks, frames and description logics. They are described in more detail in the following, along with other knowledge representation techniques. Due to their importance for this work, ontologies are discussed separately in the following subsection 2.2.4.

Rules

According to a study presented by Wagner (2017), rules are the most frequently used knowledge representation technique for expert systems. Rules are conditional declarations linking given conditions to actions (Abraham, 2005). If a rule is satisfied, logical conclusions must be made and existing facts can be extended or changed (Styczynski et al., 2017). The most commonly used rules, production rules, are written in the form of IF-THEN rules. Examples for IF-THEN rules are shown in Figure 2.4. The three rules listed describe the mapping of high-level requirements (e. g. application spray painting) to a list of constraints for the selection of possible robotic systems (Boubekri et al., 1991).



```
Rule i:  If application = spray painting
         Then minimum number of axes = 3
         and vertical robot motion-present
         and horizontal robot motion-present
         and workload = medium.
Rule j:  If unclean environment
         Then drive = hydraulic.
Rule k:  If object moving
         Then programmability = leadthrough.
```

Figure 2.4.: Rule-based expert system for the selection of robot arms, using IF-THEN rules. *Source:* (Boubekri et al., 1991) © 1991 Elsevier Science Ltd.

The example demonstrates that IF-THEN rules represent knowledge in a natural way that is easy to understand and needs no translation (Nagori and Trivedi, 2014). Another advantage of rule-based systems is the strict separation of the knowledge base containing rules and the inference engine, which allows both parts of the expert system to be updated independently of one another (Nagori and Trivedi, 2014).

However, a single IF-THEN rule can only encode a small chunk of knowledge, which results in a knowledge base with a large number of rules even for simple problems (Maher et al., 1984). Domain knowledge is often represented by hundreds of rules and when facts are changed by the effect of rules, all rules should be re-examined to check whether they lead to a different conclusion (Styczynski et al., 2017). To overcome these problems, rules are often combined with other knowledge representation techniques.

By using predicate logic, IF-THEN rules can be recasted into Horn-clauses (Maher et al., 1984). A Horn-clause is a set of literals with at most one unnegated literal (Maher et al., 1984). The rule "If A and B and C then D" can be expressed by clause 2.1 which is equivalent to Horn-clause 2.2.

$$A \wedge B \wedge C \Rightarrow D \tag{2.1}$$

$$\neg A \vee \neg B \vee \neg C \vee D \tag{2.2}$$

This mapping of IF-THEN rules allows for taking advantage of logical programming.

Semantic Networks

Semantic networks are graphical structures that represent static world knowledge (Jakus et al., 2013). The nodes of the graph represent objects, the edges represent the semantic binary relationships between the objects, e. g. "has", "is a" or "needs" (Styczynski et al., 2017).

Frames

Frames, first introduced by Minsky (1975), are structures inspired by the organization of the human memory. They build on semantic networks (Jakus et al., 2013), but take a more object-oriented approach. Frames facilitate the representation of the entire knowledge of objects described by different, specific properties (Styczynski et al., 2017). These properties are called slots and may include information on the frame name, the relationship between frames, a range of slot values, a default slot value and procedural knowledge (Nagori and Trivedi, 2014). The instances of a concrete object are determined by the fact that each slot is concretely assigned a value (Styczynski et al., 2017).

Description Logics

Description logics are a family of knowledge representation languages that combine concept descriptions from their predecessors, semantic networks and frames, with logic-based semantics (Baader et al., 2008). They use decidable fragments of first order logic, but are more expressive than propositional logic. As a result, description logics represent knowledge in a structured and formally easily comprehensible way (Baader et al., 2008). Furthermore, they have good computational properties, resulting in efficient reasoning (Jakus et al., 2013).

Cases

In contrast to approaches using classical knowledge representations, case-based reasoning (CBR) does not rely on general knowledge, but on specific knowledge of previously experienced cases (Aamodt and Plaza, 1994). In order to solve a new problem, a similar older case is searched for, applied to the real world environment and modified if necessary (Aamodt and Plaza, 1994). In order to make use of the gained experience for future developments, the knowledge base is updated by modifying or adding new cases (Aamodt and Plaza, 1994).

Decision Trees

A decision tree represents a function that takes a vector of attributes as input and returns a single output value, the decision, by performing a series of tests (Russell and Norvig, 2010). The inner nodes of the tree represent these tests, edges correspond to possible attributes, e. g. true and false (Russell and Norvig, 2010). Figure 2.5 illustrates the principle of a decision tree used by an expert system to test all robots in the knowledge base for meeting requirements.

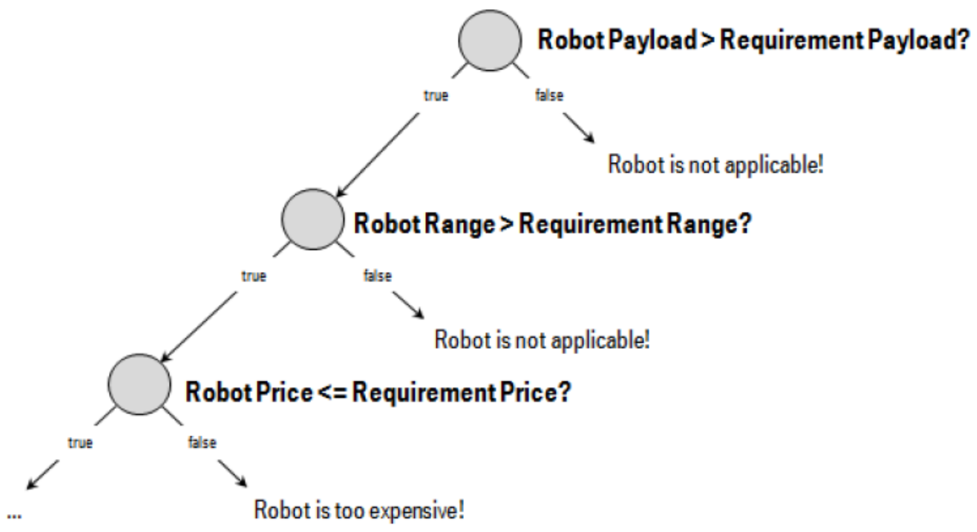

Robot Payload > Requirement Payload?
true
false
Robot is not applicable!
Robot Range > Requirement Range?
true
false
Robot is not applicable!
Robot Price <= Requirement Price?
true
false
...
Robot is too expensive!

Figure 2.5.: Expert system for robot selection, using a decision tree.
Source: (Keller et al., 2016) © 2016 IEEE.

Uncertainty

In order to take human uncertainty and vagueness into account, fuzzy logic and Bayesian networks, are used for knowledge representation.

Fuzzy Logic is an extension of the traditional logical systems (Jakus et al., 2013). It is a many-valued logic that has properties aiming at modeling the vagueness of natural language via a graded approach (Novák et al., 1999). Each value is not necessarily true (1) or false (0), but rather represented by a number between 0 and 1.

An alternative to fuzzy logics are Bayesian networks, also known as belief networks. They are probabilistic directed acyclic graph (DAG) models (Russell and Norvig, 2010). Each node represents a random variable whereas the edges represent probabilistic dependencies between the random variables (Ben-Gal, 2008). The estimation of these conditional dependencies is often based on statistical and computational methods (Ben-Gal, 2008).

Artificial Neuronal Networks

Artificial neuronal networks are not a classical knowledge representation for expert systems. However, they can be combined with the other presented techniques. An example for such a hybrid system is given by Yang et al. (2004). It combines case-based reasoning with a neuronal network to enhance fault diagnosis.

2.2.4. Ontologies

The term ontology originates from philosophy, in which it describes the study of *"being qua being"*, the furniture and entities of reality (Øhrstrøm et al., 2005). In modern computer sciences, one of the most cited definitions originates from Studer et al. (1998), who combined the definitions of Gruber (1993) and Borst (1997): *"An ontology is a formal, explicit specification of a shared conceptualization."*

Ontologies define entities and their relations in a domain of interest. Entities can be classes, instances of the classes (called individuals), properties and data types. Similar to taxonomies, ontologies can be used to classify objects hierarchically. However, they go much further and are not limited to a strict hierarchy. They can include several taxonomies, allow for multiple inheritance and are able to describe relations between different concepts. Various languages have been developed to formalize ontologies. The most widely used are the *web ontology language OWL* (Horrocks et al., 2003) and its successor *OWL2* (Grau et al., 2008). By using description logics, they combine the advantages of different representation techniques presented in subsection 2.2.3, in particular semantic networks, frames and first order logics. Initially, rule-based knowledge representation was an alternative to ontologies and widely used for industrial application (Staab and Studer, 2004). In order to benefit from the advantages of both knowledge representations, the *Semantic Web Rule Language SWRL* (O'Connor et al., 2005) was developed, allowing the addition of rules and Horn-clauses to the expressive power of *OWL*.

Ontology Components

Ontologies consist of different components. The most important components in OWL are presented in the following (Horridge et al., 2004):

- **Classes** are the main building blocks of an OWL ontology. They are used to define and categorize objects. Class hierarchies, also called taxonomies, can be created by inheritance.
- **Individuals** are *instances* of OWL classes. Thus classes can also be interpreted as a set of individuals.
- **Properties** are used to describe binary relations between individuals and other entities. A distinction is made between different kinds of properties:
  - **Object properties** link an individual to an individual. For the specific individuals this is called an *object assertion*.
  - **Data properties** relate an individual to a concrete data value. For a specific individual and value this is called a *data assertion*.

Upper Ontologies

One of the most important goals of ontology engineering is to share knowledge between different domains of discourse by combining ontologies. However, due to the ambiguity of semantic expressions it is often difficult to match concepts and relations. *Upper Ontologies* try to overcome this problem by defining common frameworks for concepts with placeholders that allow adding new knowledge for other domains (Russell and Norvig, 2016).

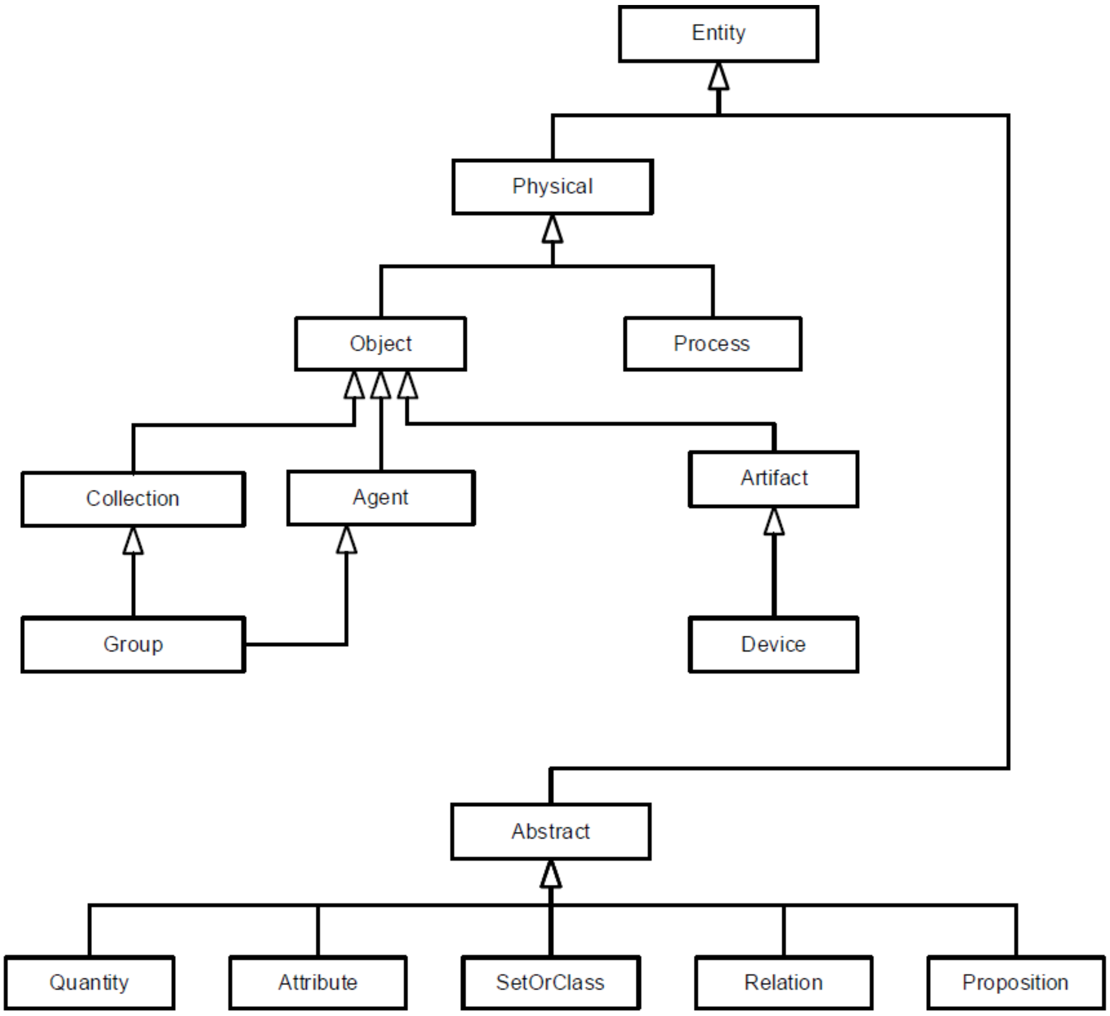


Figure 2.6.: Basis taxonomy of the Suggested Upper Merged Ontology (SUMO).
Source: (IEEE Robotics and Automation Society, 2015) © 2015 IEEE.

An example for a commonly used Upper Ontology is the Suggested Upper Merged Ontology (SUMO). It was developed by the IEEE Standard Upper Ontology Working Group with the aim of developing *"a standard upper ontology that will promote data interoperability, information search and retrieval, automated inferencing, and natural language processing"* (Pease et al., 2002). Figure 2.6 illustrates the basic SUMO taxonomy. Its main category is *Entity*, which can be divided into *Abstract* and *Physical* concepts. Subcategories can be further broken into subsubcategories such as *Object* or *Process* and so on. The general terms defined in SUMO make it very easy to match it with new ontologies for different domains.

Ontologies for Robotics

The Core Ontology for Robotics and Automation (CORA) is a domain specific upper ontology developed by a working group of the IEEE Robotics and Automation Society (2015) that aims to *"provide a methodology for knowledge representation and reasoning in robotics and automation"*. CORA uses SUMO as upper ontology, which allows to describe new entities like *Robot*, *ArtificialSystem*, *RoboticSystem*, *PhysicalEnvironment* and *RoboticEnvironment* by relations to categories defined in SUMO (Figure 2.7).

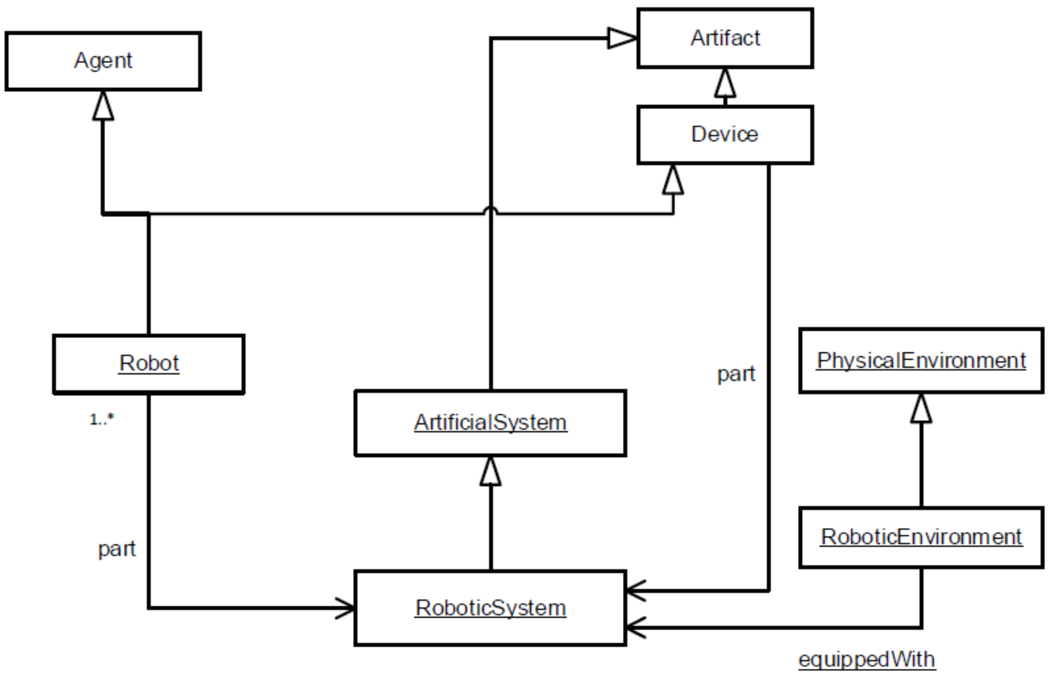


Figure 2.7.: Categories and relations defined in CORA to describe robotic systems and environments. *Source:* (IEEE Robotics and Automation Society, 2015) © 2015 IEEE.

In the field of service robotics, various ontologies have been developed (Haidegger et al., 2013). However, few (Juarez et al., 2011; Ramos et al., 2017) describe the design and embodiment of humanoid robots. Juarez et al. (2011) work on an ontology, called RoboDB, to describe robotic embodiments such as body segments and sensors. However, the authors have only reported a single prototype implementation so far. Ramos et al. (2017) present the Automatic Design of Robots Ontology (ADRO) that uses the upper ontologies CORA and SUMO. It provides additional definitions on structural robot parts, robot actions, robot types and relations between actions and structural parts. ADRO is part of a computational system (Ramos et al., 2018) that aims for automated robot design (see subsection 2.3.4). With the help of ADRO, possible robot types can be derived from desired robot actions. However, by focusing on robot classification and actions as well as by using a modular design approach, ADRO does not provide much detail about mechatronic subcomponents.

2.2.5. Knowledge Reasoning

In order to support problem solving, expert systems have to derive conclusions from the explicitly represented knowledge. According to Jakus et al. (2013), the most common types of reasoning are:

- **Deduction:** A specific conclusion is derived from more general evidence.
- **Induction:** General conclusions are drawn from observations of specific instances. Thus induction is the opposite of deduction.
- **Abductive reasoning:** Instead of certain conclusions, observations are explained or hypotheses are made.
- **Analogical reasoning:** Two concepts are compared in terms of a specific detail and it is concluded that they may be alike in terms of others.

The reasoning technique used by expert systems depends essentially on the knowledge representation. In the following, *deductive reasoning* for rule-based and ontology-based expert systems is described in more detail.

Rule-Based Reasoning

For rule-based expert systems, a distinction can be made between *forward chaining* and *backward chaining* when deducing new knowledge. *Forward chaining* is a data-driven mode for evaluating IF THEN rules. Available parameters are inserted in the rules and in an iterative way, each rule will be checked. In the case that the IF-part is fulfilled, the THEN-part will be executed. In contrast, *backward chaining* starts with the goal and checks which rules have to be satisfied in order to fulfill the goal. Consequently, *backward chaining* is goal-driven.

Ontological Reasoning

Considering ontologies, there are many different reasoners. Abburu (2012) presents several popular reasoners, including Pellet (Sirin et al., 2007), RacerPro (Haarslev et al., 2012) and FaCT++ (Tsarkov and Horrocks, 2006). Just as with reasoning for rule-based systems, ontological reasoning also distinguishes between *forward chaining* and *backward chaining*. When *forward chaining*, the reasoner starts with known facts and derives valid inferences from them, while when *backward chaining*, the reasoner starts from a query or particular fact and searches for all valid solutions (Abburu, 2012).

A special form of ontological reasoning is presented by Šaša Bastinos and Krisper (2013). They describe a *multi-criteria decision making method* that uses ontologies to model a decision tree. The tree, which consist of hierarchically ordered criteria to which different values can be assigned, structures the decision problem in such a way that solutions are obtained through reasoning on the ontologies.

2.3. Automated Design of Robotic and Mechatronic Systems

As discussed in subsection 2.2.1, expert systems can be used to assist users during the design process. Nowadays, however, not all knowledge-based systems that support the design process are called expert systems. The boundaries between expert systems and automated design are fluid. Therefore, this section presents not only the state-of-the-art on expert systems, but also similar systems that support mechatronic and robot design. In supporting the design process, a distinction can be made between systems that only support the conceptual phase and those that also pursue detailed design with existing components. The synthesis of robot morphologies with evolutionary algorithms is another category that, unlike the other two, does not follow traditional design approaches. Expert systems which support the user only during the selection of complete robotic systems represent a fourth category.

As a result, this section presents expert systems and similar automated systems to support the selection, conceptual design, detailed design or evolutionary design of robotic and other complex mechatronic systems. The classification of related work into these four main categories is illustrated in Figure 2.8. Since this thesis presents an expert system to support the detailed mechatronic design of humanoid robot components, the focus is on the comparison of the detailed design approaches. The section concludes by comparing the different types of knowledge representation used by the presented expert systems.

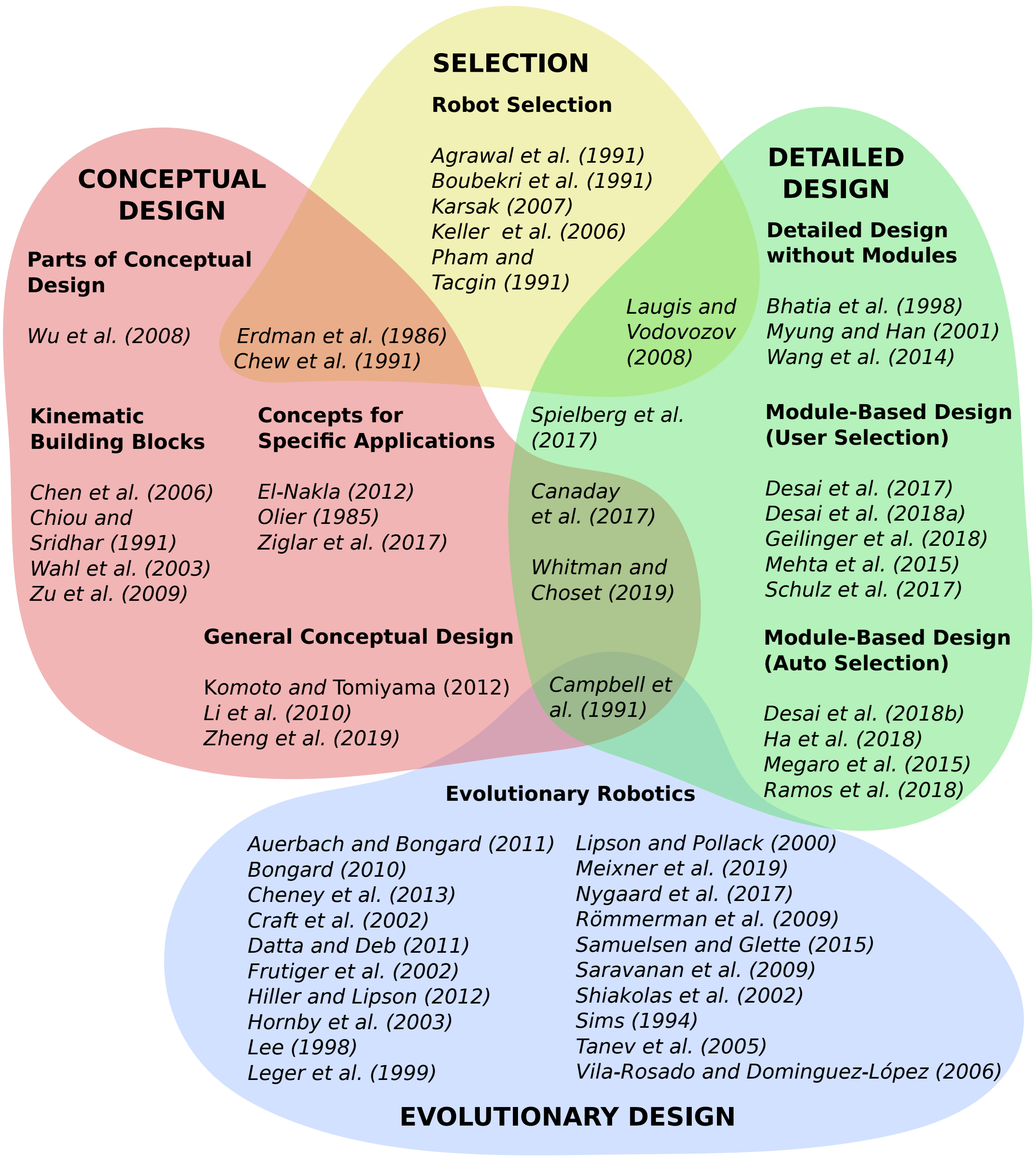


Figure 2.8.: Related work in the field of automated design of robotic and mechatronic systems. The diagram illustrates the approaches by categorizing them into four main categories: Selection, Conceptual Design, Evolutionary Design and Detailed Design.

2.3.1. Robot Selection

Robot selection can be seen as a subproblem of robot design. While in robot selection the synthesis of subcomponents to a new system is missing, in both activities the search for suitable solutions for different requirements is carried out, resulting in a multi-criteria decision problem. For more than four decades, different methods have been proposed to solve the problem of selecting an appropriate robot for a given industrial application (Koulouriotis and Ketipi, 2014).

There are various examples of expert systems that support industrial robot selection (Agrawal et al., 1991; Boubekri et al., 1991; Karsak, 2007). Based on different key attributes, such as payload, reach, repeatability and configuration, the expert systems conduct an elimination search to obtain appropriate industrial robot solutions. In a second step, Agrawal et al. (1991) and Karsak (2007) rank the remaining solutions, which fulfill the threshold, by normalizing the attributes and taking user-given weights into account.

A comparatively novel expert system by Keller et al. (2016) assists in choosing a suitable robot for human-robot interaction. After a comparison of different knowledge representations, the authors chose a decision tree. The decision tree is used to find suitable solutions from a knowledge base with collaborative robots by testing a single attribute in each node. The important attributes were determined by interviews with experts. The knowledge base comprises 21 robots from different manufacturers such as KUKA or Universal Robots.

Pham and Tacgin (1991) present a learning expert system for detailed selection of commercial robot grippers in two stages: First it guides the user in choosing the suitable gripper type. Second, specific grippers from commercial catalogs are recommended. A noteworthy feature of this expert system is the use of a Bayesian uncertainty handling technique for computing confidence values, which are used to rank the proposed grippers. If the user agrees with the displayed results, the expert system takes this as a positive reinforcement of its conclusions and modifies the confidence values accordingly.

2.3.2. Conceptual Design

Conceptual design comprises the early steps of the design process before embodiment design (Subsection 2.1.2). Based on a requirements list, principle solutions are built by searching and combining solution principles. Common representations are graphs and building blocks. Physical realizability is often evaluated on an abstract level, for example by using geometric primitives.

Chen et al. (2006), Chiou and Sridhar (1999), Wahl et al. (2003) and Zu et al. (2009) propose automated systems for the conceptual design of mechanisms through the combination of kinematic building blocks. The idea of combining building blocks is closely related to the morphological box, a heuristic problem-solving method presented in subsection 2.1.4. Kinematic building blocks represent basic mechanical elements such as gears, screw mechanisms and belt-pulleys. They have characteristics that describe (1) the types of motion that are transmitted between the input and output (e. g. rotation and translation), (2) the orientation between the axes and (3) the direction of the motion (positive or negative). To enable automated synthetic reasoning, the building blocks can be represented by motion transfer matrices (Chiou and Sridhar, 1999), motion vectors (Chen et al., 2006) or function codes (Zu et al., 2009). Starting from a desired input and output motion, a reasoning process is conducted in which the kinematic building blocks are combined and unsuitable solutions are filtered out. Chen et al. (2006) and Wahl et al. (2003) further rank the solutions according to the fulfillment of user-weighted criteria. Finally, the system displays the synthesized solutions as a list of the combined building blocks (Chen et al., 2006; Wahl et al., 2003) or a simplified 3D model based on geometric primitives (Chiou and Sridhar, 1999; Wahl et al., 2003).

Campbell et al. (1999), Komoto and Tomiyama (2012), Li et al. (2010) and Zheng et al. (2019) propose more general approaches to support the conceptual design. Starting from requirements, a hierarchical system decomposition is performed. As a result, a hierarchical model of the *functions* as well as a hierarchical model of the *functional structures* to realize these functions can be built. Li et al. (2010) represent both hierarchical models with trees that are intended to have the same structure. By comparing the matching status between the two trees, conflicting functional structures requiring further extension are iteratively identified. Komoto and Tomiyama (2012) integrate both models into a single meta-model. This meta-model is based on knowledge from a physical concept ontology and interpreted as directed hypergraph. The meta-model is used to develop a corresponding parameter network, which is also represented by a graph. The parameter network describes relevant system parameters (e. g. length, velocity, temperature) and their relations. Based on the parameter network, an engineer develops a rough geometric model based on geometric primitives. The system supports this process by evaluating consistency between the geometric model and spatial constraints defined in the meta-model. The approach allows for a concurrent development of functional and parameter-level descriptions by taking physical phenomena into account when building blocks are unavailable. One limitation of this approach is that it mainly supports system decomposition. Solutions for the physical realization of functions are not automatically found by the system. In contrast, Zheng et al. (2019) propose a procedure that includes an automated two-step selection process after system decomposition. The first step is a preselection to eliminate incompatible component combinations based on functional requirements. In the second step, the remaining combinations, consisting of various component alternatives, are ranked. For this purpose, non-functional requirements (e. g. weight, size, heat resistance) can be selected from a list of 125 entries according to user preferences. Campbell et al. (1999) introduce an agent-based conceptual design methodology that combines knowledge-based strategies with stochastic optimization. The agents are analogous to individual specialists within a company. Maker agents reason on desired user inputs and outputs using different reasoning strategies such as tree and pattern matching. Manager agents make decisions about designs and penalize the maker agents for bad design contributions. After every run, the results are evaluated and a new run is started by modifying good and Pareto optimal design solutions until the maker agents terminate the process. A case study for weighting machines demonstrates that the iterative approach is suitable for changing multi-objective problems. It uses a catalog with over 300 real electro-mechanical components, but similar to other conceptual approaches, the components' spatial dimensions are not considered.

Besides the presented general approaches and mechanism design, there is work intended to support conceptual design for specific mechatronic applications (El-Nakla, 2012; Olier, 1985; Ziglar et al., 2017). A case-based expert system to support the electronic filter design in mechatronic systems is proposed by El-Nakla (2012). It assists non-domain experts to explore areas of conflict and contradiction, e. g. a mismatch between the input of an A/D converter and the output of an electronic filter. Approaches to support conceptual robot design are presented by Olier (1985) and Ziglar et al. (2017). Olier (1985) describes the theoretical architecture of an expert system that could support the preliminary steps of space robot design. The input are mission requirements given by spacecraft engineers. Based on a rule-driven analysis, synthesis and evaluation, a set of design parameters and spatial relations among subsystems should be determined. Ziglar et al. (2017) propose an approach that aims for automatic synthesis of robots based on a set of available hardware and software components. Taking functional requirements, mission context and modularity into account, the components are automatically selected and organized into an optimal interconnection structure. The results of the synthesis are a hardware and a software graph that express connections between the components, e. g. with respect to data transmission. Case studies are a wheeled rescue robot and ESCHER, a humanoid disaster robot for the DARPA Robotics Challenge. Similar to the other conceptual approaches, the approach evaluates the physical realizability of the hardware components only on an abstract level by focusing on their interconnections.

The approaches of Erdman et al. (1986), Chew et al. (1991) and Wu et al. (2008) do not cover the whole conceptual design process, but parts of it. Wu et al. (2008) focus on the generation of product ideas. The fuzzy case-based reasoning framework aims for enhancing a given product by retrieving other scenario-compatible products. The database comprises 1600 products that are modeled by a 100-attribute vector. Attributes are described by linguistic variables in fuzzy theory. They model the use scenario as well as manufacturing/recycling features. Based on five use scenario attributes chosen by the user, a retrieve-and-filtering mechanism is conducted. The retrieving mechanism is a fuzzy case-based reasoning technique that uses the use scenario attributes for retrieving ideas. The retrieved ideas are subsequently filtered using the manufacturing/recycling features as filter criteria. Erdman et al. (1986) and Chew et al. (1991) present expert systems that focus on a single part of conceptual robot design: the selection of kinematic structures. The work of Erdman et al. (1986) searches for suitable robot gripper kinematics whereas the approach of Chew et al. (1991) evaluates kinematic structures of robot hands. Both expert systems use graphs to represent different kinematic structures.

2.3.3. Evolutionary Design

The term "automated robot design" is widely used for systems that use evolutionary methods to co-optimize the morphology and control of robots (Figure 2.9). In contrast to traditional design, evolutionary robotics use ideas and principles of biology (Pfeifer et al., 2005). Inspired by Sims (1994) early work on evolving virtual creatures, evolutionary algorithms have been used to design various robotic systems:

- Sphere-based robots (Auerbach and Bongard, 2011; Tanev et al., 2005)
- Rod-based robots (Hornby et al., 2003; Lipson and Pollack, 2000)
- Voxel-based soft robots (Cheney et al., 2013; Hiller and Lipson, 2012)
- Robot gripper configurations (Datta and Deb, 2011; Saravanan et al., 2009)
- Robot hands (Meixner et al., 2019)
- Robot arms (Bongard, 2010; Leger et al., 1999; Shiakolas et al., 2002; Vila-Rosado and Domınguez-López, 2006)

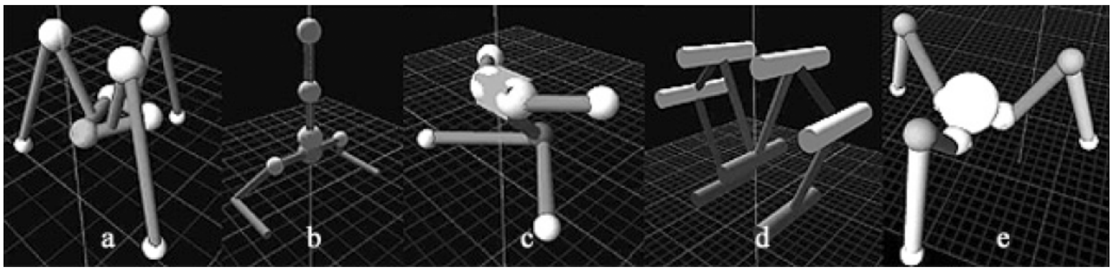


Figure 2.9.: Evolutionary design of a three-legged robot creature.
Source: (Frutiger et al., 2002) © 2002 IMechE and John Wiley & Sons, Inc.

- Legged or wheeled robot creatures (Craft et al., 2002; Frutiger et al., 2002; Lee, 1998; Nygaard et al., 2017; Römmerman et al., 2009; Samuelsen and Glette, 2015).

A classical knowledge base with expert knowledge is not required. Instead, initial robot morphologies and controllers are repeatedly modified using stochastic, metaheuristic optimization algorithms. This results in whole populations of potential solutions, which are usually optimized for a single task such as an efficient locomotion (Auerbach and Bongard, 2011; Lipson and Pollack, 2000) or successful manipulation of different objects (Bongard, 2010; Meixner et al., 2019). Most approaches test the robot morphologies only in simulation. However, some authors build up prototypes based on 3D printed structures that are actuated by pneumatic pipes (Hiller and Lipson, 2012) or servomotors (Frutiger et al., 2002; Hornby et al., 2003; Lipson and Pollack, 2000; Nygaard et al., 2017; Samuelsen and Glette, 2015). For building up the prototypes, the simulated results usually have to be modified (Frutiger et al., 2002; Hornby et al., 2003; Lipson and Pollack, 2000). Considering the performance of the results, there is a high variance: Some robots meet the simulated results whereas others do not even reach half the performance (e. g. speed). This reality gap is reported by several authors (Frutiger et al., 2002; Lipson and Pollack, 2000; Nygaard et al., 2017; Samuelsen and Glette, 2015). It can be explained by the fact that evolutionary algorithms are "blind" optimization algorithms which tend to exploit every inaccuracy of the simulation (Doncieux et al., 2015).

2.3.4. Detailed Design

In contrast to most work on conceptual and evolutionary design, the approaches summarized in the detailed design category consider existing subcomponents with their properties. Mechatronic systems are compositions of various mechanical and electronic subcomponents. A major challenge during the design

Table 2.3.: Automated detailed design of robotic and mechatronic systems.

Source	**Case Study**	**Selection**		**Arrangement**	
		Auto	**Catalog**	**Auto**	**Various**
Bhatia et al. (1998)	Robot arm	●	●	●	○
Canaday et al. (2017)	Legged robot	○	○	●	○
Desai et al. (2017)	Legged & wheeled robot	○	○	OPT	●
Desai et al. (2018a)	Electromechanical devices	○	○	●	●
Desai et al. (2018b)	Robot arm	●	○	●	●
Geilinger et al. (2018)	Legged & wheeled robot	○	○	OPT	●
Ha et al. (2018)	Legged robot & robot arm	●	○	●	●
Laugis and Vodovozov (2008)	Motor, gearbox & power converter	●	●	○	○
Megaro et al. (2015)	Legged robot	●	○	●	●
Mehta et al. (2015)	Legged & wheeled robot, robot arm	○	○	●	●
Myung and Han (2001)	Machine tool	○	●	●	○
Ramos et al. (2018)	Legged & wheeled robot, robot arm	●	○	●	○
Schulz et al. (2017)	Legged & wheeled robot	○	○	OPT	●
Spielberg et al. (2017)	Legged robot & quadcopter	●	○	●	○
Wang et al. (2014)	Spindle box	○	○	●	○
Whitman and Choset (2019)	Robot arm	○	○	●	●
Approach of this thesis	Sensor-actuator units & robotic hands	●	●	●	●

Abbreviations: OPT = User defines initial arrangement, but system can optimize it.
Symbols: ● = yes; ○ = no;

of highly integrated mechatronic systems such as humanoid robots is the selection of these subcomponents. To ensure a compact mechatronic system, the *selection* must not only take the subcomponents physical dimensions and interfaces into account, but also their *arrangement*. In this way, the resulting dimensions of the overall system can be calculated and it can be checked whether subcomponent combinations are suitable to meet the spatial requirements.

In the following, related work in the area of detailed design is presented and compared in Table 2.3 with regard to the degree of *automation* in the *selection* and *arrangement* of subcomponents. The automation of both aspects can already lead to a detailed, fully automated system that takes physical feasibility into account. However, a solution tailored to the conflicting requirements of highly-integrated components is most likely to be achieved if the system provides numerous possibilities, i. e. a large solution space. This can be achieved by not having only a few options in subcomponent *selection*, but entire subcomponent libraries based on manufacturer *catalogs*. Likewise, completely new solutions result if there are *various* possibilities to *arrange* subcomponents relative to each other instead of only a single parameterized arrangement solution.

Canaday et al. (2017) and Whitman and Choset (2019) present approaches to automate parts of the robot design process based on computational simulations. The iterative approach described by Canaday et al. (2017) starts with a fixed topological structure of a quadruped robot. Geometric and mass parameters are automatically adjusted until a target velocity is achieved and the resulting actuation parameters can be fulfilled by a given actuator. Whitman and Choset (2019) optimize the topology of an initial robot arm based on a set of given waypoints that describe the arm's path to fulfill a task. Design parameters are the link lengths, the orientation of the axes and the base locations. These parameters are involved in the inverse kinematics problem which has to be solved to follow the specified waypoints. The elimination of joints is also considered in the approach. However, the preselected HEBI robotics actuators are only taken into account in terms of their torque limits. Both approaches (Canaday et al., 2017; Whitman and Choset, 2019) are evaluated by building prototypes based on the experimental results. However, the authors report that their results are not always physically realizable either because of self-collisions (Whitman and Choset, 2019) or by not taking fabrication constraints into account (Canaday et al., 2017). Since both approaches only consider few characteristics of a single catalog component, the actuator, they are classified between conceptual and detailed design approaches in Figure 2.8.

A possibility to ensure physical realizability for a similar approach is presented by Spielberg et al. (2017) who present parametric trajectory optimization of legged robots and quadcopters. The authors model the constraints resulting from the masses and dimensions of the actuators conservatively. Their optimization approach does not only vary the lengths and masses of the robot

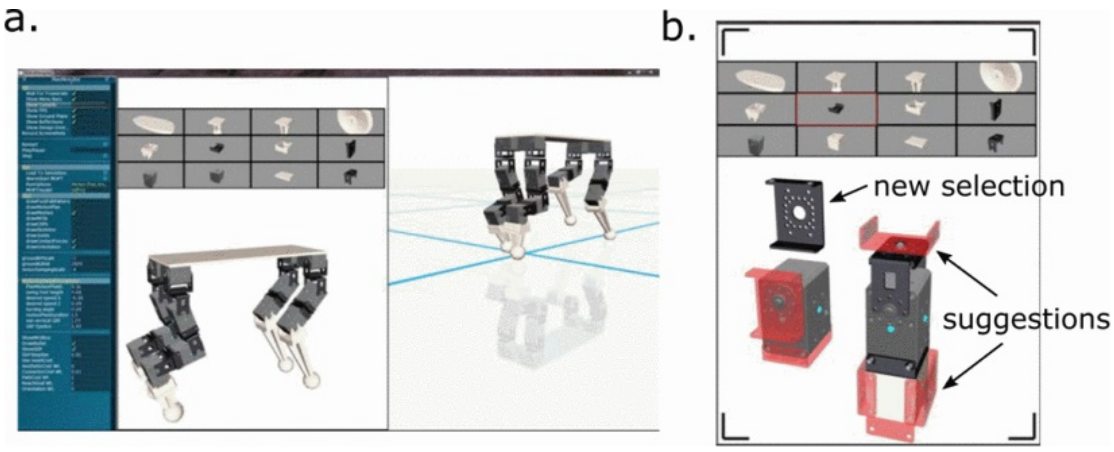


Figure 2.10.: Interactive design of robot devices via drag-and-drop (a). The system also makes suggestions about possible connections for a chosen module (b). *Source:* (Desai et al., 2017) © 2017 IEEE.

topology but also considers actuator selection based on the required torques. However, the subcomponent library comprises only three different Dynamixel actuators. Furthermore, the approach only considers a single, parameterized topology for each case study.

Modules represent another possibility to achieve physical realizability of mechatronic systems easily by taking a minimum of electromechanical interfaces into account. A popular approach for automated design is to synthesize mechatronic systems by combining few commercially available modules with parameterized 3D printable structures. Desai et al. (2017), Desai et al. (2018a), Geilinger et al. (2018), Mehta et al. (2015) and Schulz et al. (2017) present software that enables the user to design mechatronic systems very easily in an interactive way (Figure 2.10). Through a simple drag-and-drop interface, the user selects modules to build a mechatronic system and obtains a visualization on the screen. The modules can be off-the-shelf servo motors (Geilinger et al., 2018), sensor-actuator units such as Dynamixel actuators with fitting hinges (Desai et al., 2017), modified catalog components (Desai et al., 2018a), electromechanical building blocks designed by experts (Mehta et al., 2015) and in all cases simple structures or end-effectors that can be 3D printed. Even if there are only a few modules to choose from, the almost free arrangement allows a wide range of possibilities. When arranging the modules, the systems support the user in different ways: While the systems of Desai et al. (2017) and Schulz et al. (2017) provide support in finding a local optimum to enable function and manufacturing, the approach of Geilinger et al. (2018) provides a mode for automatic design optimization of segment lengths. Some approaches go further

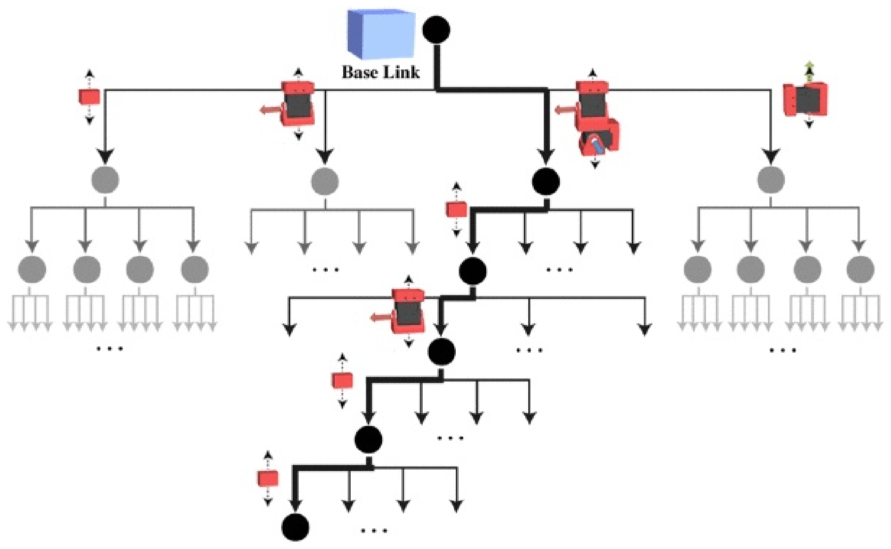


Figure 2.11.: Example for a search tree, used for design optimization.
Source: (Ha et al., 2018) © 2018 IEEE.

by automatizing the whole arrangement process. Mehta et al. (2015) arrange the user-selected modules based on predefined interfaces automatically. Desai et al. (2018a) provide an auto-arrangement mode that solves an optimization problem by placing subcomponents according to their size, starting with the largest.

Other module-based systems do not only support arrangement, but also perform automated selection (Desai et al., 2018b; Ha et al., 2018; Megaro et al., 2015; Ramos et al., 2018). Different approaches are used for automated module selection. Regarding the system of Megaro et al. (2015), the user loads an existing skeletal structure of a legged robot and defines motion targets. This is used to generate the geometry of the robot, including the selection and arrangement of the modules. Based on simulations, the user can edit the structure in an iterative process by adding, removing, repositioning or aligning motors. The systems of Desai et al. (2018b) and Ha et al. (2018) also use motion trajectories as input, but no predefined skeletal structures. Instead, an A* search is performed to find a robot that can perform the motion trajectory while being as simple as possible. As illustrated in Figure 2.11, the problem is formulated as shortest path problem, starting with the robot's base link. The nodes of the directed acyclic graph represent the solutions that are possible on the basis of a library containing modules, mostly sensor-actuator units and mechanical links. Ramos et al. (2018) introduce a system that generates robot structures and controllers based on an input set of abilities such as grasping or walking. These abilities are used to find suitable robot configurations. For example, a robot that is supposed to walk and grasp leads to the configuration of a humanoid robot. For this purpose, the system uses an ontology that relates robot configurations and abilities. The chosen robot configuration is passed to a structure generator that loads and modifies a corresponding, parameterized model. Parameters and motors are selected based on further requirements, e. g. payload and workspace.

All the presented module-based design systems have in common that they aim for an end-to-end system. As a result, they do not only focus on an easy design, but also simple manufacturing using rapid prototyping techniques. By reducing the required expert knowledge through the use of modules and an easily accessible GUI, such interactive design systems also allow novices such as students to build simple robot creatures (Desai et al., 2017; Mehta et al., 2015; Ramos et al., 2018; Schulz et al., 2017) or electromechanical devices (Desai et al., 2018a). In the evaluation of the systems, this is demonstrated by building simple, small prototypes, consisting of 3D printed structures and low-cost servo motors. However, this also means that these systems are more focused on educational and research applications rather than on industrial use. Furthermore, engineers often cannot rely on modules to design highly integrated mechatronic products that meet the requirements. Instead, they have to select and arrange catalog components as well as they have to build customized parts that take the necessary electromechanical interfaces into account. As a result, detailed design becomes more complex and interdisciplinary expert knowledge is necessary. Only a few systems support the mechatronic design process without modules in a high level of detail (Bhatia et al., 1998; Laugis and Vodovozov, 2008; Myung and Han, 2001; Wang et al., 2014). In order to deal with the complexity of the mechatronic design process, all these approaches use a comprehensive knowledge base.

The automated design optimization of a comparatively complex system, a spindle box system, is presented by Wang et al. (2014). The geometry of the spindle is highly decisive for the stiffness and the natural frequencies of the whole system. However, it cannot be optimized without taking the properties and arrangement of other subcomponents such as the hydrostatic bearings into account. To handle these dependencies, ontologies are used. They represent design knowledge by describing the spindle box system with respect to its requirements, component configuration, design parameters, linking interfaces and control loops. It should be noted that the subcomponents under consideration are neither modules nor catalog components, but customized parts based on parametric models.

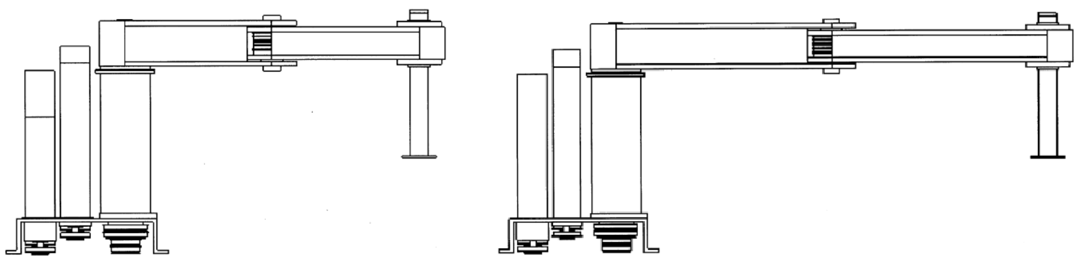


Figure 2.12.: Left: Parametric model of a SCARA, Right: Adapted for a larger reach. *Source:* (Bhatia et al., 1998) © 1998 Elsevier Science Ltd.

Bhatia et al. (1998), Laugis and Vodovozov (2008) and Myung and Han (2001) present expert systems that incorporate libraries with existing catalog components. Similar to the module-based approaches, they differ in the way they automate the design process. The expert system of Laugis and Vodovozov (2008) supports a simultaneous gear-motor-converter assemblies calculation and selection. Structured query language (SQL) sentences are used to extract information from a database that includes catalog components from different manufacturers. This results in more than 20,000 possible drive variants. However, the system is limited to the selection of catalog components and does not consider their spatial dimensions or arrangement. Conversely, the expert system of Myung and Han (2001) automates the arrangement but not the selection of catalog components. It combines a CAD model with an expert system to handle the parametric design of machine tool assemblies. With the help of the GUI, the user selects the catalog components. Based on this selection, the resulting dimensions of the assembly are calculated and the parameterized CAD model is adapted. Bhatia et al. (1998) present another expert system that interacts with a CAD package. In order to design a SCARA (Selective Compliance Assembly Robot Arm), the user specifies top-level design parameters such as workspace, reach, and payload. Based on this input, calculations are carried out which are used to decide on link lengths, motors, bearings, belt lengths and transmissions. The variables resulting from the catalog components and calculations are used to adjust a parametric SCARA model in CAD (Figure 2.12), which in turn allows a more accurate calculation of the masses and joint torques. In an iterative process, the values calculated by the expert system and the parameters of the CAD model are adjusted to refine the model. Other than the expert systems of Laugis and Vodovozov (2008) and Myung and Han (2001), the expert system of Bhatia et al. (1998) considers both, the automated selection and the arrangement of catalog components.

Summary

This subsection compared detailed design approaches with regard to the automation of subcomponent selection and arrangement (Table 2.3). Although there are a few approaches that automate both, selection and arrangement, strong simplifications are made, most notably modularization and parameterization of a single solution. Both simplifications are plausible to a certain extent and can lead to useful computational systems for automated design. However, they drastically reduce the number of possible solutions, which is crucial for the design of highly integrated mechatronic systems such as humanoid robots. Considering the described case studies in the field of robot design, the presented detailed approaches mostly focus on laboratory and educational use. There is no detailed approach that enables automated design of highly integrated mechatronic components for high-performance humanoid robots based on expert knowledge.

A comparison of related work (Figure 2.8) in the field of automated robot design shows that detailed design approaches have been increasingly pursued in recent years. It is to be expected that this field will continue to develop strongly in the future. An example for this trend is the recently started BMBF project *Q-Rock* (Roehr et al., 2019), which pursues the ambitious goal to semi-automate robot hardware/software co-development as follows: First, robots explore their own capabilities based on their existing hardware configuration in simulation. These capabilities are then clustered and semantically annotated by humans, creating a database that links capabilities to hardware/software components. Based on this database and a user-specified task description, suitable robot software and hardware components shall be proposed. To date, however, no detailed results have been published.

2.3.5. Knowledge Representation

Many of the presented approaches for automated design use expert systems (ES) or other knowledge-based (KB) systems. In this context, other knowledge-based systems are defined as systems that use a knowledge base but are not explicitly described as expert systems by their authors. Table 2.4 compares these systems with regard to the chosen techniques for knowledge representation. The comparison is restricted to common representations (subsection 2.2.3) that have been chosen at least once. Sorting the approaches by the main categories introduced in this section shows that the presented "Evolutionary Design" ap-

Table 2.4.: Knowledge representation of expert systems (ES) and other knowledge-based systems (KB).

Category	**Source**	**System Type**	**Rule-Based**	**Frames**	**Case-Based**	**Decision Tree / Matrix**	**Fuzzy Logic**	**Bayesian Uncertainty**	**Ontology**	**Miscellaneous**
Robot Selection	Agrawal et al. (1991)	ES	○	○	○	●	○	○	○	●
	Boubekri et al. (1991)	ES	●	○	○	○	○	○	○	●
	Karsak (2007)	ES	●	○	○	○	●	○	○	●
	Keller et al. (2016)	ES	○	○	○	●	○	○	○	●
	Pham and Tacgin (1991)	ES	●	●	○	○	○	●	○	○
Conceptual Design	Campbell et al. (1999)	KB	○	○	○	○	○	○	○	●
	Chen et al. (2006)	KB	●	○	○	○	○	○	○	●
	Chew et al. (1991)	ES	●	○	○	○	○	○	○	●
	Chiou and Sridhar (1999)	KB	○	○	○	○	○	○	○	●
	El-Nakla (2012)	ES	○	○	●	○	○	○	○	○
	Erdman et al. (1986)	ES	●	○	○	○	○	○	○	●
	Komoto and Tomiyama (2012)	KB	●	○	○	○	○	○	●	○
	Olier (1985)	ES	●	○	○	○	○	○	○	○
	Wu et al. (2008)	ES	○	○	●	○	●	○	○	○
	Zu et al. (2009)	KB	●	○	○	○	○	○	○	●
Detailed Design	Bhatia et al. (1998)	ES	●	○	○	○	○	○	○	○
	Laugis and Vodovozov (2008)	ES	●	○	○	○	○	○	○	●
	Myung and Han (2001)	ES	●	●	○	○	○	○	○	○
	Ramos et al. (2018)	KB	○	○	○	○	○	○	●	○
	Wang et al. (2014)	KB	○	○	○	○	○	○	●	○

Abbreviations:

ES	Expert system
KB	Other knowledge-based system

Symbols:

●	yes
○	no

proaches do not use a classical knowledge base. This can be explained by the fact that the approach of using evolutionary algorithms is contradictory to a knowledge base in which expert knowledge is stored. Nevertheless, Campbell et al. (1999) demonstrate that a hybrid system can combine both approaches.

With regard to the frequencies of the individual representation techniques, it can be stated that rule-based knowledge representation is by far the most frequently used: 12 of the 20 knowledge-based systems presented (60%) use rules to represent design knowledge. This corresponds approximately to the same proportion (63%) that Wagner (2017) determined in his analysis on knowledge representation techniques used for expert systems. Since rules are well suited to represent procedural knowledge, but ineffective to describe numerous solutions and components, rules are often combined with other knowledge representation techniques that can express concepts such as frames or ontologies. Considering these other knowledge representation schemes, a large variance can be observed. Furthermore, many authors do no not use classical knowledge representation techniques and introduce new schemes instead. They are summarized under the term "Miscellaneous" and represent knowledge on components or partial solutions through matrices (Chiou and Sridhar, 1999; Erdman et al., 1986), vectors (Chen et al., 2006), codes (Agrawal et al., 1991; Boubekri et al., 1991; Karsak, 2007; Zu et al., 2009), SQL databases (Laugis and Vodovozov, 2008) and other schemes with component properties (Campbell et al., 1999; Chen et al., 2006; Keller et al., 2016). Graphs are also named by different authors as chosen knowledge representation technique (Erdman et al., 1986; Chew et al., 1991). However, it is difficult to distinguish clearly between graphs and other knowledge representation schemes when categorizing, since graphs can be used to visualize most other schemes such as ontologies or rules.

Wagner (2017), who in his study also examined the temporal development of knowledge representations with regard to their frequency, notices that ontologies have become increasingly popular in recent years. This trend can also be observed when looking at the publications presented in Table 2.4: Three of the five knowledge-based systems that have been published since 2012, use ontologies to represent design knowledge.

2.4. Sensor-Actuator Units for Robot Joints

This section motivates the design of sensor-actuator units and presents related work to the SAC units, which were developed as part of this thesis. An earlier version on related work was previously published in Rader et al. (2017).

2.4.1. Motivation and Requirements

Sensor-actuator (SA) units are modules that include a motor as well as one or several sensors. Other typical mechanical components are gearboxes and bearings. SA units, which also integrate electronics for motor control and communication, are referred to as sensor-actuator-controller (SAC) units.

The main motivation for the development of SA units is to simplify the design of high-performance robotic systems by using easy-to-use components. Ideally, a SA unit is placed in each robot joint and just by adding simple mechanical structures as well as a minimum of cable connections between the SA units, a robotic system is created. In order to meet the different space and performance requirements of robot joints, SA units are therefore usually developed in different sizes. The capabilities of the resulting robotic system depend largely on the components integrated in the SA units. To realize as many capabilities as possible, a high degree of integration is necessary. Besides modularity, scalability and a high degree of integration, robustness and reliability are key requirements for SA units.

A distinction can be made between rotary and linear SA units. Rotary SA units can usually be integrated directly into the axis of the robot joint, which makes them a popular choice for robot arms. Linear SA units, on the other hand, facilitate the integration of spring compliance and are often used in robot legs. An example of such a linear Series Elastic Actuator (SEA) is the RRLab SEA (Schütz et al., 2016), which is used for the Compliant Robot Leg CARL (Schütz et al., 2017).

2.4.2. Related Work

Due to their increased popularity in recent years, there are numerous sensor-actuator units that have been developed by research facilities and companies. Therefore, the following related work is limited to a choice of compact modular rotary sensor-actuator units based on an electric motor, which are built for use in human-centered robotics applications such as humanoid and service robots.

An early work on rotary SA units as they are used today was published by Albu-Schäffer et al. (2007). The paper introduces the lightweight arm LWR III, whose SA units consist of the following components: a brushless DC motor, a brake, a Harmonic Drive, position sensors on the motor and the output, a torque sensor and an electronic stack for control and power supply. The LWR III includes 7 SA units, which are linked by a carbon fibre hollow structure. Its design serves as the basis for the commercially available KUKA LBR arms. But also other purchasable robot arms designed for collaborative work with humans are based on the principle of connecting SA units through various hollow structures, including the popular arms of Franka Emika (Panda), Universal Robots (UR3, UR5, UR10) and Kinova (JACO, Gen2, Gen3).

Asfour et al. (2013) presented an early work on highly integrated, rotary SA units for humanoid robots. The SA of the humanoid robot ARMAR-4 include a brushless DC motor, a Harmonic Drive, an incremental encoder, an absolute encoder and a strain-gauge-based torque sensor. Particularly noteworthy are the compact rotary SA units of the hip and legs: By applying the strain gauges to a spoke wheel on the output and placing the drive bearings under the motor, these SA units are very short (84 mm) for a maximum peak torque of 157 Nm. As a result, they allow for an anthropomorphic appearance of ARMAR-4.

The DARPA Robotics Challenge (DARPA, 2015) demonstrated that nowadays not only companies producing robot arms, but also many research institutions developing humanoid robots are taking advantage of highly integrated, modular SA units. Beside robots based on self-developed sensor-actuator units (Negrello et al. (2015); Stentz et al. (2015); Radford et al. (2015)), at least seven teams used commercially available Dynamixel units (Robotis, 2015). The Dynamixel Pro series offers sensor-actuator units in different sizes, which all include a motor and gear box as well as sensors (incremental and absolute position encoder), a controller and a network module. Furthermore, the units include connectors and flanges for an easy electrical and mechanical integration in robots of a wide variety of physical shapes. This encapsulation and the degree of integration is high compared to many other commercially available SA units (Schunk, 2011; TQ-Systems, 2016; Harmonic Drive AG, 2016; SENSODRIVE, 2020). However, in recent years, the need for easy-to-use SA units has led to more developments with a high degree of integration of which some have been commercialized. The ANYdrive joint (ANYbotics, 2017) and the R-Series actuators of HEBI Robotics (HEBI Robotics, 2019) offer even more functionalities. Com-

Table 2.5.: Comparison of integrated components and functionalities in state-of-the-art sensor-actuator (SA) units and sensor-actuator-controller (SAC). The comparison is limited to rotary SA/SAC units for human-centered robotics.

	Motor + Gear Unit	**Brake**	**Absolute Encoder**	**Torque Sensing**	**IMU**	**Slip Ring**	**Integrated Controller**	**High-Speed Bus**
ANYdrive Joint Unit[1] (ANYbotics, 2017)	●	○	●	●	○	○	●	●
CanisDrive (Harmonic Drive AG, 2016)	●	(●)	(●)	○	○	○	○	○
CMU NREC Drive Joint (Stentz et al., 2015)	●	●	●	●	○	●	○	○
DLR Joint Unit (Albu-Schäffer et al., 2007)	●	●	●	●	○	○	D	●
Dynamixel Pro Series (Robotis, 2015)	●	○	●	C	○	○	●	○
Gear Motor RD-HD (TQ-Systems, 2016)	●	(●)	●	○	○	○	○	○
IIT CENTAURO SEA[2] (Baccelliere et al., 2017)	●	○	●	●	○	○	D	●
IIT WALK-MAN SEA (Negrello et al., 2015)	●	○	●	●	○	○	D	●
K-Series Actuators[1] (Kinova Robotics, 2015)	●	○	●	●	A	●	●	○
KIT ARMAR-4 SA Unit (Asfour et al., 2013)	●	○	●	●	○	○	○	○
RoboSimian Actuator (Radford et al., 2015)	●	●	●	○	○	○	D	●
SENSO-Joint (SENSODRIVE, 2020)	●	(●)	(●)	●	○	○	○	○
Schunk PRL Actuator (Schunk, 2011)	●	●	●	○	○	○	●	○
X-Series Actuator[1] (HEBI Robotics, 2019)	●	○	●	●	●	○	●	●
KIT Sensor-Actuator-Controller Unit	●	○	●	●	●	●	●	●

Symbols: ● = fully integrated; (●) = optional; ○ = not integrated/placed outside
A = No IMU, but accelerometers
C = Torque sensing based on current sensing
D = Controller is directly attached to the unit, but not encapsulated
[1] SA unit series with a maximum peak torque of less than 50 Nm
[2] Similar units were made commercially available by IIT (TREE ACTUATORS)

pared to the Dynamixel series, they allow for precise torque control as they are not based on current control which needs a complex friction model to be reliable. Furthermore, they use EtherCAT, an Ethernet-based fieldbus system, for high-speed control and communication that is suitable for real-time computing.

In addition, the HEBI-Robotics actuators contain an inertial measurement unit (IMU), that is composed of a 3-axis accelerometer/gyroscope. A major disadvantage of the HEBI Robotics actuators and the ANYdrive robot joint, however, is that the maximum peak torque is 38 Nm and 40 Nm, respectively. Therefore these SA units are only very limited usable for the design of high-performance humanoid robots.

For safe interaction with humans and the environment, compliance is of utmost importance (Vanderborght et al., 2013). Therefore, in recent years, several Series Elastic Actuators (SEA) have been developed for humanoid robots such as WALK-MAN (Negrello et al., 2015), CHIMP (Stentz et al., 2015), iCub (Tsagarakis et al., 2009), COMAN (Vo-Gia et al., 2014), Robonaut 2 (Diftler et al., 2011), Valkyrie (Radford et al., 2015) and CENTAURO (Baccelliere et al., 2017). SEA include spring components for passive compliance. With regard to safety, passive compliance has the advantage of inherent reliability since it is realized in hardware. However, compliance parameters of most SEA are fixed and potentially not appropriate for a given interaction task. Furthermore, such passive compliance significantly increases the complexity of control.

Another possibility for realizing compliance is active compliance, where compliance parameters are freely adaptable during operation. Based on accurate and fast torque control, the motors are controlled in a way that emulates naturally compliant behavior. Common techniques for torque sensing in torque control loops are either current sensing (Robotis, 2015) or the measurement of small mechanical deformations in the actuator's output. This deflection can be measured by strain gauges (Albu-Schäffer et al., 2007; Asfour et al., 2013; Vo-Gia et al., 2014; Englsberger et al., 2014), two highly accurate position encoders (Negrello et al., 2015; Stentz et al., 2015; Diftler et al., 2011; Radford et al., 2015) or a combination of both (Baccelliere et al., 2017). For a precise and reliable realization of active compliance, a high control bandwidth is necessary. Therefore, a fast communication bus such as EtherCAT is needed.

Table 2.5 compares rotary sensor-actuator units from related work in terms of their integrated components and functionality. The last entry refers to the KIT sensor-actuator-controller (SAC) units, which differ from related work by a comparatively high degree of integration, in particular compared to units which allow peak torques of more than 50 Nm. Their design is described in more detail in subsection 5.2.1.

2.5. Summary and Review

This chapter presented fundamentals and related work. The fundamentals addressed systematic design and expert systems. Guidelines such as classical product development methodologies or NASA Systems Engineering offer procedural knowledge for the systematic design of technical systems. However, such procedures cannot be applied without the necessary domain knowledge (Pahl et al., 2007). Answers to the question of how domain knowledge can be made usable were provided by the fundamentals of the AI research fields of knowledge representation and expert systems.

The related work presented approaches aiming to automate the design of robotic and mechatronic systems. It focused on detailed design approaches by comparing them with regard to the automation of subcomponent selection and arrangement. Although there are a few approaches that automate both, selection and arrangement, strong simplifications are made, most notably modularization and parameterization of a single solution. Both simplifications are plausible to a certain extent and can lead to useful computational systems for automated design. However, they drastically reduce the number of possible solutions, which is crucial for the design of highly integrated mechatronic systems such as humanoid robots. Considering the described case studies in the field of robot design, the presented detailed approaches mostly focus on laboratory and educational use. The same applies to evolutionary design approaches that generate robot morphologies but do not consider mechatronic subcomponents in detail. In summary, it can be stated that there is no detailed approach that enables automated design of highly integrated mechatronic components for real-sized humanoid robots. Also, no systematic procedure is described showing how existing expert knowledge can be used to create such a computational system.

Finally, related works on rotary SA/SAC units for robot joints were presented and compared in terms of their degree of integration. This demonstrated the comparatively high degree of integration of the KIT SAC units, which were developed as part of this thesis and which serve as a case study (subsection 5.2.1).

3. Design Knowledge Acquisition and Formalization

This chapter presents the first two steps of the approach of this thesis: (1) the acquisition of *design knowledge* and (2) the *formalization* process. The goal of these steps is to acquire and formalize design knowledge so that it can serve as the knowledge base of an expert system.

The approach starts with the acquisition of *design knowledge*. As illustrated in Figure 3.1, a *robot developer* can gain design knowledge both by designing *humanoid robots* and by analyzing *related work*. Based on the acquired expert knowledge, the *formalization* process is carried out: First, *specific models* are created for each robot component that serves as a knowledge source. From these *specific models*, a *generalized model* of a robot component type is then induced. Finally, this *generalized model* as well as component data from manufacturer *catalogs* is represented by *ontologies* and *rules*. These *ontologies* and *rules* serve as the *knowledge base* of an expert system.

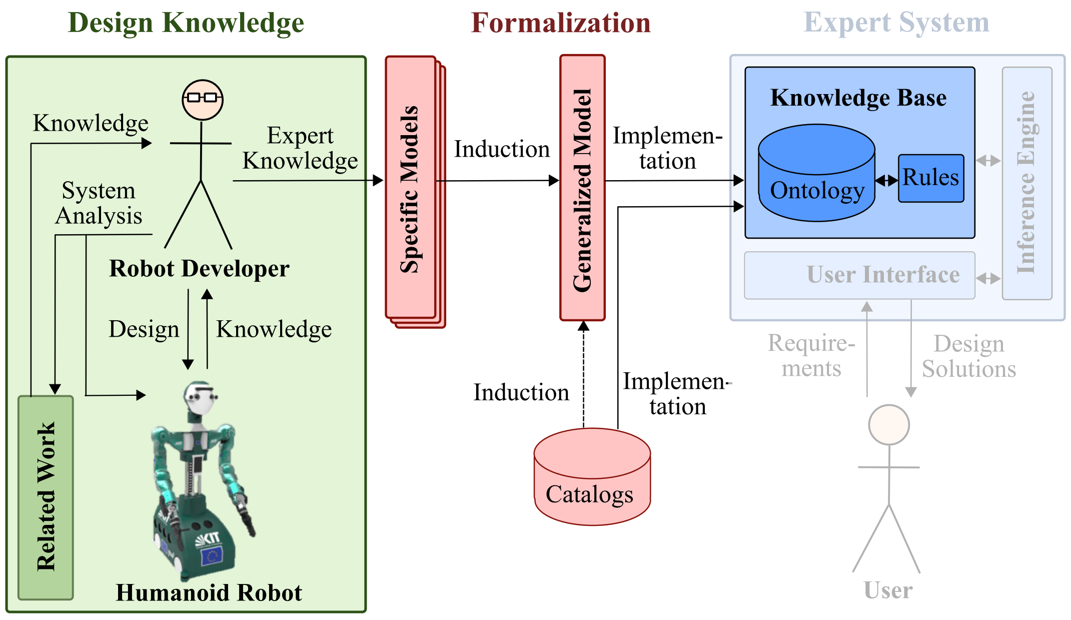


Figure 3.1.: Acquisition (green) and formalization (red) of expert design knowledge.

In the following, the design knowledge acquisition and the steps of the formalization process are described in detail. Preliminary results on the formalization process were first published in (Karrenbauer, Rader and Asfour, 2018).

3.1. Design Knowledge Acquisition

The acquisition of design knowledge represents the first step of the approach presented in this thesis. This knowledge is gained through *robot design* and *related work*. After discussing these two possibilities, a procedure for *system analysis* of robot components is described. This procedure takes into account what knowledge is required for the upcoming formalization process.

3.1.1. Knowledge Gained through Robot Design

Since the presented approach serves to support the development of complex, highly integrated mechatronic systems, detailed expert knowledge is necessary. Concepts or pure simulations are not sufficient as a source of knowledge. Gaps in knowledge or incorrect knowledge can result in fatal misconstructions. To avoid this, the successful development process of one or more humanoid robot components of the same type should serve as the basis for the formalization process and ultimately the expert system. Furthermore, the robot developer providing the expert knowledge should have been heavily involved in this development process. This provides the following advantages, among others:

- The physical feasibility of the design is proven. All parts fit together and can move as intended. For example, there are no unforeseen collisions between rotating and non-rotating parts of the mechatronic system.
- The assembly and disassembly for maintenance purposes can be ensured.
- Installation, functionality, robustness and cooling of the electronics can be verified. This includes electronic boards with all their components as well as connectors and cabling.
- The theoretically calculated properties of the robot component can be compared to measurements. If there should be larger deviations, the formulas can be adjusted to improve the model of the robot component.
- Deviations between the actual catalog components and manufacturer's specifications can be taken into account in formulas.
- Sources of friction and abrasion can be identified and eliminated at best.

- Unpredictable load cases can be detected by evaluating the robot component, in particular as part of a humanoid robot. This knowledge can be used to improve the system and future developments.

The advantages listed relate to typical problems that can arise when designing robot components. The smallest details can make the difference whether a robot component fulfills its function or not. Many of the problems can be avoided in advance through exact models and a careful development process that considers expert knowledge of all mechatronic disciplines. But only assembly and integration as well as practical tests can guarantee that the robot component works. And only then is the design knowledge of the robot component a good basis for future developments.

Although the knowledge gained by robot design should be the main source of knowledge, this does not mean that it would not be useful to derive additional knowledge from related work.

3.1.2. Knowledge Gained through Related Work

Since the development of new robot components is usually inspired by related work, some knowledge from related work may already have been considered. In addition, well-documented knowledge of related work can be used to extend the existing knowledge of designing a robot component type. In most cases it is worth considering alternative solution principles and information on other manufacturers of suitable catalog components. It is particularly valuable if the robot components are physically available and can be inspected and tested, but detailed CAD models and PCB layouts also prove to be helpful. The more detailed the related work can be analyzed, the more knowledge can be used.

3.1.3. System Analysis

The system analysis can be understood as a reverse engineering process, which closes the missing knowledge gaps. If not already available, the following knowledge about specific robot components should be gained in order to use it for the formalization process:

1. **Robot component functions and properties:** The relevant functions and properties of the robot component must be determined. The functions are usually known, but should be evaluated. Considering the properties, this might not always be the case. In order to obtain the data, measurements

are ideally carried out on the real, physically realized system. This can be supplemented by calculations and the use of very exact models such as the final CAD model. For robotic components, typical properties are: spatial dimensions, mechanical performance data, weight and costs.

2. **Robot component interfaces:** The interfaces between the system and its environment must be specifically defined. This is crucial because interface information determines whether a system can be used for a future project without any changes, only with an adapter, or not at all. For mechatronic systems, these are mainly mechanical interfaces (such as screw connections), electrical interfaces (electrical plugs with information on electrical power data) and communication interfaces (e.g. bus systems).

3. **Subcomponents of the robot component:** The relevant subcomponents of the robot component must be determined. These may be specific catalog components, but also customized parts that have been specially designed for the system, such as connecting structural parts, covers or PCBs.

4. **Subcomponent functions and properties:** The relevant functions and properties of the subcomponents can be determined by consulting manufacturer catalogs and other documentation. In case of gaps, the same procedure as for determining the properties of the overall system must be followed.

5. **Relations between robot component and subcomponents:** The properties and functions of the overall system, the robot component, result from the combination of the subcomponents. Therefore, this step examines for each property and function which subcomponents it depends on. While the integration of a single subcomponent can be sufficient for the boolean fulfillment of a function, properties of the robot component usually have to be calculated from properties of several subcomponents. For example, the length of the robot component results from the spatial dimensions of the subcomponents. Also in this step, models, especially CAD models, are an important source of knowledge.

6. **Relations between different subcomponents:** Finally, the dependencies between the subcomponents are investigated. In order to understand why the robot component is designed the way it is, the interaction of subcomponents should be examined with regard to functions and properties as well as interfaces and compatibilities. Of particular importance is the arrangement of subcomponents to each other. Whether a subcomponent

can be placed in, around, or only beside another subcomponent can be decisive for its selection.

3.2. Development of Specific Models

The development of specific models is the first part of the formalization process. Based on the knowledge gained through robot design and system analysis, a specific model is created for each specific humanoid robot component, which serves as a source of knowledge. Its development starts with modeling the overall system as a composition of its subcomponents. This work distinguishes between two types of subcomponents: *Catalog components* and *customized parts*.

$$Subcomponents = \{\{Catalog\ Components\} \cup \{Customized\ Parts\}\}$$

Catalog components are already existing components, which can either be purchased from manufacturers or have been used for earlier developments. Their properties are given (usually by catalog data) and cannot be changed. With regard to the overall system, they are regarded as the smallest part that cannot be further subdivided. Examples of catalog components are motors, gearboxes, bearings and most sensors.

Customized parts, on the other hand, are subcomponents that are created specifically for a new design. Their properties can be changed and they are designed depending on catalog components and other customized parts. Examples of typical customized parts are structural parts, covers, cables and PCBs. Of particular importance for this approach are structural parts that mechanically connect a set of subcomponents. A typical example of a structural part is a motor housing. It has dependencies on the dimensions and interfaces of catalog components such as the motor, gearbox and bearings, but also on other customized parts such as the motor shaft, covers or cable connections.

To model the specific robot components, SysML block definition diagrams (BDD) are used (subsection 2.1.3). SysML BDD allow to represent a robot component and its subcomponents as blocks, which optionally include properties (*value*s) and functions (*operations*). However, in this formalization process, the BDD is primarily used to express *relationships* between a robot component and its subcomponents. The BDD in Figure 3.2 illustrates a *Specific Robot Component* as composition (lines with diamond) of specific *Catalog Components (A,B,C,...,N)*

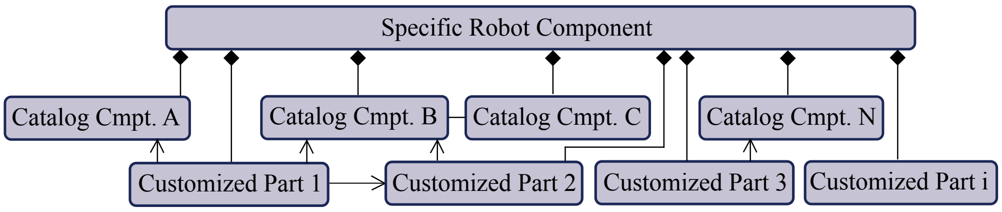


Figure 3.2.: SysML block definition diagram (BDD): Specific robot component.

and *Customized Parts (1,2,3,...,i)*. In addition to illustrating the subcomponents, the BDD also serves to express relationships between individual subcomponents. This is illustrated by lines, optionally with an open arrowhead to express an unidirectional access from the client (no arrowhead) to the supplier (arrowhead). The definition of these relationships is an important preliminary task for setting up the hierarchical structure in the next step.

It should be noted that the BDD only represents a part of each specific model describing a robot component. The specific models include further data collected during the design process and the system analysis, in particular the subcomponent properties and the calculation of the robot component properties as a function of these specific subcomponent properties.

3.3. Development of a Generalized Model

So far, only knowledge on the design of a few, specific robot components has been modeled. By identifying similarities and differences, a generalized model is induced from the specific models. The goal is to use the generalized model to represent knowledge about numerous specific robot components of the same robot component type.

The development of the generalized model can be divided into three steps. In the first step, the robot component type is *modeled hierarchically*. The overall system is broken down into subsystems up to placeholders for catalog components. In the second step, the model is extended by different *solution principle variants* for conceptual design parameters. This includes different subcomponent arrangements, kinematics and possibilities to realize functions. In the third step, it is explained how *requirements* can be considered in the generalized model. The individual steps are explained in more detail in the following.

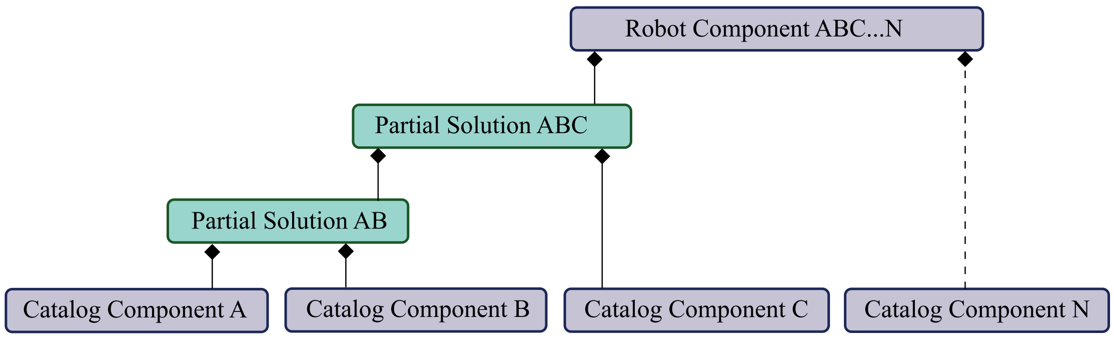


Figure 3.3.: SysML block definition diagram (BDD): System hierarchy tree.

3.3.1. System Hierarchy Tree

Trees are a frequently used form for representing hierarchical relationships in the product development process (section 2.1). In particular, in the conceptual design phase and corresponding processes of systems engineering (technical requirement definition and logical decomposition) hierarchical trees are used for function trees and functional structure trees (subsection 2.3.2). The necessity of a hierarchical representation can be explained by the typical procedure of systematic product development: A top-down decomposition of requirements and functions is followed by a bottom-up solution definition and implementation.

Based on the specific models, a tree is created, which describes the general hierarchical structure of a robot component type: The leaves of the tree are the *catalog components*, the root is the *robot component* to be designed, and all other nodes are *partial solutions* of different hierarchy levels. If the tree is traversed bottom-up, it describes the design of the robot component: The design starts with existing catalog components, which are not further subdivided and thus represent the lowest hierarchy level, the leaves. Catalog components are combined to build partial solutions. These partial solutions are combined with other catalog components or partial solutions to build partial solutions at a higher level of the robot component. This is continued until the robot component to be designed, the root, is reached.

For example, the procedure for the robot component type *sensor-actuator unit* is as follows: Catalog components of type *motor* and *gearbox* are combined to form a partial solution, the *motor-gearbox combination*. This partial solution is further combined with catalog components of the *bearing* type, resulting in a higher-level partial solution, the *drive section*. This procedure is continued until solutions for the overall system, *sensor-actuator units*, are formed.

The tree nodes can also be regarded as states, while the edges are the transitions from one state to another. These transitions are described by rules and formulas. In contrast to the specific model, specific catalog components are replaced by catalog component types. Consequently, concrete values of specific catalog components are also replaced by parameters in the formulas and rules. When setting up rules and formulas, it is necessary to consider how exactly catalog component properties and interfaces are specified by manufacturers. Therefore it is important to *induce* general knowledge from *catalogs* as well (Figure 3.1).

As with the specific models, a SysML BDD is used to represent the generalized model of the system (Figure 3.3). It describes the robot component hierarchically by using *blocks* for nodes and *composite associations* for edges. The BDD for the specific robot components (Figure 3.2) serve as the starting point. Their *relationships* are particularly relevant for deciding how the tree describing the robot component is structured. For example, they help to identify when catalog components have to be combined with other catalog components in order to have all relevant information for selecting and generating partial solutions when bottom-up traversing the tree. Another difference compared to the specific models is that *customized parts* are no longer represented by separate blocks. Since they are designed depending on catalog components and cannot be treated separately, they are defined when partial solutions are built.

3.3.2. Solution Principle Variants

The crucial difference between a specific and the generalized model is that the generalized model describes millions, if not billions, of different variants of the robot component instead of a single one. Replacing specific catalog components with catalog component types is one measure to describe more possibilities for building a robot component type. However, from a conceptual point of view, the result is only a single, parameterized solution. When developing highly integrated robot components that have to meet a wide variety of requirements in a very confined construction space, it is worth considering different concept variants and evaluating them against each other. As described in subsection 2.1.2, such *concept variants* are combinations of *solution principles* that reflect selected characteristics for the relevant conceptual design *parameters* of the system.

For the conceptual design of robot components, this work introduces three groups into which these *parameters* and their corresponding *solution principles* can be divided: structural, kinematic and functional parameters (Table 3.1).

Table 3.1.: Parameter groups for the conceptual design of robot components.

Parameter group	Solution principles	Description	Example
Functional parameter	2^n	Use of component types (or groups of catalog components) to meet functional requirements	Possibilities for torque sensing
Structural parameter	n	Arrangement of catalog components relative to each other	Bearing arrangement relative to motor
Kinematic parameter	n	Kinematic configuration and coupling of degrees of freedom	Number of finger joints

$n \in \mathbb{N} \setminus \{0\}$ realization possibilities

Unlike the other two parameter groups, the number of *solution principles* for a *functional parameter* does not correspond to the *realization possibilities* n, but 2^n (Table 3.1). In the simplest case, this can be explained by the fact that a catalog component type (or a group of catalog components) is either used to fulfill a certain function or not $\{0, 1\}$. If there exist n independent realization possibilities (i. e. binary features), the number of solution principles results from the Cartesian product $|\{0, 1\}^n| = 2^n$. As a result, this definition allows to represent *optional functions*, i. e. functions which do not have to be fulfilled at all. Furthermore, it can represent *redundancy* by solution principles that describe the fulfillment of a function by multiple independent realization possibilities.

The procedure how parameters and solution principles are defined and processed is based on the procedure of the *morphological analysis* (subsection 2.1.4): By determining the relevant parameters for the problem and possible characteristics for each parameter (solution principles), a matrix can be built up, the morphological box. Concept variants are built by selecting exactly one solution principle for each parameter. In order to consider each concept variant contained in the morphological box, all possible combinations of solution principles must be considered. However, to prevent the emergence of too many concept variants, only the most relevant parameters and only the solution principles that differ significantly are taken into account. The use of morphological boxes adds a heuristic method to the formalization process and at the same time allows the classification of existing robot components.

With regard to the SysML BDD for the generalized model, concept variants of the morphological box are represented by inheritance (generalization) between the parameter (supertype) and several solution principles (subtypes). In Fig-

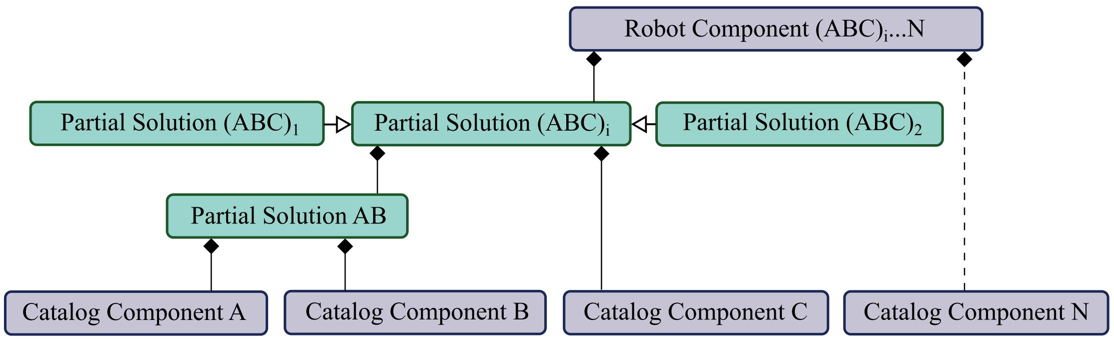


Figure 3.4.: Expanded SysML BDD: System hierarchy with solution principle variants.

ure 3.4, *Partial Solution* $(ABC)_i$ is the parameter, while *Partial Solution* $(ABC)_1$ and *Partial Solution* $(ABC)_2$ are possible solution principles for this parameter. In this case, for instance, the parameter $(ABC)_i$ could define how the drive bearings are arranged (*structural parameter*). Solution principle $(ABC)_1$ could describe an arrangement of the bearings beside the motor, while solution principle $(ABC)_2$ could describe an arrangement below the motor's rotor.

Solution principles, in particular for *functional parameters*, can be represented similarly to catalog components by leaves of the hierarchical structure tree. However, if their effect on the selection and arrangement of catalog components is large (e.g. for *structural parameters*), the strict tree structure may turn out to be unsuitable and has to be abandoned. These solution principles are then no longer represented by a single block, but by several blocks with relationships between them. As a result, the subsequent transformation of the BDD into a graph will result in a directed acyclic graph (DAG), which in most cases will no longer be a tree. In the DAG, different solution principles with significant influence on selection and arrangement of catalog components are represented by different paths.

With regard to the rules and formulas that describe the transition from partial solutions or catalog components to higher-level partial solutions, different solution principles are taken into account by making case distinctions. Thus, the rules and formulas are formulated and evaluated depending on the chosen solution principles.

3.3.3. Requirements

So far, the generalized model describes various possibilities for designing a robot component type. In order to use this knowledge to generate a suitable solution for future designs, the properties of the robot component must meet technical requirements. NASA Systems Engineering (Kapurch, 2010) defines requirement types and describes the processes by which they are decomposed, allocated and validated (subsection 2.1.3). In the following, it serves as a basis for integrating requirements into the generalized model of the robot component.

Types of Requirements

Kapurch (2010) identifies technical requirements, performance requirements and interface requirements as the most important types of requirements (subsection 2.1.3). Table 3.2 shows typical examples for these three types of requirements, that have to be met by mechatronic robot components.

Table 3.2.: Types of requirements and examples for mechatronic robot components.

Functional requirement	Examples
Mechanical	Provides torque, has brake, continuous rotation
Sensing	Position/torque sensing capability
Data processing	Has PCB for internal data processing
Performance requirement	Examples
Spatial dimensions	Max. length, max. diameter, max. height
Mass	Max. mass (“weight”)
Costs	Max. costs
Mechanical	Min. peak torque/force, min. (angular) speed
Sensing	Min. sensor resolution, min. absolute accuracy
Data processing	Min. bit rate
Interface requirement	Examples
Mechanical	Specific screw flange (ISO flange)
Electrical	Power supply (supply voltage, max. current)
Communication	Bus system, serial ports

Functional requirements, which define what functions must be accomplished, are largely determined by the type of robot component selected. For example, a sensor-actuator unit is expected to *provide torque*. However, other functional requirements such as a braking function (*has brake*) or a *torque sensing capability* may or may not be necessary depending on the application. With regard to the fulfillment of functional requirements, there are only two possibilities: either they are fulfilled or not.

Performance requirements, on the other hand, define how well a system must perform functions. In order not to be too restrictive, Kapurch (2010) recommends to distinguish between the threshold value (i.e. the minimum acceptable value) and the desired baseline. Inspired by this, performance requirements are defined in the following not only by a single value, but by a desired value with a permitted deviation. Thus, different design solutions that meet all functional requirements can also be compared quantitatively. Typical performance requirements for physical components are *spatial dimensions*, *mass ("weight")* and *costs*. With regard to mechatronic robot components, *max. torque/force* and *max. (angular) speed* are typical *mechanical* performance requirements. Further typical performance requirements arise in the areas of *sensing* and *data processing*. The similarity of the categories of functional and performance requirements in Table 3.2 is explained by the fact that theoretically there is a performance requirement for each functional requirement. For example, *min. sensor resolution* and *min. absolute accuracy* are the corresponding performance requirements for the functional requirement *torque sensing capability*.

Interface requirements define interfaces between the robot component and its environment. Mechanically, these are usually fastening options such as *specific screw flanges*. A typical electrical interface requirement is the *power supply*, whereby not only the *supply voltage* but also the *maximum current* must be taken into account. In order to combine different robot components with each other, the *communication interfaces* for *bus systems* or *serial ports* are also crucial.

Transformation into a Directed Acyclic Graph (DAG)

As described in subsection 3.3.1, the classic design process consists of a top-down approach to allocate and decompose the requirements, followed by a bottom-up approach with regard to the synthesis of design solutions. So far, only the bottom-up synthesis of the robot component is reflected by the generalized model, in particular by the structure of the SysML BDD (Figure 3.4).

To consider requirements in the generalized model, first the SysML BDD (Figure 3.4) is transformed into a directed acyclic graph (DAG) $G = (V, E)$ comprising a set of nodes (or vertices) V and a set of edges E (Figure 3.5). Therefore, all blocks of the SysML BDD are transformed into nodes v and all relationships between the blocks (composite associations or generalizations) are transformed into edges e. The direction between the nodes v is chosen so that the robot component is built bottom-up, from catalog components (sources) to higher-level partial solutions up to the overall system, the robot component (sink). It should

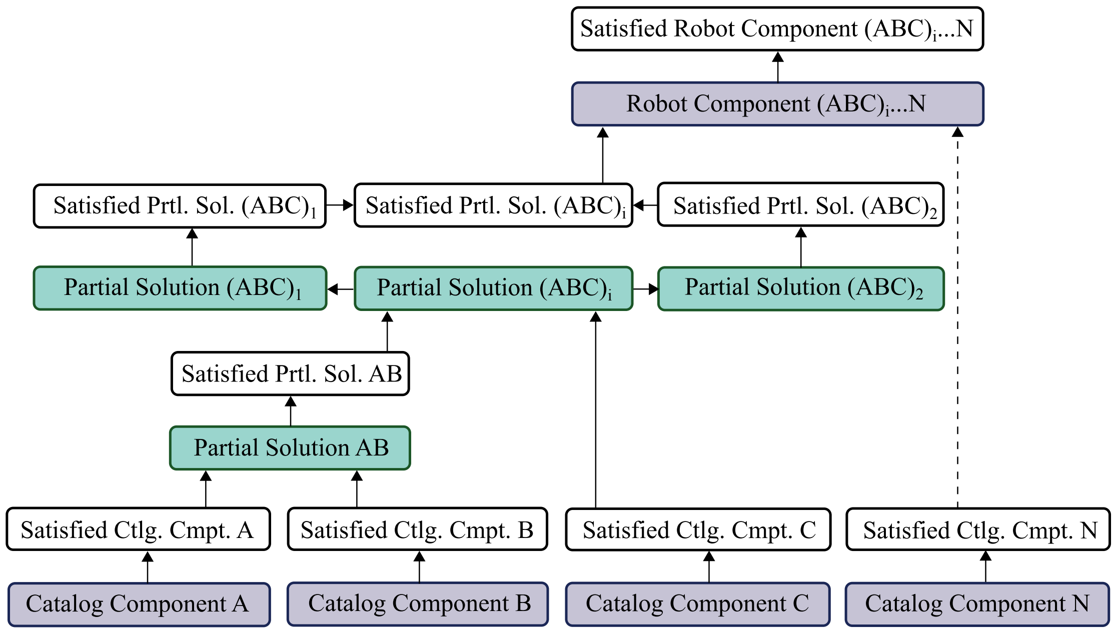


Figure 3.5.: DAG with requirement checks based on expanded SysML BDD.
It is later referred to as a **Multi-Stage Design Graph**.

be noted that due to the transformation from a tree to a DAG all leaf nodes are now referred to as sources and the root node as the sink. As described in subsection 3.3.2, the representation of the robot component as DAG allows to model different solution principles as different paths.

In the next step, *requirements checks* are added so that the DAG not only describes the bottom-up design of the robot component, but also takes requirements into account. These requirement checks are performed at each step during the generation of the robot component. As will be explained in more detail later in subsection 4.3.2, this serves to exclude unsuitable catalog components and partial solutions as early as possible. In the DAG, requirement checks are represented by new nodes (white) with the prefix *Satisfied*. To include these nodes, the number of nodes is first doubled by creating exactly one counterpart v_s for each node v of the DAG. This pairing is expressed by the name of counterpart node v_s, which differs only from v by the prefix *Satisfied*. For example, in Figure 3.5 *Satisfied Catalog Component A* is the counterpart node to *Catalog Component A*. These new counterpart nodes v_s are integrated into the DAG in a way that they are the only successor nodes of v. The original successor nodes of v will then become the successor nodes of the counterpart node v_s. The only exception to this are the nodes for the original supertypes of the SysML BDD for solution principles, from which different paths now originate. As illustrated in Figure 3.4, the successor nodes of *Partial Solution* $(ABC)_i$ are still the two solution principle variants *Partial Solution* $(ABC)_1$ and *Partial Solution* $(ABC)_2$, while its counterpart node *Satisfied Partial Solution* $(ABC)_i$ is used to reunite the two different paths.

Graph Traversal with Requirement Checks

As in the initially presented system hierarchy tree (subsection 3.3.1), the edges of the graph represent transitions from one state to another. This is described by rules and formulas. The edges between nodes v and their counterpart v_s represent requirement checks. IF-THEN rules define under which constraints requirements are *satisfied* and when they are not. For the bottom-up traversal of the DAG (see also Figure 3.5), this proceeds as follows:

1. The source nodes of the DAG, catalog components or solution principles, are checked whether they are suitable to be part of a robot component satisfying a given set of high-level requirements or not. Depending on the type of requirement, the procedure is different:

   a) **Functional requirements**: These requirements can usually be allocated directly to catalog components or solution principles. Thereby the functional requirement is mapped to a property, which serves as a constraint. An IF-THEN rule checks if the catalog component or solution principle has this property or not:
   IF *ComponentHasProperty==true* THEN *SatisfiedRequirement*

   b) **Performance requirements**: In contrast to functional requirements, performance requirements for the overall system such as spatial dimensions usually cannot be mapped directly to a property of a single catalog component. The performance requirement must therefore be hierarchically decomposed until constraints on the selection of the catalog component can be derived. Since at this point is not yet determined which other catalog components and solution principles will be selected, it is possible that parameters in the formulas are still unknown. In this case, heuristic assumptions based on expert knowledge must be made, which allow early recognition of which catalog components will not satisfy the requirements even in the best case. For example, catalog components should be excluded directly if they are already larger, heavier or more expensive than the overall system is allowed to be.

   c) **Interface requirements**: This type of requirement can be decomposed into functional and performance requirements for catalog components or solution principles - see a) and b).

2. If all requirements for a catalog component (or solution principle) are satisfied, it is considered a satisfied catalog component (or solution principle) of its type. Satisfied catalog components and solution principles are represented in the DAG by the counterpart nodes with the prefix *Satisfied*.

3. During further traversing of the graph, partial solutions are formed by combining different catalog components and/or solution principles. As indicated by the graph structure, only satisfied catalog components and satisfied solution principles are considered.

4. The newly formed partial solutions are checked like catalog components and solution principles with regard to high-level requirements for the overall system. High-level functional and interface requirements can be completely satisfied by selecting suitable catalog components and solution principles. Therefore, the checks for partial solutions concentrate on performance requirements. Since the partial solutions are already more precisely defined with regard to the selection of catalog components and solution principles, more parameters of the formulas are known. This allows a more precise calculation of the property used to check the performance requirement. As a result, the check can be more restrictive.

5. In order not only to consider high-level requirements but also physical feasibility and compatibility between catalog components and solution principles, it may be necessary to define further constraints that must be satisfied by the partial solution.

6. If all requirements and constraints for a partial solution are satisfied, it is considered a satisfied partial solution of its type and represented by a corresponding counterpart node with the prefix *Satisfied*.

7. to n.: In the following, the DAG is traversed further bottom-up and steps 3 to 6 are performed for partial solutions with increasing hierarchy level. The properties for the performance requirements checks can be calculated more and more precisely. At each transition from a node to its satisfied counterpart node, partial solutions are discarded before the remaining ones are combined to higher-level partial solutions. As a consequence, catalog components and solution principles that were only part of discarded partial solutions are also no longer considered. The process ends

at the satisfied counterpart node of the overall system, the *Satisfied Robot Component*, representing design solutions that meet all requirements.

The generalized model in combination with the described procedure takes both parts of the design process into account, the (1) top-down requirement decomposition and allocation process as well as the (2) bottom-up synthesis of design solutions. While bottom-up synthesis is reflected in the structure of the DAG and its bottom-up traversal, requirements are mainly considered by IF-THEN rules that are used to determine which catalog components, solution principles, partial solutions or overall solutions meet the requirements. Functional and interface requirements can be decomposed and allocated directly to catalog components or solution principles. As a result, they can be validated very early. Performance requirements, on the other hand, are usually checked several times when traversing the DAG with increasing accuracy of the calculated, relevant properties. This reflects the iterative and recursive nature of the design process as described by systems engineering (subsection 2.1.3).

3.4. Ontological Knowledge Representation

The final step of the formalization process is the transformation of the generalized model into ontologies and rules that can be used as knowledge base of an expert system. Besides the generalized model, knowledge about catalog components is also stored in the ontologies. After motivating the selection of ontologies as a knowledge representation technique, this section describes the architecture of the ontological knowledge base and the rules in detail.

3.4.1. Why Ontologies?

As mentioned in subsection 2.2.4, ontologies have evolved from frames, semantic networks and finally description logics. By using *SWRL* (O'Connor et al., 2005), IF-THEN rules can also be stored in the ontology. Ontologies therefore combine the advantages of several classical knowledge representation techniques. This results in a powerful knowledge representation, which, as for many other applications, is well suited to store design knowledge about robot components. The main advantages are as follows:

- **Machine-readable:** Ontologies are formal and explicit specification of conceptualizations that are suitable for automated reasoning.

- **Knowledge generation:** Ontologies have the advantage that not all facts have to be explicitly specified. If they are well designed they can also be used to extract explicit knowledge. This is of particular interest for design processes during which knowledge gaps often have to be bridged (Hu et al., 2004).
- **Accessibility and extendability:** Ontologies are easy to understand for human users because of their structure, which is mostly based on taxonomies. They can be easily modified and extended by user-friendly editors like Protégé (Noy et al., 2000).
- **Common language:** Ontologies provide a common language for software developers and experts providing domain knowledge thanks to their high semantic expressiveness.
- **Separate use:** The separation of knowledge representation and reasoning allows both parts to be used separately. Thus ontological knowledge bases can be used for different applications.
- **Standardization:** Due to their increasing popularity (Wagner, 2017), ontologies are the best known representation of knowledge in current literature (Ramos et al., 2018). Standardization in the form of upper ontologies allows the use of this trend to match ontologies from different sources and persons.
- **Description of complex domains:** Ontologies provide enough concepts and relations to describe even complex domains. An example of this is the biomedical domain: The Open Biomedical Ontology (OBO) Foundry (Smith et al., 2007), a collective of numerous ontology developers, works on interoperable ontologies that can be used for applications such as expert systems. Also with regard to design, different developers confirm that ontologies are a suitable form of knowledge representation (Hu et al., 2004; Juarez et al., 2011; Nilsson et al., 2009; Ramos et al., 2017). They allow for describing the concepts underlying the design process (functionalities, structures and resources) and their relationships to one another (Ramos et al., 2018).

3.4.2. Ontology Architecture

In the final step of the formalization process, the knowledge describing the design of a humanoid robot component type is stored in several ontologies that import each other. Figure 3.6 shows the modular ontology architecture,

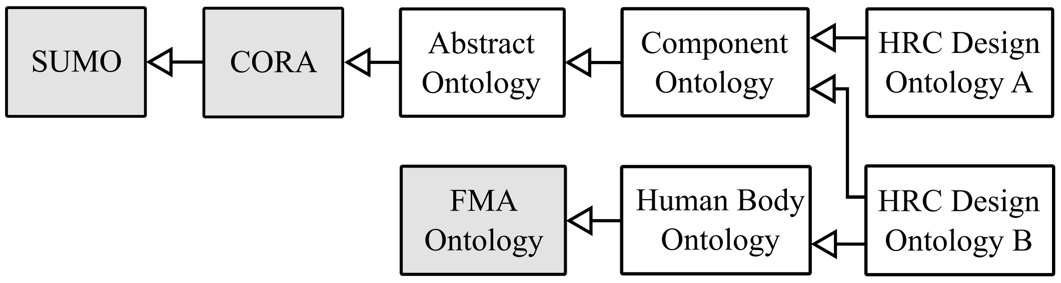


Figure 3.6.: UML diagram of the ontology architecture describing relationships between upper ontologies (gray) and newly developed ontologies (white).

consisting of existing upper ontologies (gray) and newly developed ontologies (white). The modular development offers the advantage of well defined interfaces that simplify the independent development of ontologies, especially when different developers are involved. Furthermore, modularity also simplifies the separate use of ontologies for different applications. And finally, it avoids prolonging the reasoning process by unnecessarily extensive ontologies. All ontologies shown in Figure 3.6 are implemented in *OWL2* (subsection 2.2.4). In the following they are described in detail.

Upper Ontologies

As described in subsection 2.2.4, the development of new ontologies based on upper ontologies, existing ontologies describing general concepts, brings many benefits. For this reason, when creating new ontologies, one or more upper ontologies are usually loaded first. Entities of the new ontology will then, if possible, be linked by inheritance or other relationships to existing classes or other entities. Three upper ontologies were chosen as a basis for the ontologies presented in this work: SUMO, CORA and FMA. The *Suggested Upper Merged Ontology (SUMO)* is a very general upper ontology that can be used for various domains. The *Core Ontology for Robotics and Automation (CORA)* uses SUMO as its upper ontology and extends it by adding entities for the domain of robotics and automation. SUMO and CORA are described in more detail in subsection 2.2.4. The *Foundational Model of Anatomy (FMA) ontology* is a reference ontology for anatomy (Rosse and Mejino, 2008). It represents entities and relationships that are necessary for the symbolic modeling of the structure of the human body. In this work, the FMA serves as a basis for representing knowledge on human kinematics, which is required for the design of humanoid robot components.

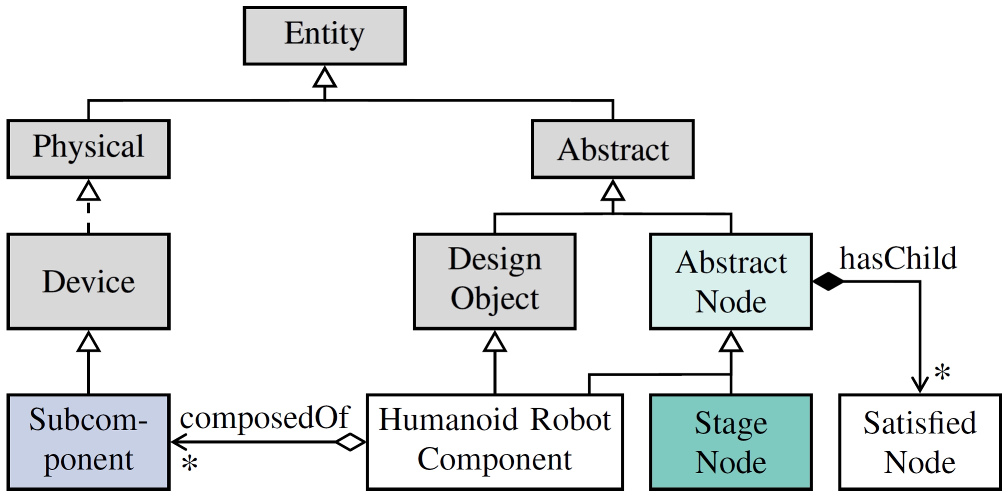


Figure 3.7.: UML class diagram of the Abstract Ontology. It is based on the upper ontologies SUMO and CORA (gray).
Source: (Karrenbauer, Rader and Asfour, 2018) © 2018 IEEE.

Abstract Ontology

The Abstract Ontology describes the general concept of what a humanoid robot component is and how it can be systematically designed. This includes the representation of a humanoid robot component as a Multi-Stage Design Graph as well as the description of the associated procedure to generate it in a bottom-up process with requirement checks (section 3.3).

Figure 3.7 illustrates classes of the Abstract Ontology and their relationships, which are used to describe the nodes of the Multi-Stage Design Graph: *Subcomponent* (catalog component), *Stage Node* (partial solution), *Humanoid Robot Component* (overall system) and *Satisfied Node* (counterpart nodes for requirement checks). An important relationship is the object property *hasChild*: It describes that the overall solution (*HumanoidRobotComponent*) and partial solutions (*StageNode*) include lower-level partial solutions that fulfill all requirements (*SatisfiedNode*). In order to distinguish *Physical* catalog components from purely *Abstract* partial solutions when composing the overall solution , the *composedOf* object property (Figure 3.7) is used to express the relationship between *HumanoidRobotComponent* and *Subcomponent* instead of *hasChild*.

Besides presenting important classes and relations of the Abstract Ontology, Figure 3.7 illustrates classes of CORA and SUMO from which they inherit. As a consequence, the Abstract Ontology also represents the interface between CORA, SUMO and the more specific ontologies describing the design of a robot component type. But the Abstract Ontology does not only act as an interface to

Table 3.3.: Examples of classes of the different ontologies.

Ontology	Example (OWL 2)
Abstract	*SubClassOf (AbstractNode, SUMO:Abstract)* *SubClassOf (AbstractNode,* *restriction (hasChild, someValuesFrom (SatisfiedNode)))* *SubClassOf (StageNode, AbstractNode)* *SubClassOf (Subcomponent, SUMO:Device)*
Component	*SubClassOf (Motor, Abstract:Subcomponent)* *SubClassOf (ElectricMotor, Motor)* *SubClassOf (FramelessBrushlessDCMotor, ElectricMotor)* *SubClassOf (RoboDrive, FramelessBrushlessDCMotor)* *SubClassOf (ServoKitILM, RoboDrive)* *SubClassOf (ILM85x13, ServoKitILM)* *SubClassOf (ILM85x13,* *restriction (has_n_max, hasValue(2900)))*
Human Body	*SubClassOf (IndexFinger50thPercentileMale, FMA:IndexFinger)* *SubClassOf (IndexFinger50thPercentileMale,* *restriction (hasLengthDistalPhalanx, hasValue(16)))*
HRC Design	*SubClassOf (MotorGearboxMatch, Abstract:StageNode)* *SubPropertyOf (hasMotor, hasChild)* *SubClassOf (MotorGearboxMatch,* *restriction (hasMotor, someValuesFrom (SatisfiedMotor)))*

the upper ontologies. It also facilitates the realization of a minimum of interfaces between the ontologies and a possible inference machine. The code of the inference engine does not have to refer to entities describing the design of a specific robot component type, but can use the concepts generally described in the Abstract Ontology. The resulting encapsulation between knowledge related to specific robot components and the inference engine allows easy extension and reuse of both parts.

Component Ontology

The Component Ontology contains knowledge of catalog components and other existing hardware components that can be used to build robot components and similar mechatronic systems. Within the ontology, the catalog components are defined hierarchically. All catalog components are represented by the class *Subcomponent*, which is defined in the Abstract Ontology. Children of this class are main categories into which catalog components can be grouped, i.e. *Motor*, *Gearbox*, *Bearings*, *MotorController*, *Brake*, *Screw* and *Sensor*. These groups are further broken down hierarchically until specific catalog components are specified. Table 3.3 demonstrates this at the example of the motor *ILM85x13*. In order to describe this specific catalog component, the group *Motor* is further broken down by technology (*ElectricMotor*), structure (*FramelessBrushless-DCMotor*), manufacturer (*RoboDrive*) and series (*ServoKitILM*).

At each hierarchy level, data properties are assigned to the catalog component. Inheriting these properties facilitates adding similar catalog components. The data properties used have similarities with the requirements for mechatronic robot components (subsection 3.3.3) to which they are mapped. Typical data properties are spatial dimensions (*hasDimensionInnerDiameter*), costs (*has-Cost*), weight (*hasWeight*) as well as electrical (*hasVoltage*) and mechanical performance data such as maximum rotational speed (*has_n_max*). But also knowledge about the installation of a component (*hasScrewCount*) and compatibilities between components (*isCompatibleWithEncoder*) is stored as data property. In the ontology, data properties are treated like superclasses (*SubClassOf*), with the difference that in addition to the identifier (*has_n_max*), a value is specified (*hasValue(2900)*).

Most of the knowledge about catalog components comes from manufacturers' catalogs, which may provide specifications for the component itself, but also information about installing it. If relevant knowledge for the use of a catalog component is missing in the manufacturer's catalogs, it must be supplemented by further documentation material such as technical drawings, CAD models, circuit diagrams, enquiries to manufacturers or expert knowledge gained from the use of similar systems.

The Component Ontology is created in such a way that its knowledge can be used in the design of any robot component or any other mechatronic system. Furthermore, it is also possible to use the ontology independently of an expert system. For example, it is possible to use the Component Ontology to retrieve information on catalog components or to search for all known catalog components that have a certain property.

Human Body Ontology

The purpose of the Human Body Ontology is to provide knowledge about the human body needed to design robot components based on their human counterparts. It uses fragments of the FMA, which describe the *Musculoskeletal system* of the *Human body*, as upper ontology. The Human Body Ontology extends the FMA ontology by adding data on human proportions. The data source is DIN 33402, which comprises 69 human body measures based on the German resident population aged between 18 and 65 (Jürgens, 2004). All measures are given for the 5th, 50th and 95th percentile male and female. To include data for other percentiles in the Human Body Ontology, measures for missing percentiles were derived from known data by assuming a normal distribution.

Table 3.3 lists the class *IndexFinger50thPercentileMale* as an example for the Human Body Ontology. It comprises measures for the class *IndexFinger* from the FMA ontology. Specifically, it describes measures for the 50th percentile male, i. e. measures of a man who is larger than 50% of the population and smaller than the other 50%. The specified data property, the length of the distal phalanx of the index finger (*hasLengthDistalPhalanx*), is assigned a value of 16 mm (*hasValue(16)*).

Humanoid Robot Component (HRC) Design Ontologies

The Humanoid Robot Component (HRC) Design Ontologies are the only specific ontologies of the modular ontology architecture. Each of them includes knowledge that describes the design of a specific robot component type. As shown in Figure 3.6, a distinction is made between *HRC Design Ontology A* and *B*. Type A imports the upper ontologies SUMO and CORA as well as the Abstract and Component Ontology. Type B additionally imports the FMA and Human Body Ontology. It is intended for the design of robot components based on their human counterparts.

Each HRC Design Ontology includes knowledge describing the generalized model of a robot component type (section 3.3). Specifically, these are all nodes of the *Multi-Stage Design Graph* and classes that confirm that single requirements such as a maximum rotation speed are met (*SatisfiedRotationSpeed*). Additionally, the HRC Design Ontology defines object properties, which are used to define the order of the nodes when traversing the graph. Furthermore, relationships between individuals of the node classes are defined, based on the *hasChild* object property defined in the Abstract Ontology. Table 3.3 demonstrates this using the example of the object property *hasMotor*, which assigns an individual motor to a partial solution (*StageNode*) consisting of a motor and a gearbox (*MotorGearboxMatch*).

Besides definitions of classes and properties, the HRC Design Ontology also comprises the rule set associated with the generalized model of a specific robot type, which is explained in the following.

Table 3.4.: Examples of formulas modeled as SWRL rules.

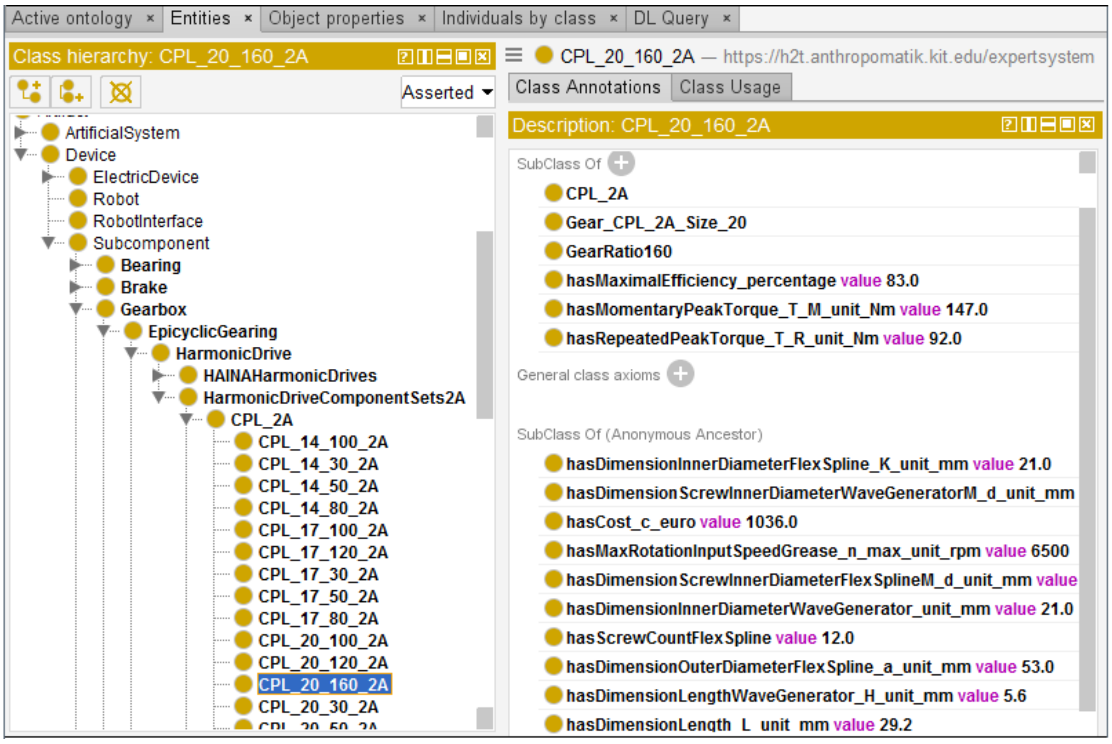
Formula	**SWRL Rule**
$n_{Mi,max} = \frac{n_{M,max}}{i}$	$MGM(?mgm)\hat{}hasM(?mgm,\ ?m)\hat{}$ $hasG(?mgm,\ ?g)\hat{}has\_n\_M\_max(?m,$ $?n\_M\_max)\hat{}has\_i(?g,\ ?i)\hat{}swrlb:$ $divide(?n\_Mi\_max,\ ?n\_M\_max,\ ?i)$ $\rightarrow has\_n\_Mi\_max(?mgm, ?n\_Mi\_max)$
$n_{Req,max} \leq n_{Sol,max}$	$MGM(?mgm)\hat{}Requirement(?req)\hat{}$ $has\_n\_Sol\_max(?mgm,\ ?n\_Sol\_max)\hat{}$ $has\_n\_Req\_max(?req,\ ?n\_Req\_max)\hat{}$ $swrlb: lessThanOrEqual($ $?n\_Req\_max,\ ?n\_Sol\_max)$ $\rightarrow Satisfied\_n\_max(?mgm)$

3.4.3. Rule Set

Both the rules and formulas describing the transitions between the nodes of the Multi-Stage Design Graph are stored in the HRC Design Ontologies of the corresponding robot components. This is made possible by the fact that rules formulated in *Semantic Web Rule Language SWRL* syntax can be stored in OWL files (subsection 2.2.4). Ultimately, this means that rules can be part of an ontology just like classes, individuals, and properties.

SWRL rules provide the means to express IF-THEN rules (antecedent $\rightarrow$ consequent), which can be solved in the sense of Horn-clauses. The examples in Table 3.4 demonstrate that SWRL rules can express both, formulas for calculating properties ($n_{Mi,max} = \frac{n_{M,max}}{i}$) as well as rules for requirement checks ($n_{Req,max} \leq n_{Sol,max}$). The SWRL rule for $n_{Mi,max} = \frac{n_{M,max}}{i}$ describes the maximum speed $n_{Mi,max}$ of a motor-gearbox combination after a gearbox reduction of i. It checks first, if there are motor-gearbox combinations $MGM(?mgm)$ that include a motor $hasM(?mgm, ?m)$ and a gearbox $hasG(?mgm, ?g)$. Then the rule refers to the speed of the motor $?n\_M\_max$ and the ratio $?i$ of the gearbox that are needed to perform the arithmetic operation $swrlb: divide(?n\_Mi\_max,\ ?n\_M\_max,\ ?i)$. If all conditions are fulfilled, the calculated value is assigned to the motor-gearbox combination $has\_n\_Mi\_max(?mgm, ?n\_Mi\_max)$.

The second rule $n_{Req,max} \leq n_{Sol,max}$ describes a constraint for a requirement check. The constraint is derived from a performance requirement regarding the maximum speed of a motor-gearbox combination. If the speed of the solu-



Class hierarchy: CPL_20_160_2A

- ArtificialSystem
- Device
  - ElectricDevice
  - Robot
  - RobotInterface
  - Subcomponent
    - Bearing
    - Brake
    - Gearbox
      - EpicyclicGearing
        - HarmonicDrive
          - HAINAHarmonicDrives
          - HarmonicDriveComponentSets2A
            - CPL_2A
              - CPL_14_100_2A
              - CPL_14_30_2A
              - CPL_14_50_2A
              - CPL_14_80_2A
              - CPL_17_100_2A
              - CPL_17_120_2A
              - CPL_17_30_2A
              - CPL_17_50_2A
              - CPL_17_80_2A
              - CPL_20_100_2A
              - CPL_20_120_2A
              - CPL_20_160_2A
              - CPL_20_30_2A

CPL_20_160_2A — https://h2t.anthropomatik.kit.edu/expertsystem

Description: CPL_20_160_2A

SubClass Of
- CPL_2A
- Gear_CPL_2A_Size_20
- GearRatio160
- hasMaximalEfficiency_percentage value 83.0
- hasMomentaryPeakTorque_T_M_unit_Nm value 147.0
- hasRepeatedPeakTorque_T_R_unit_Nm value 92.0

General class axioms

SubClass Of (Anonymous Ancestor)
- hasDimensionInnerDiameterFlexSpline_K_unit_mm value 21.0
- hasDimensionScrewInnerDiameterWaveGeneratorM_d_unit_mm
- hasCost_c_euro value 1036.0
- hasMaxRotationInputSpeedGrease_n_max_unit_rpm value 6500
- hasDimensionScrewInnerDiameterFlexSplineM_d_unit_mm value
- hasDimensionInnerDiameterWaveGenerator_unit_mm value 21.0
- hasScrewCountFlexSpline value 12.0
- hasDimensionOuterDiameterFlexSpline_a_unit_mm value 53.0
- hasDimensionLengthWaveGenerator_H_unit_mm value 5.6
- hasDimensionLength_L_unit_mm value 29.2

Figure 3.8.: Definition of the catalog component CPL-20-160-2A, a Harmonic Drive gearbox, with Protégé (Left: Class hierarchy; Right: Description).

tion $?n\_Sol\_max$ is greater than the speed $?n\_Req\_max$ defined in the requirement $Requirement(?req)$, the property $Satisfied\_n\_max(?mgm)$ is assigned to the motor-gearbox combination.

3.4.4. Ontology Modeling

The ontologies are created with Protégé (Noy et al., 2000), an easy-to-use ontology editor and knowledge management system. It allows both, the ontology developers and domain experts, to easily extend the ontologies. Figure 3.8 illustrates how ontology classes are displayed in the Protégé editor using the catalog component *CPL-20-160-2A* as an example. By displaying the class hierarchy (Figure 3.8, left) it is easy to categorize the class defining the catalog component within the ontology. In this case, *CPL-20-160-2A* is a *Harmonic Drive Component Set* of the *CPL-2A* series. The description of the component (Figure 3.8, right) lists the classes from which the class inherits as well as properties. In this case, three data properties are used to relate individuals of *CPL-20-160-2A* to concrete values: One data property describes the maximal efficiency of the catalog component (83%), two other data properties are peak torque specifications of the manufacturer (92 Nm and 147 Nm). Further properties are inherited by superclasses (Figure 3.8, bottom right). For example, *CPL-20-160-2A* inherits properties such as spatial dimensions, costs and weight from very similar Harmonic Drives (*Gear CPL-2A Size 20*) that differ only in the gear ratio. The inheritance of properties considerably reduces the development effort of ontologies. It should be noted that ontologies, unlike taxonomies, allow multiple inheritance.

Unlike catalog components, customized parts are not represented by a class in the Component Ontology. Instead, their properties (e. g. spatial dimensions, mass, costs) are defined in the course of reasoning as part of the generated partial solutions (subsection 4.3.3).

3.5. Summary and Review

This chapter presented the first two steps of the approach of this thesis: the acquisition and the formalization of expert design knowledge.

First, it was described how to acquire the design knowledge required for the approach presented in this work. The main source is the knowledge gained through the design of robot components, because successful implementation and evaluation ensures the reliability of the knowledge. This is extended by knowledge gained through the analysis of well-documented designs from related work. Furthermore, it was described which knowledge must be gained through system analysis in order to formalize the design knowledge.

This was followed by the introduction of a novel formalization process, which allows the transformation of knowledge gained during robot design into an ontological knowledge base of an expert system. To this end, it was described how a detailed generalized model of a robot component can be developed that takes both *procedural knowledge* as well as *domain-specific expert knowledge* into account.

Procedural knowledge as it is provided by product development and systems engineering is considered by modeling the design of the robot component as a directed acyclic graph (DAG). The graph describes the incremental bottom-up design of the robot component. Inspired by morphological boxes, different solution principles for conceptual design parameters are taken into account. They are represented by different nodes or paths. Further nodes serve to evaluate requirements. The identification of structures, the search for different solution principles and their subsequent use for a bottom-up synthesis while continuously evaluating requirements represents typical procedures of product development and systems engineering.

Domain-specific expert knowledge is mainly represented by formulas and rules that describe the transition between the nodes of the DAG. They define which subcomponents, solution principles and partial solutions under which conditions are combined to build higher-level partial solutions. In addition, they define which properties result from this combination. Domain-specific expert knowledge on specific catalog components is taken into account by collecting data from manufacturer catalogs.

By considering procedural as well as domain-specific knowledge, the presented graph-based approach allows to model knowledge about systematic design of complex, highly integrated robot components in detail. However, the novel formalization process does not only include the development of the generalized model, but also its transformation into an ontological knowledge base that allows for automated reasoning. Hence, it describes how systematic development of mechatronic systems can be enriched by methods of knowledge representation and reasoning, a branch of artificial intelligence, so that parts of the design process can be automated.

With regard to the research question, the presented formalization process described *how knowledge on the design of humanoid robot components can be preserved*. The following chapter will describe how an expert system can be created that uses this knowledge to *support future developments*.

4. Expert System for Robot Design

Once formalized, the knowledge is used to support the design of humanoid robot components as part of an expert system. The development of expert systems is located in the field of artificial intelligence and aims to create programs that can assist in solving problems similar to human experts (section 2.2). In this work an expert system is developed, which generates design solutions for robot components based on user requirements (Figure 4.1). This automates large parts of the design process. As motivated in chapter 1, with regard to NASA Systems Engineering this encompasses the technical requirement definition, the logical decomposition and finally the design solution definition.

This chapter describes the architecture and the individual components of the expert system. Furthermore, it includes a run-time analysis for the presented Multi-Stage Reasoning Process. First results on the presented expert system framework were published in (Karrenbauer, Rader and Asfour, 2018).

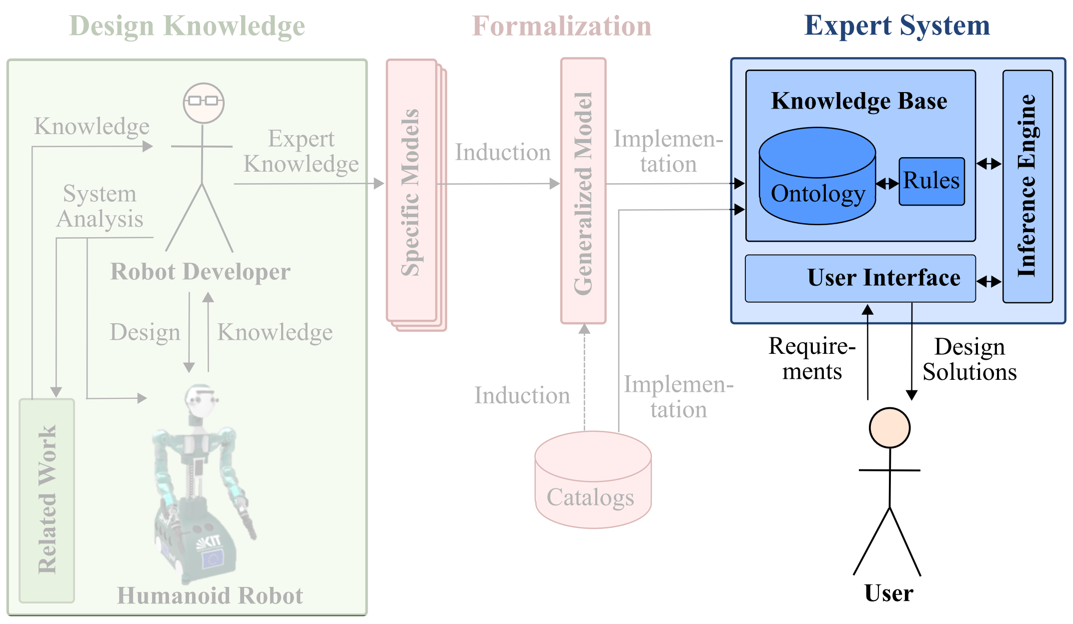


Figure 4.1.: Expert system (blue) to support the design of humanoid robot components.

4.1. Architecture

As illustrated in Figure 4.1, the expert system consists of three components: a knowledge base, an inference engine and a user interface (UI). These are the typical components that today's expert systems usually consist of. The user selects the robot component to be designed in a user interface. Depending on his selection, different sets of requirements can be specified. Performance requirements can be prioritized by weight and allowed deviations can be set. In addition to defining and prioritizing requirements, the user interface also serves to start the reasoning process. The reasoning is controlled by an inference engine. It infers design solutions from the ontological knowledge base by executing a Multi-Stage Reasoning Process. At the end of a reasoning procedure, the resulting design solutions are visualized by the user interface, including catalog components, chosen solution principles and calculated properties.

4.2. Knowledge Base

The ontological knowledge base of the expert system is the result of the formalization process presented in chapter 3. It is composed of several ontologies (subsection 3.4.2) that contain all the knowledge needed to automatically generate design solutions for a given robot component type: the general approach how to design robot components (*Abstract Ontology*), the Multi-Stage Design Graph that describes the robot component type (*HRC Design Ontology*), catalog component data (*Catalog Ontology*) and - if needed - knowledge on human body proportions (*Human Body Ontology*). Besides classes, individuals and properties, the *HRC Design Ontology* also includes *SWRL rules*, which are used to describe the transitions of the Multi-Stage Design Graph by calculating properties and checking requirements. All ontologies are implemented in *OWL2*. Their structure and creation with Protégé have already been discussed in detail in section 3.4. During runtime, all ontologies are combined into a single ontology from which the reasoner derives design solutions. In the following, this ontology is referred to as *working ontology*.

4.3. Inference Engine

The purpose of the inference engine is to infer new information from the knowledge base. In the case of this particular expert system, its task is to generate design solutions for robot components based on the ontological knowledge base. After a short introduction to the reasoner, a novel Multi-Stage Reasoning Process is described, which is used to generate robot components in a bottom-up approach taking user requirements into account.

4.3.1. Pellet Reasoner

The inference engine is written in Java and utilizes Pellet (Sirin et al., 2007), an open-source Java-based reasoner. Pellet is very popular since it provides the most complete support for OWL and SWRL expressions (Šaša Bastinos and Krisper, 2013). Therefore, it is suitable for both ontological and rule-based reasoning. In order to access and modify the working ontology, it uses the OWL API (Horridge and Bechhofer, 2011).

In this work the Pellet reasoner is used for a *deductive reasoning* process, in which knowledge is derived from ontologies and rules by *forward chaining* (subsection 2.2.5). However, the reasoning process presented in the following differs significantly from conventional approaches, since the reasoner does not perform just one but several runs.

4.3.2. Multi-Stage Reasoning

Each robot component represents a combination of catalog components and solution principles. A brute force approach would consist of firstly generating any possible combination and secondly evaluating them. However, this is inefficient and leads to a combinatorial explosion. Instead of generating each combination for a robot component type at once, it is more effective to generate partial solutions first, sub-combinations of catalog components and solution principles. These partial solutions are then evaluated against the requirements. Only partial solutions that fulfill these requirement checks are pursued further by combining them with other partial solutions to form higher-level partial solutions, i.e. combinations of sub-combinations. This procedure is continued until solutions for the robot component type, complete combinations, are generated and evaluated. As a result, unsuitable combinations can be discarded at the earliest stage possible.

But how can this approach be realized when inferring from an ontological knowledge base? Šaša Bastinos and Krisper (2013) use ontologies and SWRL rules to model a decision tree that structures the reasoning process for a multi-criteria decision problem. Thus, an individual can be evaluated according to several criteria in sequence. The idea of combining a graph-based approach with ontological reasoning is adapted for the approach presented in this work. But instead of applying the graph to an individual like Šaša Bastinos and Krisper (2013), this approach combines individuals in order to build new individuals while traversing the graph. The creation of new individuals is performed generically by the Java code at the transitions between the graph nodes. Since new individuals are evaluated by the reasoner after their creation, but before their combination with other individuals, the reasoner is not only called once, but several times. The approach is therefore referred to as *Multi-Stage Reasoning* in the following.

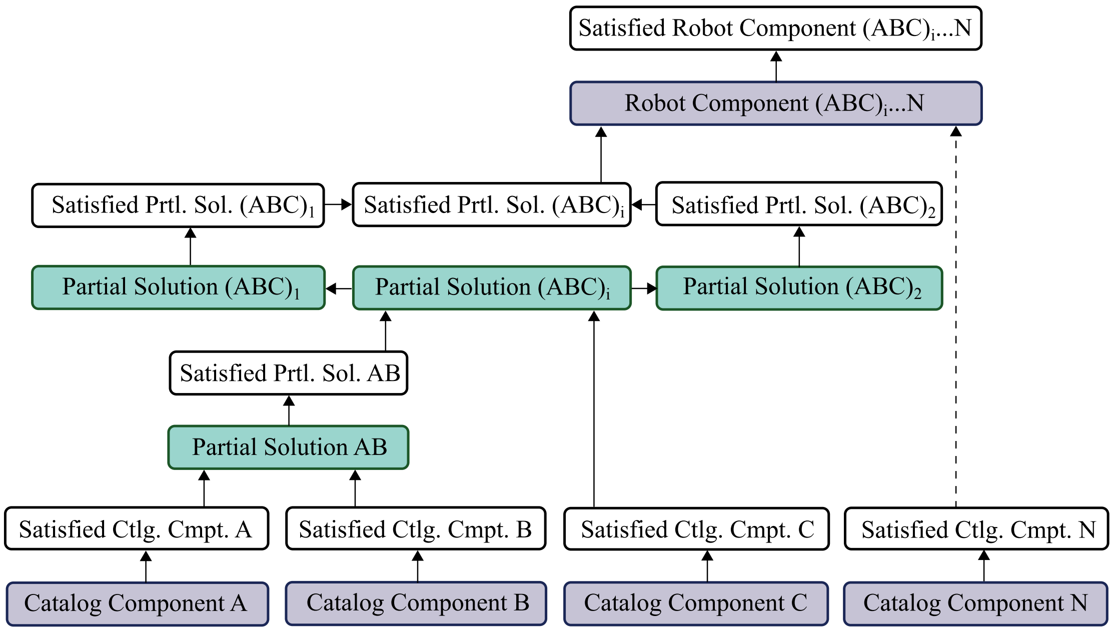


Figure 4.2.: During the Multi-Stage Reasoning Process the Multi-Stage Design Graph is traversed bottom-up.

The *Multi-Stage Reasoning Process* is based on a bottom-up traversal of the *Multi-Stage Design Graph*, which describes how a robot component is built up incrementally with requirement checks (Figure 4.2). The creation and the structure of this graph as well as its transformation into an ontological knowledge base have already been described in chapter 3. In the following the algorithm for the Multi-Stage Reasoning is presented, which refers to elements of the Multi-Stage Design Graph represented by the Abstract Ontology (Figure 3.7).

The Multi-Stage Design Graph is a directed acyclic graph $G = (V, E)$.

$$V = \{(v_1, .., v_k), (v_{k+1}, .., v_{n-1}), v_n\} \tag{4.1}$$

$$E = \{hasChild\} \tag{4.2}$$

Algorithm 1 Multi-Stage Reasoning

```
procedure REASONING($G = (V, E)$)                    ▷ $G$ is traversed bottom-up
    for all $\{v \mid v \in \{v_1, .., v_k\}\}$ do
        $createIndividuals(v.type)$
    end for
    for all $\{v \mid v \in \{v_{k+1}, .., v_n\}\}$ do
        for all $\{v_j \mid v_j \in V, e \in E, e = (v, v_j)\}$ do
            $\mathcal{I} := \mathcal{I} \cup \{(v_j, getIndividuals(v_j))\}$
        end for
        if $\forall(v_j, I_j) \in \mathcal{I} \mid I_j \neq \emptyset$ then
            $C := generateCombinations(\mathcal{I})$
            for all $\{c \mid c \in C\}$ do
                $i_{parent} := createIndividual(v.type)$
                for all $\{i_{child} \in c.I\}$ do
                    $v_j := getNode(i_{child})$
                    $e := (v, v_j)$
                    $createObjectPropertyAssertion($
                            $e.type, i_{parent}, i_{child})$
                end for
            end for
        end if
    end for
end procedure
```

Abbreviations: i individual, I_j set of satisfied individuals for v_j, $\mathcal{I}$ map of individuals I_j per v_j

The vertexes $(v_1, .., v_n)$ represent different entities of the reasoning process:

- $(v_1, .., v_k)$ are the source nodes of G and represented mostly catalog components, but also solution principles. In the Abstract Ontology they are modeled by the class $Subcomponent$.
- $(v_{k+1}, .., v_{n-1})$ are nodes of G that represent partial solutions. They describe dependencies between other nodes. In the Abstract Ontology they are modeled by the class $StageNode$.
- v_n is the sink node of G and represents the robot component to be designed, the overall solution. In the Abstract Ontology it is modeled by the class $HumanoidRobotComponent$.

The edges E are composed of concrete subclasses of the abstract $hasChild$ object property between the nodes V.

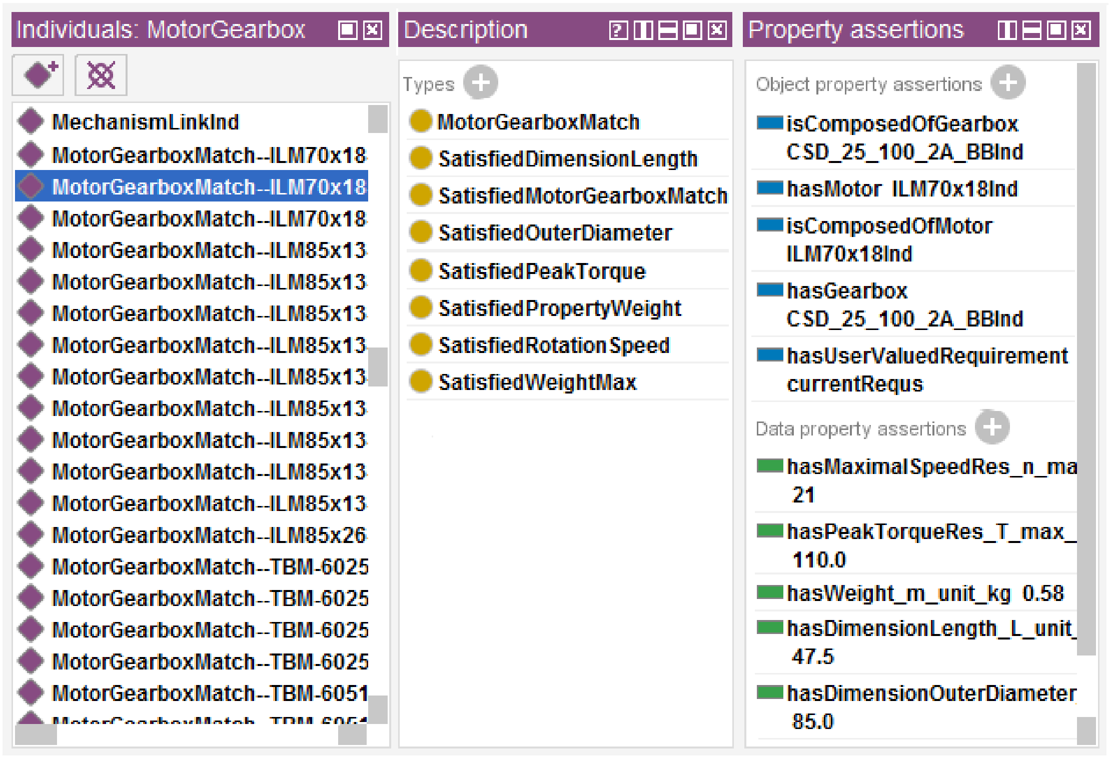
The Multi-Stage Reasoning (Algorithm 1) is executed for a Multi-Stage Design Graph G, which is traversed bottom-up. At the beginning, an individual for each *Subcomponent* node $(v_1, .., v_k)$ is created in the working ontology.

In the next step, $getIndividuals(v_j)$ is called for a given node v and each subnode v_j. This function uses the reasoner to evaluate which individuals of v_j have satisfied the SWRL rule set. The rule set is modeled in a way, that for each v_j, calculations and evaluation of constraints are executed. When v_j satisfies the constraints it will be assigned to a respective $SatisfiedNode$, for which $getIndividuals(v_j)$ is called. Only if each v_j has satisfied individuals ($I_j \neq \emptyset$), the node v will be further processed.

A set of combinations C is generated with $generateCombinations(\mathcal{I})$ for each individual over the set of $\{v_j\}$. For example, a motor-gearbox combination has two $hasChild$ dependencies to a motor and a gearbox. Each individual of a motor is then combined with each individual of a gearbox. For each combination c a new individual (i_{parent}) is created in the working ontology with the type of v (the motor-gearbox combination). The edges $\{e = (v, v_j)\}$ are mapped to a new object property assertion like $hasMotor$ ($e.type$) from the motor-gearbox combination (i_{parent}) to a motor (i_{child}). The algorithm terminates when all nodes V are traversed. Afterwards, the satisfied individuals of the $HumanoidRobotComponent$ v_n can be inferred along with the $composedOf$ subcomponents and the calculated properties, which can be partially mapped to the requirements.

4.3.3. Ontology Modifications during Reasoning

During the execution of Multi-Stage Reasoning Process, the working ontology (section 4.2) is continuously changed by creating, modifying and deleting individuals. Figure 4.3 illustrates how individuals of the working ontology are represented in Protégé using the example of motor-gearbox combinations. On the left side of Figure 4.3 all individuals of the working ontology are listed, usually several thousand. These individuals represent specific catalog components, solution principles, partial solutions as well as overall solutions that are created and modified during reasoning. The right side of Figure 4.3 shows the property assertions of the selected individual. For example, an individual of the motor *ILM70x18* is assigned to the selected motor-gearbox combination via the object property assertion *hasMotor*. In addition to other individuals, each motor-gearbox combination is also assigned data properties by executing rules, e.g. a weight of 580 g (*has_Weight_m_unit_kg 0.58*). The center of Figure 4.3 illustrates the classes of which the selected individual is a member. The *Satisfied* classes shown are assigned to individuals during reasoning to indicate which individuals meet requirements such as a maximum weight (*SatisfiedWeightMax*) and are therefore pursued further.



Individuals: MotorGearbox

- MechanismLinkInd
- MotorGearboxMatch--ILM70x18
- MotorGearboxMatch--ILM70x18
- MotorGearboxMatch--ILM70x18
- MotorGearboxMatch--ILM85x13
- MotorGearboxMatch--ILM85x13
- MotorGearboxMatch--ILM85x13
- MotorGearboxMatch--ILM85x13
- MotorGearboxMatch--ILM85x13
- MotorGearboxMatch--ILM85x13
- MotorGearboxMatch--ILM85x13
- MotorGearboxMatch--ILM85x13
- MotorGearboxMatch--ILM85x13
- MotorGearboxMatch--ILM85x26
- MotorGearboxMatch--TBM-6025
- MotorGearboxMatch--TBM-6025
- MotorGearboxMatch--TBM-6025
- MotorGearboxMatch--TBM-6025
- MotorGearboxMatch--TBM-6051

Description

Types

- MotorGearboxMatch
- SatisfiedDimensionLength
- SatisfiedMotorGearboxMatch
- SatisfiedOuterDiameter
- SatisfiedPeakTorque
- SatisfiedPropertyWeight
- SatisfiedRotationSpeed
- SatisfiedWeightMax

Property assertions

Object property assertions

- isComposedOfGearbox CSD_25_100_2A_BBInd
- hasMotor ILM70x18Ind
- isComposedOfMotor ILM70x18Ind
- hasGearbox CSD_25_100_2A_BBInd
- hasUserValuedRequirement currentRequs

Data property assertions

- hasMaximalSpeedRes_n_ma 21
- hasPeakTorqueRes_T_max_ 110.0
- hasWeight_m_unit_kg 0.58
- hasDimensionLength_L_unit_ 47.5
- hasDimensionOuterDiameter 85.0

Figure 4.3.: Individuals for motor-gearbox combinations in the working ontology.

4.3.4. Run-Time Analysis

For the run-time analysis of the Multi-Stage Reasoning Process, it is assumed that the number of individuals i created in the working ontology by the combination of catalog components and solution principles is the decisive variable. This can be explained by the fact that these individuals are not only created but also tested by the rules. The number of individuals strongly depends on the structure of the Multi-Stage Design Graph. Therefore, several parameters are identified to describe it (Table 4.1).

For the run-time analysis, it is assumed that the nodes of the Multi-Stage Design Graph are not changed, which means that the maximum number of direct predecessor nodes p and the longest path h of the directed acyclic graph (DAG) are fixed. Only the number of catalog components and solution principles n represented by the source nodes can be changed. Each source node represents n_j catalog components of one catalog component type or n_j solution principles for one parameter.

$$n = \sum_{j=1}^{k} n_j \tag{4.3}$$

Table 4.1.: Parameters describing the structure of the Multi-Stage Design Graph and the individuals resulting from it.

Parameter	**Description**
i	Total number of individuals created by the combination of catalog components and solution principles
i_v	Number of individuals (combinations) assigned to a node v
n	Total number of catalog components and solution principles assigned to multiple source nodes (resp. leaf nodes)
n_j	Number of catalog components for one specific component type or number of solution principles for one specific parameter assigned to a single source node j (resp. leaf node)
k	Number of source nodes (resp. leaf nodes) representing component types and parameters
p	Maximum number of direct predecessor nodes
h	Maximum path length of the DAG (resp. height of the tree)

The Multi-Stage Reasoning Process aims to discard solutions as early as possible. It can be interpreted as a brute force graph search with pruning, which is realized by rules. However, pruning is too case-specific to be considered in the run-time analysis. In this respect, the early exclusion of individuals by rules cannot be considered. The worst-case would be that all possible individuals are created because all rules are fulfilled. For a Multi-Stage Design Graph as shown in Figure 4.4 (a), the worst-case number of individuals to be created and tested is therefore:

$$i(n) \in \mathcal{O}(n^m) \tag{4.4}$$

$$m = p^h \tag{4.5}$$

Since m is defined by the structure of the DAG and thus fixed, the number of individuals grows polynomially to the number of catalog components and solution principles. To avoid a combinatorial explosion, the graph should be constructed in a way that m is as small as possible. First of all, the number of predecessor nodes should be reduced to $p \leq 3$ or even better $p \leq 2$. When determining the maximum path length h of the DAG, the *Satisfied Nodes* should not be taken into account, since during the transition from nodes to their *Satisfied* counterparts no new individuals are created. In general, before calculating the worst-case runtime, the DAG should be shortened by all nodes that have only one predecessor node ($p = 1.$).

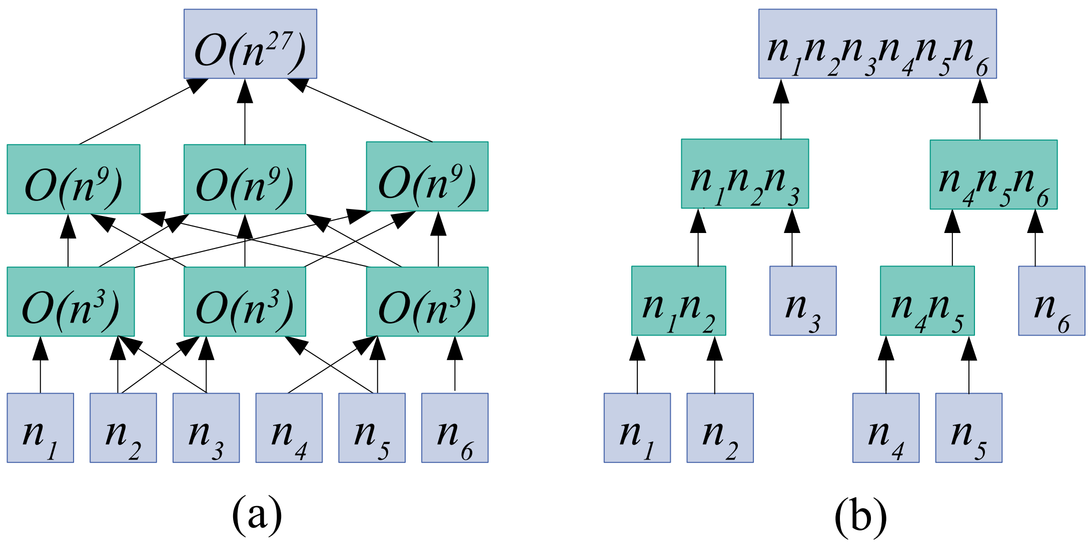


Figure 4.4.: Examples for Multi-Stage Design Graphs with a maximum path length h=3: (a) Directed Acyclic Graph (DAG) (b) Directed Tree.

Considering the Multi-Stage Design Graph as any DAG results in an upper boundary for generated individuals, which in most cases is far above the actual limit of a graph describing the design of a specific robot component. The average case is therefore based on a directed tree from which the DAG develops by a few additional paths. For a directed tree with k leaf nodes as shown in Figure 4.4 (b), the following upper limit results for the number of individuals for the root node r, the $HumanoidRobotComponent$:

$$i_r = \prod_{j=1}^{k} n_j \in \mathcal{O}(n^k)\ ,\ n_j \leq n \tag{4.6}$$

Similarly, the maximum number of individuals i_v can be calculated for every node v. By summing up all individuals that are assigned to a node, the total maximum i is obtained:

$$i = \sum_{v=1}^{r} i_v \in \mathcal{O}(n^k) \tag{4.7}$$

For the directed tree, the number of individuals also increases polynomially to the number of catalog components and solution principles. But the exponent k, the number of leaf nodes, is usually much lower.

4.4. User Interface

The user interface (UI) is developed in Java SWT (Northover and Wilson, 2004), a graphical widget toolkit that allows to quickly build user interfaces. The UI serves to define requirements, set allowed deviations and present design solutions to the user. Therefore, it provides three main tabs (*Requirements*, *Optimization* and *Solution*), which are presented in the following.

4.4.1. Requirements Definition

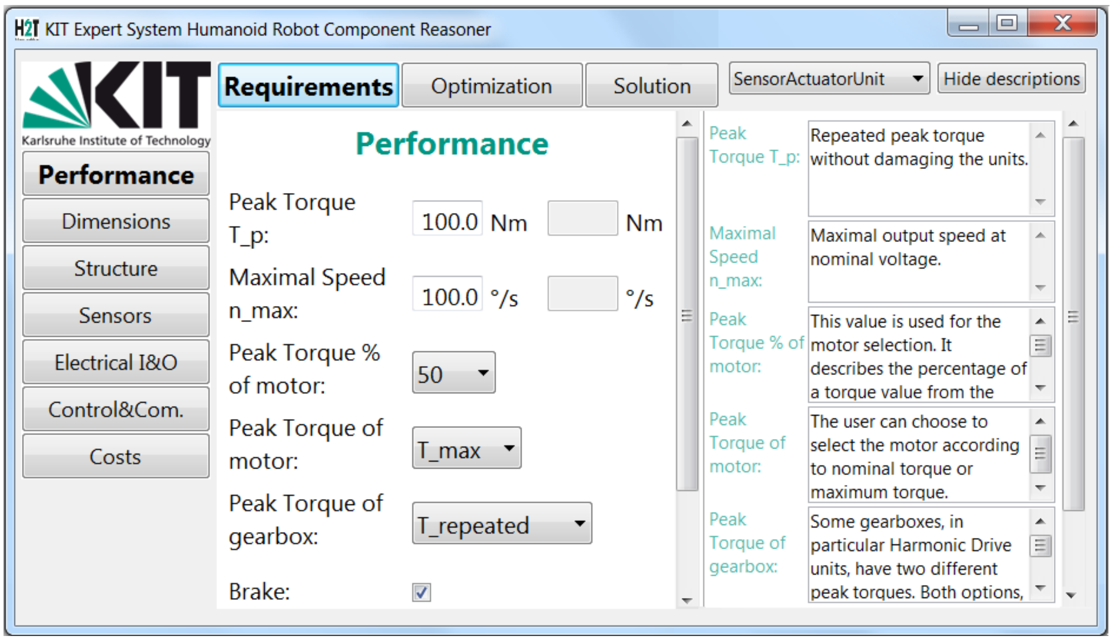
The requirements definition is specifically adapted to the robot component type. Therefore, when starting the program, the user must first select which robot component type is to be designed with the help of the expert system. This selection then defines which ontologies are loaded into the working ontology. The specific HRC Design Ontology contains individuals representing specific UI requirements. Each individual defines which user input is allowed for a requirement, which default values are used, in which category the requirement is classified and which description is displayed. Based on this ontology data, the UI is generated.

In the requirements definition tab the user can define functional, performance and interface requirements:

- **Functional requirements** are visualized by checkboxes or drop-down lists.
- **Performance requirements** are visualized by two input fields each for defining a minimum and maximum value.
- **Interface requirements** are visualized like functional or performance requirements.

Besides requirements definition, an expert mode allows the user to customize values used to calculate formulas and to choose specific solution principles. Furthermore, each requirement and value is explained to the user.

Figure 4.5 shows an example of a requirements definition tab. On the left side, the user can select the requirement category (*Performance*, *Dimensions*, *Structure*,...). For the selected category, the user defines functional requirements (*Brake*) and performance requirements (*Peak Torque* and *Maximal Speed*). Furthermore, the values used to calculate the formulas can be changed in expert mode (*Peak Torque % of motor*, *Peak Torque of motor* and *Peak Torque of gearbox*).



Performance

Peak Torque T_p:	100.0	Nm		Nm
Maximal Speed n_max:	100.0	°/s		°/s
Peak Torque % of motor:	50			
Peak Torque of motor:	T_max			
Peak Torque of gearbox:	T_repeated			
Brake:	✓			

Peak Torque T_p:	Repeated peak torque without damaging the units.
Maximal Speed n_max:	Maximal output speed at nominal voltage.
Peak Torque % of motor:	This value is used for the motor selection. It describes the percentage of a torque value from the
Peak Torque of motor:	The user can choose to select the motor according to nominal torque or maximum torque.
Peak Torque of gearbox:	Some gearboxes, in particular Harmonic Drive units, have two different peak torques. Both options,

Figure 4.5.: Requirements tab of the expert system in which the user can specify the requirements.

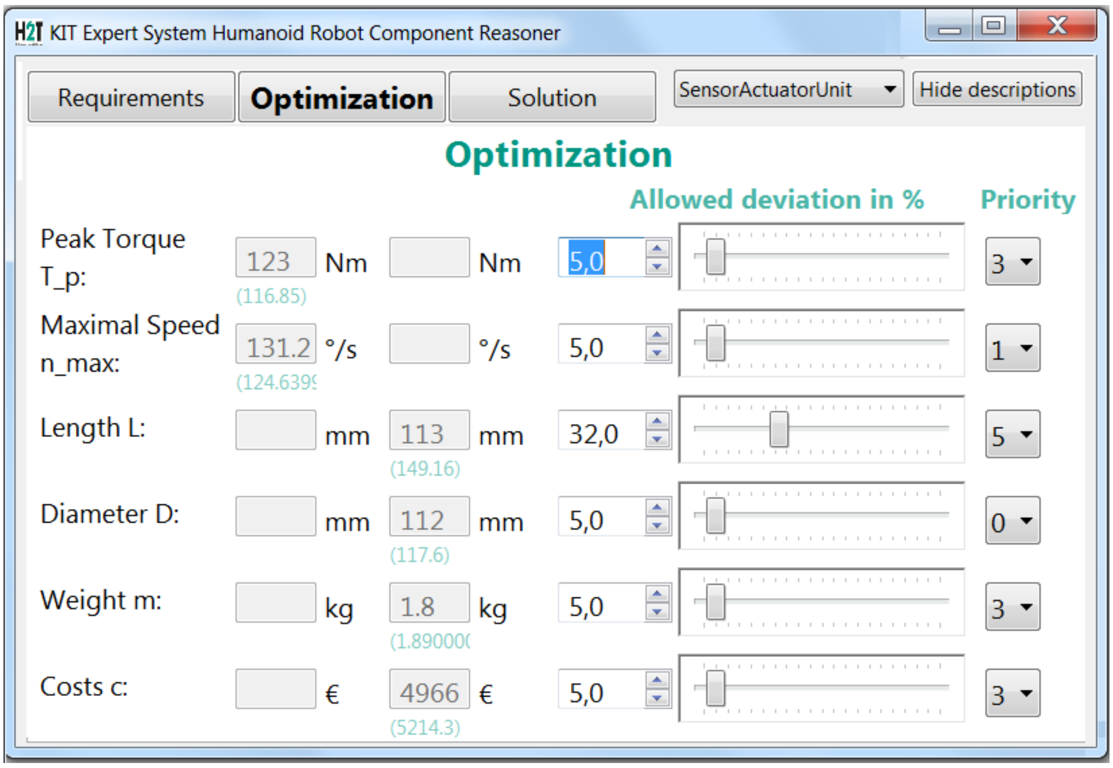
In this case, this enables experts to dimension motor-gearbox combinations according to different key characteristics, which are selected via a drop-down list. For each requirement and value a description is given on the right side.

4.4.2. Deviations, Rating and Optimization

As with all other requirements, performance requirements are defined in the requirements tab of the UI. For performance requirements, this is done by a minimum or maximum threshold value. However, as described in subsection 3.3.3, it is advantageous to define performance requirements not only by a threshold value, but also by an allowed deviation. This deviation can be adjusted in the optimization tab by a percentage value between 0 and 100 relative to the threshold value. The resulting constraints are displayed as numerical values in the UI (Figure 4.6). Instead of entering the deviation value in an input field, the user can also use a slider.

If several design solutions meet all requirements, selected performance requirements can be used to rank the solutions. To prioritize performance requirements differently, a weight $w_i \in \mathbb{N}_0$ between 0 (do not consider) and 5 (highest priority) can be selected in the optimization tab for each requirement i.

In the following two rating functions are introduced to rank design solutions on the basis of selected performance requirements and their weighting w_i: the *Nor-*



KIT Expert System Humanoid Robot Component Reasoner

Requirements | **Optimization** | Solution | SensorActuatorUnit | Hide descriptions

Optimization

					Allowed deviation in %	Priority
Peak Torque T_p:	123 (116.85)	Nm		Nm	5,0	3
Maximal Speed n_max:	131.2 (124.639	°/s		°/s	5,0	1
Length L:		mm	113 (149.16)	mm	32,0	5
Diameter D:		mm	112 (117.6)	mm	5,0	0
Weight m:		kg	1.8 (1.89000	kg	5,0	3
Costs c:		€	4966 (5214.3)	€	5,0	3

Figure 4.6.: Optimization tab of the expert system in which the user can specify the allowed deviations and priorities of the performance requirements.

malized Root-Mean-Square Deviation (NRMSD) and the *Performance Index (PX)*. The NRMSD is used to rank design solutions so that the solution with the least negative deviation ("error") from the performance requirements comes first. Positive deviations are not considered. In contrast, PX ranks first the solutions for which the positive deviations outweigh the negative deviations the most. PX is therefore particularly suitable for optimizing existing solutions.

Normalized Root-Mean-Square Deviation (NRMSD)

A performance requirement $u_{Req,i} \in U = U_{min} \cup U_{max}$ should either be maximized $u_{Req,i,min} \in U_{min}$ or minimized $u_{Req,i,max} \in U_{max}$. This allows to distinguish between a positive and a negative deviation. A deviation is negative, if $u_{Sol,i} < u_{Req,i,min}$ or $u_{Req,i,max} < u_{Sol,i}$, with $u_{Sol,i}$ the solution value for the performance requirement $u_{Req,i}$. A positive deviation is defined inversely and represents a value $u_{Sol,i}$ which is better than required. The relative error e_i is:

$$e_i = \begin{cases} (u_{Req,i})^{-1}\,(u_{Req,i} - u_{Sol,i}) & , u_{Req,i} \in U_{min} \\ (u_{Req,i})^{-1}\,(u_{Sol,i} - u_{Req,i}) & , u_{Req,i} \in U_{max} \end{cases} \tag{4.8}$$

As error function for the negative deviation a weighted variation of the *Normalized Root-Mean-Square Deviation* (*NRMSD*) is used. The error e_i is normalized between 0 and 1 to compare requirements with different ranges.

$$NRMSD = \sqrt{\left(\sum_{i=1}^{\#U} w_i\right)^{-1} \sum_{i=1}^{\#U} w_i \ \max(0, e_i)^2} \tag{4.9}$$

$$NRMSD \in [0,1] \tag{4.10}$$

The smaller the negative deviations, the smaller the NRMSD. If all performance requirements are fully met, meaning, if there are no negative deviations, $NRMSD = 0$. In the worst case, if all performance requirements have a negative deviation of 100% from their thresholds, which is the maximum allowed by the UI, $NRMSD = 1$.

Performance Index (PX)

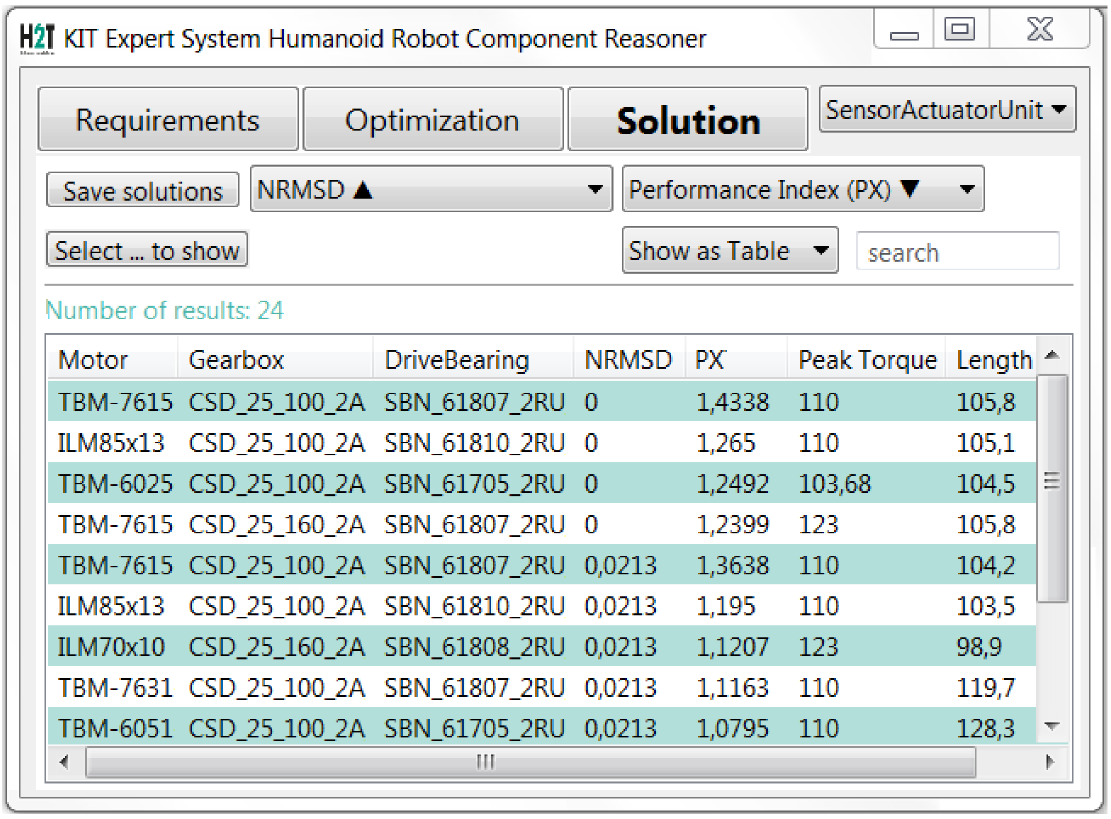
Beside the $NRMSD$ the *Performance Index* (PX) is calculated, which also takes positive deviations into account to represent an overall performance of a solution:

$$PX = \left(\sum_{i=1}^{\#U} w_i\right)^{-1} \sum_{i=1}^{\#U} w_i \ (1 - e_i) \tag{4.11}$$

$$PX \geq 0 \tag{4.12}$$

The higher the performance index PX, the better the design solution. In case of a negative deviation of all performance requirements by 100%, $PX = 0$ applies. If the values of the solution correspond exactly to the threshold values of the performance requirements, $PX = 1$. In case that $PX > 1$, the solution can be classified as better than required, the positive deviations outweigh the negative deviations.

PX can thus be used to check whether existing solutions, represented by threshold values of performance requirements, can be optimized. In order to control which priority the individual performance requirements have in this optimization, the user can use different weights w_i for them.



Number of results: 24

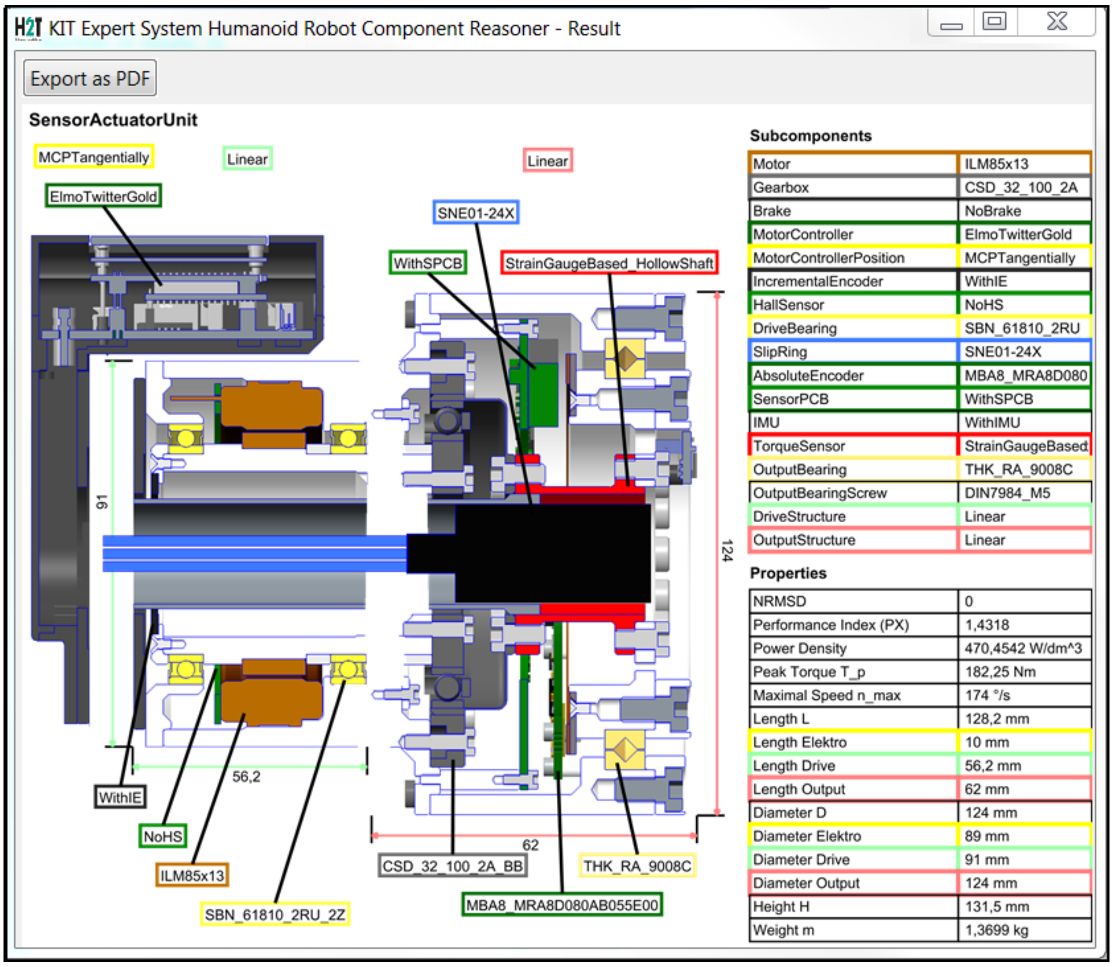
Motor	Gearbox	DriveBearing	NRMSD	PX	Peak Torque	Length
TBM-7615	CSD_25_100_2A	SBN_61807_2RU	0	1,4338	110	105,8
ILM85x13	CSD_25_100_2A	SBN_61810_2RU	0	1,265	110	105,1
TBM-6025	CSD_25_100_2A	SBN_61705_2RU	0	1,2492	103,68	104,5
TBM-7615	CSD_25_160_2A	SBN_61807_2RU	0	1,2399	123	105,8
TBM-7615	CSD_25_100_2A	SBN_61807_2RU	0,0213	1,3638	110	104,2
ILM85x13	CSD_25_100_2A	SBN_61810_2RU	0,0213	1,195	110	103,5
ILM70x10	CSD_25_160_2A	SBN_61808_2RU	0,0213	1,1207	123	98,9
TBM-7631	CSD_25_100_2A	SBN_61807_2RU	0,0213	1,1163	110	119,7
TBM-6051	CSD_25_100_2A	SBN_61705_2RU	0,0213	1,0795	110	128,3

Figure 4.7.: Solution tab of the expert system in which design solutions are listed.

4.4.3. Solution Presentation

The solutions tab has two functions: Firstly, clicking on this tab starts the reasoning process, and secondly it presents the design solutions to the user.

Each design solution consists of the selected catalog components and solution principles as well as the resulting properties, i.e. values for the performance requirements and the two rating functions. Only those solutions are presented that meet all requirements or at least lie within the allowed deviation range. The solutions are initially presented in tabular form (Figure 4.7). Each design solution is represented by one row of the table. The associated catalog components, solution principles and properties are listed in the columns of the table. The user can decide according to which performance property or rating function (NRMSD, PX) the solutions are ranked. A second property or rating function can be specified to further differentiate design solutions with identical values. To support the exploration of the design solutions, there are various tools, for example a search window and the possibility to show only entries in which there are differences between the design solutions.



Subcomponents

Motor	ILM85x13
Gearbox	CSD_32_100_2A
Brake	NoBrake
MotorController	ElmoTwitterGold
MotorControllerPosition	MCPTangentially
IncrementalEncoder	WithIE
HallSensor	NoHS
DriveBearing	SBN_61810_2RU
SlipRing	SNE01-24X
AbsoluteEncoder	MBA8_MRA8D080
SensorPCB	WithSPCB
IMU	WithIMU
TorqueSensor	StrainGaugeBased
OutputBearing	THK_RA_9008C
OutputBearingScrew	DIN7984_M5
DriveStructure	Linear
OutputStructure	Linear

Properties

NRMSD	0
Performance Index (PX)	1,4318
Power Density	470,4542 W/dm^3
Peak Torque T_p	182,25 Nm
Maximal Speed n_max	174 °/s
Length L	128,2 mm
Length Elektro	10 mm
Length Drive	56,2 mm
Length Output	62 mm
Diameter D	124 mm
Diameter Elektro	89 mm
Diameter Drive	91 mm
Diameter Output	124 mm
Height H	131,5 mm
Weight m	1,3699 kg

Figure 4.8.: Presentation of a single design solution.
Left: Conceptual drawing visualizing the arrangement of the components.
Right: Catalog components, solution principles and properties.

Double-clicking on a design solution of the table opens a new window in which the specific solution is presented to the user in detail (Figure 4.8): On the left side, a conceptual drawing is shown that visualizes the solution principles and arrangement of the catalog components. On the right side the catalog components, solution principles and properties are listed.

The specific design solutions as presented in Figure 4.8 can be exported as PDF. Another possible export for specific solutions is a text file that lists all subcomponents with data on the manufacturer, material, prices and weights. And finally, the working ontology that results from the reasoning process, can be saved as OWL file. It contains all design solutions.

4.5. Summary and Review

This chapter concluded the description of the approach that aims to preserve knowledge on robot design in order to support future developments. The result is an expert system that presents design solutions for robot components based on user requirements. It is composed of an ontological knowledge base, an inference engine and a user interface.

A novelty is the Multi-Stage Reasoning Process, which is used by the expert system. Similar to the work presented by Šaša Bastinos and Krisper (2013), it combines ontological reasoning with a graph-based approach. The difference, however, is that the graph is not only applied to a single individual that is evaluated according to several criteria. Instead, when traversing the graph, individuals are also combined to generate new individuals. The graph is the Multi-Stage Design Graph resulting from the formalization process. Thus, during Multi-Stage Reasoning, individuals representing catalog components or solution principles are combined to form new individuals representing partial solutions. These new individuals are then evaluated. If they meet the requirements they are further combined with other individuals until individuals are generated which represent design solutions for the overall system, the robot component. By filtering early on individuals that do not meet the user requirements, the runtime can be reduced. Since the Multi-Stage Design Graph considers multiple solution principles for important design parameters (such as different arrangements of subcomponents), this also applies to the expert system. In addition, the expert system has access to a large selection of catalog components whose data is stored in the ontological knowledge base.

By automatically selecting and arranging catalog components, the novel expert system framework can support large parts of the design process. Compared to related work, it avoids typical simplifications such as the use of modules instead of catalog components or the adaptation of only a single, parameterized solution. Its potentially high level of detail is demonstrated in the following evaluation chapter.

5. Evaluation

This chapter presents the evaluation of the approach presented in this work, which describes how expert knowledge on the design of humanoid robots can be preserved to support future developments. The evaluation is based on knowledge gained during the mechatronic design of the humanoid robot ARMAR-6. Specifically, two case studies will be conducted on the two components of ARMAR-6 that have the highest level of integration: sensor-actuator-controller (SAC) units and robotic hands (Figure 5.1).

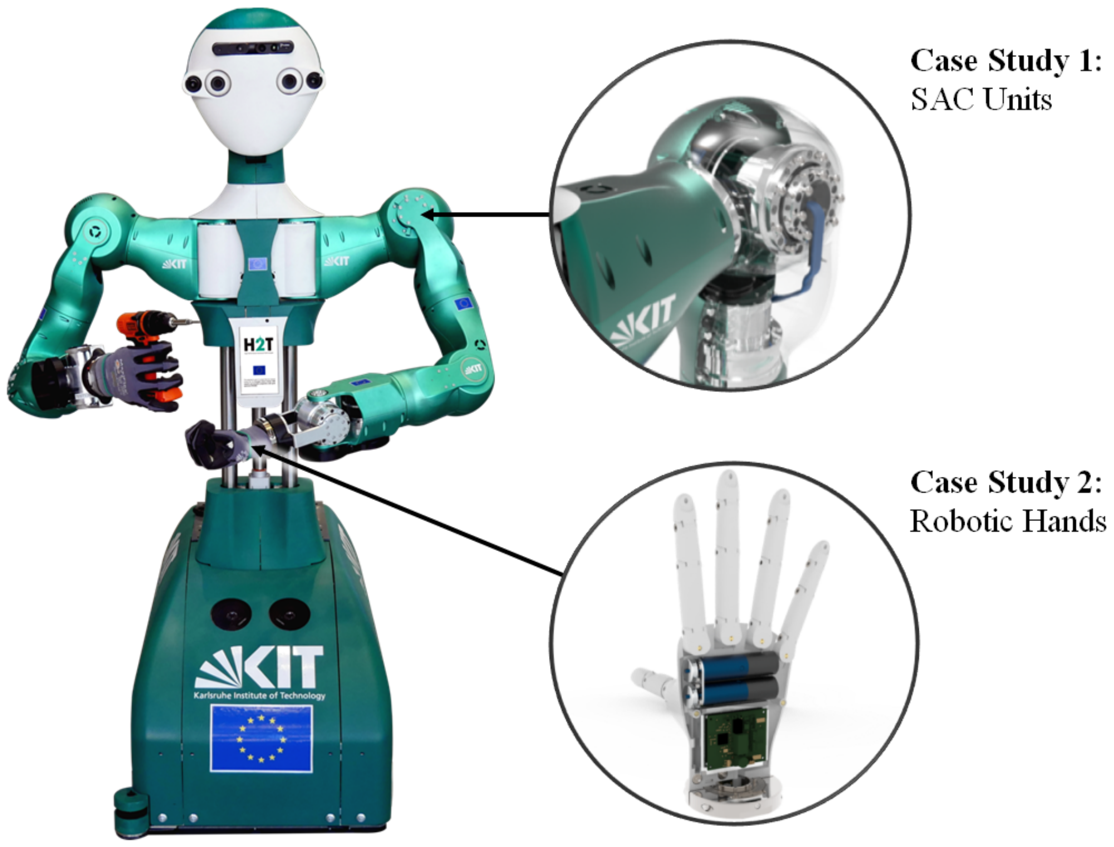


Figure 5.1.: The humanoid robot ARMAR-6: Its two components with the highest degree of integration, sensor-actuator-controller (SAC) units and robotic hands, serve as case studies.

After an introduction to ARMAR-6, the two case studies demonstrate how *design knowledge* gained through the design of robot components and analysis of related work can be *formalized* and ultimately made usable by an *expert system*. Finally, it is demonstrated that the resulting expert systems are capable of reproducing existing design solutions as well as generating optimized or novel design solutions.

5.1. The Humanoid Robot ARMAR-6

This section introduces the humanoid robot ARMAR-6, the youngest member of the ARMAR humanoid robot family (Asfour et al., 2019a). The development of hardware and software of ARMAR-6 as well as a validation of the entire system were presented in detail in Asfour et al. (2019b). The dual-arm system (Rader et al., 2016) and the sensor-actuator-controller units (Rader et al., 2017) were presented in further publications. Therefore this section is a summary of previous publications, in particular Asfour et al. (2019b), with a focus on the design of the mechatronic components of ARMAR-6.

5.1.1. Motivation

Developed as part of the *SecondHands* project, the vision in the development of ARMAR-6 was to create a humanoid robot that can not only collaborate with humans but even help them autonomously and proactively. In concrete terms, the ambitious scenario envisages that a robot assists a technician in the maintenance of conveyor belts in a highly automated warehouse.

5.1.2. Requirements

The scenario results in numerous requirements for the hardware.

To work collaboratively with a human, the robot must guarantee its safety. Known measures to achieve safe human-robot collaboration include lightweight design, anthropomorphic kinematics and high-speed torque control.

The requirement to implement cognitive abilities in software, which are necessary for the robot to help autonomously and proactively, also results in requirements for the hardware: The robot should be equipped with various exteroceptive sensors as well as sufficient on-board computer resources.

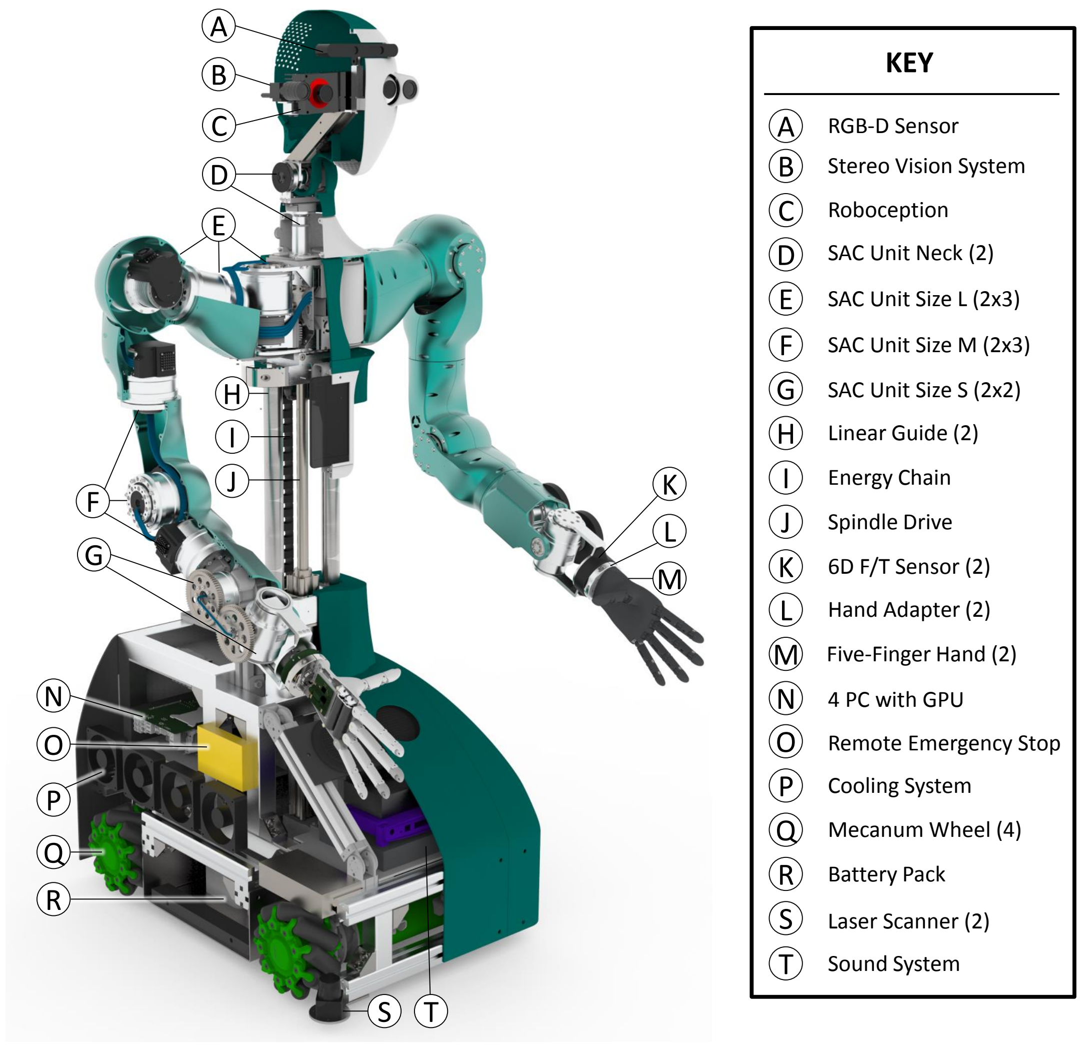


Figure 5.2.: Essential internal components (A-T) of ARMAR-6.
Adapted from *Source:* (Asfour et al., 2019b) © 2019 IEEE.

The use of the robot in a real warehouse also requires exceptionally high robustness and reliability. If the robot nevertheless needs to be repaired, a high degree of modularity should ensure that these repairs are easy to carry out.

The concrete scenario requires the mobile manipulation of human tools and other objects in confined spaces. This requires not only that the robot can move holonomically, but also that it can grasp human tools.

Furthermore, the robot should be able to pick up objects from the ground and at the same time hand over objects to a technician on a ladder at a height of 2.3 meters. The objects to be manipulated can weight up to 20 kg. As a result, a manipulation system with both a large workspace and a high payload capacity are needed.

Finally, the need to integrate this number of functionalities into a single autonomous system with a human-like appearance results in an very high level of system integration.

5.1.3. Robot Design

ARMAR-6 is a humanoid robot with a total of 28 active degrees of freedom (DoF). It has a height of 192 cm, a maximum arm span of over 300 cm and a weight of 160 kg (with battery packs up to 40 kg more). Figure 5.2 illustrates the essential components of the robot. At the highest level, the hardware of ARMAR-6 can be divided into five different subsystems, which can also be used separately due to their modularity: dual-arm system, robotic hands, torso, mobile platform and sensor head.

Dual-Arm System

The dual-arm system (Rader et al., 2016) consists of two identical high-performance robot arms with 8 DoF each. Table 5.1 enlists the Denavit-Hartenberg parameters of the arm as well as its joint limits.

Table 5.1.: Denavit-Hartenberg parameters, joint limits and sensor-actuator-controller (SAC) units of a single KIT Arm with 8 DoF.

Joint	θ [°]	α[°]	a [mm]	d [mm]	Joint limits	SAC unit (size)
Cla1	θ_1	75	0	0	±82°	Large (L)
Sho1	$-90+\theta_2$	90	0	300	$\pm\infty$	Large (L)
Sho2	$75+\theta_3$	90	0	0	-22°/+202°	Large (L)
Sho3	$-90+\theta_4$	90	-55	409	$\pm\infty$	Medium (M)
Elb1	$180+\theta_5$	90	0	0	-36°/+154°	Medium (M)
Elb2	$90+\theta_6$	90	0	364	$\pm\infty$	Medium (M)
Wri1	$90+\theta_7$	90	0	0	±40°	Small (S)
Wri2	θ_8	0	227	0	±90°	Small (S)

Each arm comprises one clavicle joint of the inner shoulder joint (*Cla1*), three intersecting shoulder joints emulating a spherical joint (*Sho1-3*), two intersecting elbow joints (*Elb1-2*) and two wrist joints (*Wri1-2*). The wrist joints intersect with *Elb2*, building a second spherical joint. This joint configuration is inspired by the work of Asfour (2003), who introduced a model of the human arm kinematics with 9 DoF. It was first realized with 8 DoF for the arms of the humanoid

ARMAR-4 (Asfour et al., 2013). The clavicle shoulder joint (*Cla1*), which is only included in few robotic arms, significantly increases the bimanual workspace of the dual-arm system (Rader et al., 2016). Combined with a reach of 1.3 meters and a large range of motion in most joints (Table 5.1), the dual-arm system has a massive total workspace of $8.4\,\mathrm{m}^3$ and a bimanual workspace of $4.9\,\mathrm{m}^3$.

To enable ARMAR-6 to carry objects weighing up to 20 kg, both arms have been designed to carry up to 11 kg at long range. The necessary mechanical power is provided by highly integrated sensor-actuator-controller (SAC) units in each joint (Rader et al., 2017). A SAC unit includes a motor, a gearbox, an IMU, sensors for torque, position and temperature measurement as well as embedded electronics for high-speed control and communication via EtherCAT. The design of the three different-sized SAC units is described in more detail in subsection 5.2.1.

In order to achieve both a lightweight design as well as high robustness, the structure of the arm follows an exoskeleton design approach: The SAC units are linked by a hollow structure made of high-strength aluminum that serves as load-bearing structure and cover. Due to the high concentration of the arm's most important subcomponents in the SAC units as well as well-defined interfaces, the actual arm assembly is greatly simplified: Mechanically the units are connected to the hollow structure by flanges and clamps, electrically the units are daisy-chained via three cables (EtherCAT bus, DC bus and emergency stop). The cables run inside the hollow structure, which protects them and further increases the robustness of the arm.

In addition to the sensors in the SAC units, both arms include a 6D force/torque sensor in the wrist, which supports the robot in perceiving its environment. The sensor is directly connected to a SCHUNK hand adapter, which allows to easily attach different robotic hands or other end effectors like the KIT Swiss Knife Gripper (Borràs et al., 2018) to the arm. The adapter offers both an electrical and mechanical interface according to DIN-ISO 9409-1-50-7-M6.

Robotic Hands

To operate in man-made environments, ARMAR-6 has two five-fingered hands. The hands have an anthropomorphic appearance and kinematics based on human proportions. The 14 finger joints per hand (two per thumb, three per finger) are driven by two motors. One motor drives the thumb, the other motor drives the four fingers. To actuate multiple joints simultaneously, a rope runs through each finger. The force distribution between the fingers is realized by

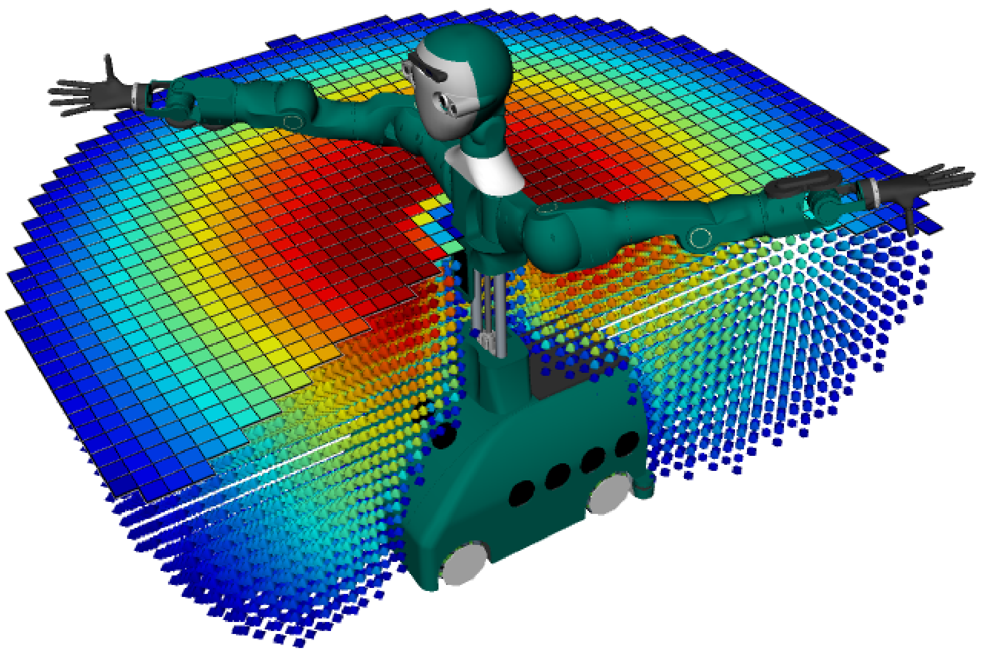


Figure 5.3.: Sectional view through the 10.7 m^3, nearly ellipsoidal workspace of the dual arm system incorporating the prismatic torso joint.
Source: (Asfour et al., 2019b) © 2019 IEEE.

a modified version of the TUAT/Karlsruhe mechanism (Fukaya et al., 2000, 2013). This adaptive underactuation mechanism makes it possible that the fingers can wrap around objects while automatically adapting to their shape. The robotic hand as well as its underactuation mechanism are described in more detail in subsection 5.3.1.

Torso

The torso of ARMAR-6 consists of a single prismatic joint, which allows a height adjustment of up to 40 cm. This results in an increase of the working space of the dual-arm system from 8.4 to 10.7 m^3. In addition, ARMAR-6 can manipulate objects both on the floor and at a height of 2.3 meters, as demanded by the requirements. The prismatic joint is driven by a DC motor whose rotation is converted into a translatory movement by a belt and a spindle. Two linear guides relieve the spindle and ensure robust force transmission between upper body and mobile platform. An energy chain serves the same purpose for the electronic connection between by protecting the cables and avoiding mechanical stress. To ensure that ARMAR-6 consumes as little energy as possible while keeping the height of the torso constant, a brake is installed. The absolute position of the torso is measured by a draw wire sensor.

Mobile Platform

To enable ARMAR-6 to move holonomically in confined spaces, the robot has an ominidirectional, mobile platform with four separately driven Mecanum wheels. Additionally, the platform offers space for all components necessary for the robot to operate autonomously and safely without external cables: four computers plus a GPU for on-board computing, network peripherals, a sound system, the power management system, battery packs, two different emergency stop systems and two laser scanners. All these internal components are hidden and protected by covers made of glass fibre reinforced plastics.

Sensor Head

The sensor head equips ARMAR-6 with five cameras, exteroceptive sensors, that enable it to perceive its environment visually. The five cameras are divided into three independent systems. The first camera system is a Roboception rc_visard 160, a stereo camera system with a baseline of 16 cm and on-board depth image processing. It is used for peripheral vision at distances between 0.5 and 3 meters. The second system is a pair of Point Grey Flea3 cameras, a second stereo vision system with a baseline of 27 cm for foveal vision. The third system is a Prime Sense Camine, an active red-green-blue-depth sensor (RGB-D) which is used for unicolored, featureless surfaces. The camera systems are mounted on a aluminum frame structure that is actuated by a pan-tilt unit (2 DoF) in the robot's neck. This pan-tilt unit is based on two sensor-actuator units, consisting of brushed DC motors, precise Harmonic Drives reduction gearboxes as well as incremental and absolute encoders. The units allows for a precise positioning of the cameras with angular speeds of up to 510 °/s.

5.1.4. Validation

To validate its hardware and software development, ARMAR-6 had to demonstrate its capabilities in a challenging demonstration scenario. This scenario required advanced scene understanding, human intention recognition, mobile manipulation and physical human-robot interaction (HRI). Derived from the objectives of the *SecondHands* project, ARMAR-6 assists a human technician in maintaining a conveyor belt in a warehouse.

The scenario starts with ARMAR-6 recognizing that the technician needs help removing a cover plate. It proactively offers its help and together, robot and

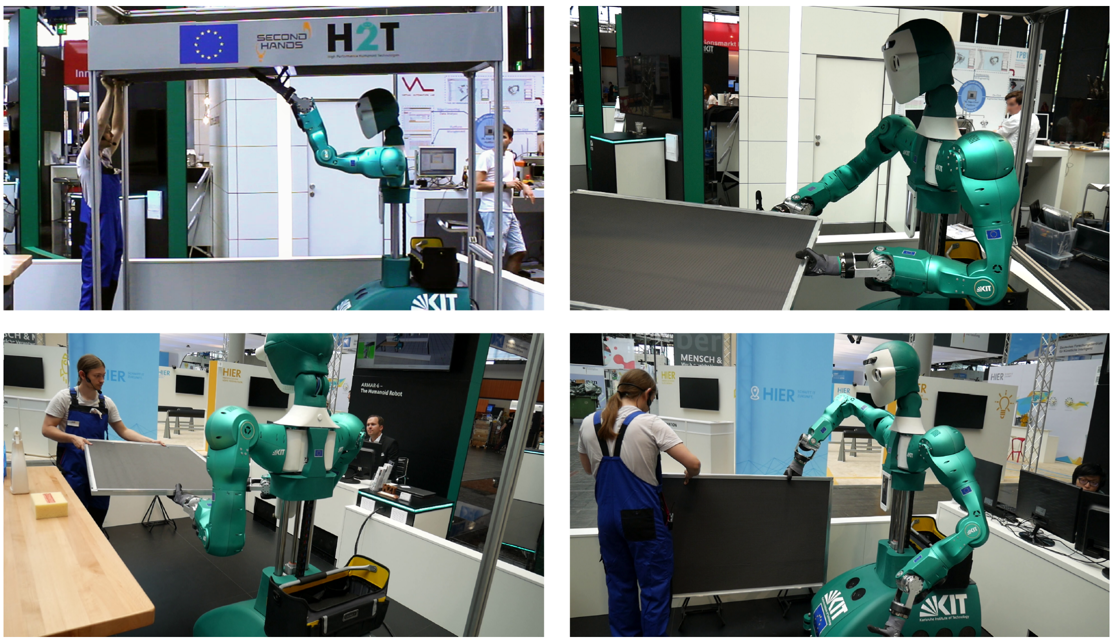


Figure 5.4.: The technician and ARMAR-6 remove the cover panel in a cooperative bimanual mobile manipulation task. *Source:* (Asfour et al., 2018) © 2018 IEEE.

technician carefully remove the cover plate, carry it to a suitable location specified by the technician and place it on the ground (Figure 5.4). Based on an initial diagnosis of the damage by the technician, ARMAR-6 picks up the appropriate tool from a workbench and hands it over to the technician, who may already have climbed a ladder in the meantime. When maintenance is complete, the robot receives the tool from the technician and places it on the workbench.

To successfully complete this scenario, ARMAR-6 must recognize the technician's need for help. For this purpose it uses various sensor modalities, including visual perception, haptic feedback and speech recognition. Human-robot interaction (HRI) is also of particular importance. In this context, ARMAR-6 is capable of estimating human poses and interacting with humans both verbally and physically.

ARMAR-6 has successfully completed the described demonstration scenario more than a hundred times with various technicians, including 20 professional maintenance technicians. However, as a humanoid robot with a wide range of capabilities, ARMAR-6 is not limited to a single scenario or use in warehouses. For a more detailed description of its software framework ArmarX (Vahrenkamp et al., 2015), its capabilities, and the demonstration scenario presented, please refer to Asfour et al. (2019b).

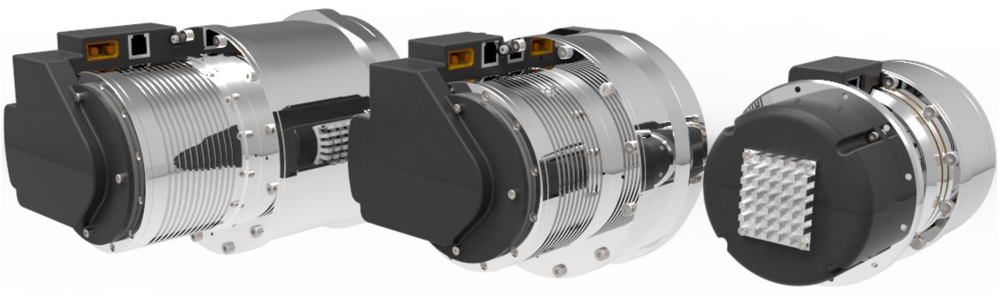


Figure 5.5.: The KIT sensor-actuator-controller (SAC) units in three different sizes
Source: (Rader et al., 2017) © 2017 IEEE.

5.2. Case Study 1: Sensor-Actuator Units

Sensor-actuator (SA) and sensor-actuator-controller (SAC) units for robot joints serve as a first case study to evaluate the approach. The case study starts with a presentation of the KIT SAC units (Figure 5.5), which are part of the dual-arm system of ARMAR-6. Thereafter, the knowledge derived from these designs and related work is formalized to serve as the knowledge base of an expert system for SA/SAC units. Finally, this expert system is used to reproduce and optimize existing SA/SAC units as well as to generate a novel design.

5.2.1. Design Knowledge

The main source of knowledge for this case study are the three KIT SAC units developed for the dual-arm system of the humanoid robot ARMAR-6. In the following, their design is described with a focus on the challenges of developing such a highly integrated mechatronic system. Furthermore, it is explained which designs from related work serve as additional sources of knowledge to expand the design knowledge. The description of the KIT SAC units was previously published in Rader et al. (2017).

KIT SAC Units: Mechatronic Design

Based on the experience gained from the development of the ARMAR robots, in particular ARMAR-4 (Asfour et al., 2013), and insights from related work, three highly integrated sensor-actuator-controller units were developed: the KIT SAC units. Theses units were originally designed for the dual-arm system of the humanoid robot ARMAR-6 (subsection 5.1.3). However, due to their high level of integration, modularity, robustness, encapsulation and scalability,

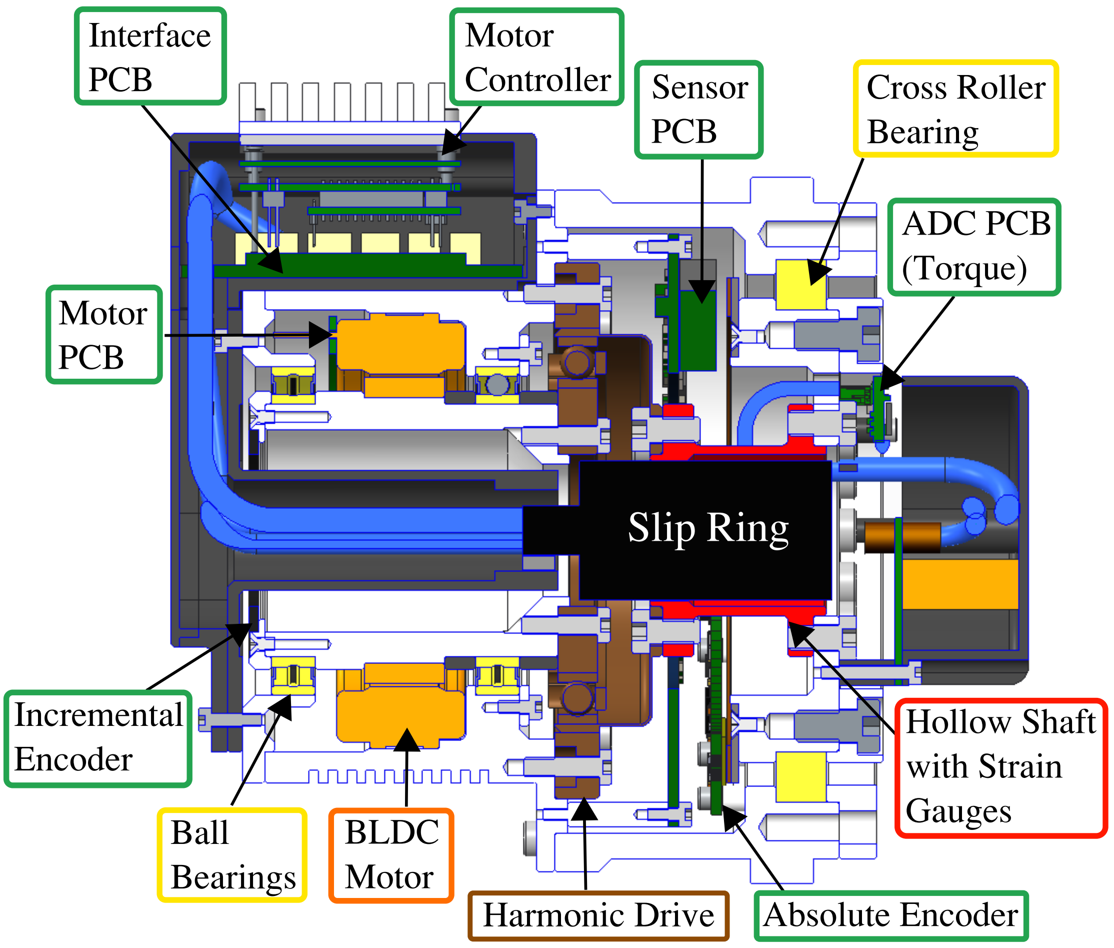


Figure 5.6.: Labeled cross section of the medium-sized KIT SAC unit. Cables are colored blue. *Source:* (Rader et al., 2017) © 2017 IEEE.

they can easily be used for other robotic systems. Table 2.5 in subsection 2.4.2 demonstrates their high level of integration compared to related work. There are no other rotary SA units that combine this high level of integration with high torques of up to 176 Nm. In the following the design of the units is described in more detail using the example of the medium-sized SAC unit (Figure 5.6).

Mechanical Design: The medium-sized SAC unit is driven by a brushless DC motor (ILM 70x10 RoboDrive), which is specifically designed for the use in robotics. It combines high torque density with low thermal losses and allows an easy integration into compact actuator units. The rotor sits on the motor shaft, which is supported by two bearings (SBN 61808 2RU 2Z), and drives the wave generator of the Harmonic Drive reduction gear unit (CSD-25-160-2A-GR-BB). Harmonic Drive units combine a high gear ratio with a compact and lightweight design. Furthermore, their lack of backlash facilitates precise position control of the actuator. The output of the Harmonic Drive, the flex spline, is linked to a hollow shaft with strain gauges for torque sensing. The hollow shaft is attached to the output flange, which is supported by a compact and rigid cross roller bearing (THK RA 7008C). A high-strength aluminum alloy was chosen as the material for the structural parts (white), as it combines low density with a high yield strength. Protective covers (dark gray) for the cables and electronic components are made from ABS plastic using 3D printing technologies.

Sensors: Each SAC unit provides a comprehensive sensor setup. For position sensing two magnetic encoders are integrated: An incremental encoder with 5760 events per motor revolution is directly connected to the motor driver, while a 20 Bit single turn absolute encoder (Renishaw Aksim MBA8) measures the absolute position of the output flange with an accuracy of $\pm 0.1^\circ$. For torque measurement the output hollow shaft was applied with four strain gauges, which are wired as H-bridge for temperature compensation. The analog signal is digitized using a differential ADC (Texas Instruments ADS1220). This setup allows very precise torque control which, in combination with a high control frequency, can be used to realize active compliance. Besides position and torque sensors, each SAC unit integrates a 9-axes absolute orientation sensing device (IMU) with on-board temperature sensing (Bosch BNO055) that is placed on the motor PCB. The comparatively large amount of sensory data is used in feedback control loops, and for in-depth system monitoring. All sensor data is available over the high-level bus interface.

Electronics: For communications the Ethernet-based EtherCAT (Ethernet for Control Automation Technology) is used, which offers real-time performance and a data rate of 100 Mbit/s. The high bandwidth allows for a high data throughput at high frequencies even on buses with many connected actuators. The two core components of the electrical architecture are the motor controller and the microcontroller for sensor data sampling. Both devices are directly connected to the EtherCAT bus and are slaves to the master PC. The motor controller is a compact industrial grade servo controller for current-, position- and velocity-control, allowing a maximum continuous current output of $10\,\mathrm{A}$ (ELMO Gold Twitter 10/100). It is placed on a specially designed PCB. The sampling microcontroller (Atmel ATmega1284P), located on the sensor PCB at the center of the SAC unit, has two main responsibilities: One of them is running the EtherCAT interface stack, including the state machine. The other task is periodically sampling all connected sensors (at $1\,\mathrm{khz}$) and maintain an up-to-date representation of all of their readings.

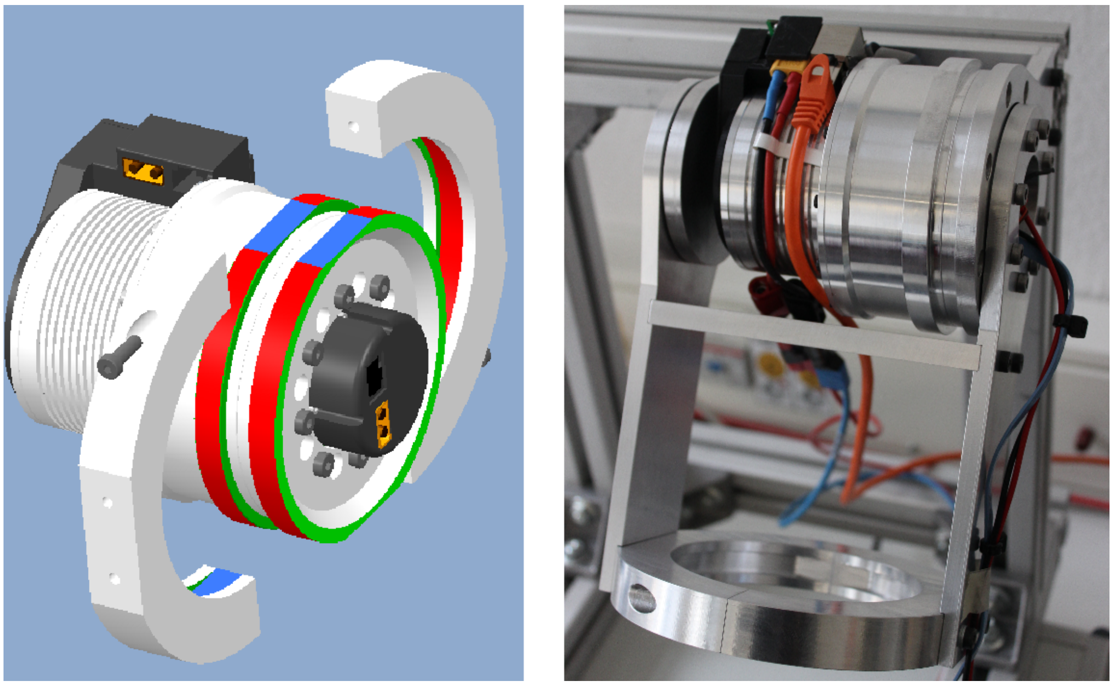


Figure 5.7.: Mechanical interfaces: Clamp ring interface highlighted in red (left); Flange interface used in an elbow joint (right). *Source:* (Rader et al., 2017) © 2017 IEEE.

Cabling: For a robust SA unit, a robust cabling strategy is crucial. Hollow shafts are a popular solution for routing cables between the input and output of a SA unit , but this does not prevent motion-induced stress. Furthermore, the maximal rotation of the actuator units is limited due to the cables. To overcome this problem, a slip ring is used (Senring SNE01-24X). Slip rings are based on the principle of transmitting power and electrical signals from brushes to metal rings. This allows a design in which cables can be fixed on the input side and still rotate infinitely on the output side

Interfaces: For the modularity and usability of SA units, well-defined mechanical and electronic interfaces are of major importance. Mechanically, the SAC units offer two alternative interfaces (Figure 5.7): While a clamping ring interface is suitable for roll joints such as forearm rotation, a screw flange interface at the output is intended for pitch joints such as elbow flexion/extension. Electrically there is a minimum of interfaces through which the SAC units can easily be daisy-chained. The input consists of a total of three connectors for the DC power bus (48 V), the communication bus (Ethernet) and a separate emergency stop (optional). For the output the same three connections are used.

Scaling: To take the different space and performance requirements of the individual robot joints into account, the KIT SAC units were designed in three

Table 5.2.: Specifications of the three KIT SAC units.

SAC Unit	Peak Torque [Nm]	Max. Speed [°/s]	Weight [kg]	Length [mm]	Diameter [mm]	Motor (ILM)	Gearbox Harmonic Drive
Small	56/64[1]	206	1.1	117	85	50x08	CSD-20-160-2A
Medium	123	131	1.8	113	112	70x10	CSD-25-160-2A
Large	176	79	2.1	159	112	70x18	CPL-25-160-2A

[1] Peak torque based on motor/gearbox

different sizes (Table 5.2). By using the same or similar subcomponents despite scaling, the design process can be drastically simplified. For the development of the large-sized SAC unit a Harmonic Drive CPL-25-2A unit was used. It has a higher peak torque than the shorter CSD-25-2A unit, while having the same diameter. As there also exist longer RoboDrive motors with the same diameter but higher nominal torques (ILM 70x18), the medium-sized SAC unit had only to be lengthened to get a unit with a higher torque capacity. As the diameter of the SAC unit did not have to be changed, each of the five PCB with all their subcomponents, all sensors and the bearings could be reused. Thus, the electrical setup for the large unit is identical to the setup described above. Furthermore, five out of the ten aluminum parts are exactly the same. The remaining five parts only differ in few parameters, mostly in length. This strategy does not only reduce the costs and time for design but also shortens the testing period. The third member of the SAC unit series is the small-sized SAC unit with a ILM 50x08 motor and a CSD-20-160-2A-GR-BB Harmonic Drive. In order to reduce the diameter compared to the medium-sized SAC unit (a critical dimension for most subcomponents), the reusability of the parts could not be achieved in the same way. However, either the same subcomponents or their smaller version and the same cabling concept were used. As shown in Table 5.2, the small-sized SAC unit roughly shares its length with the medium-sized SAC unit, whereas the medium-sized and the large-sized SAC unit share their diameter. The peak torques of the SAC units correspond to the limits for repeated peak torques of the Harmonic Drive units. Based on these limits, static analyses for every part are conducted to ensure a safety factor $S = 2$ against plastic deformation.

KIT SAC Units: Challenges

The main challenge in the mechatronic development of the SAC units was to achieve a high degree of integration on the one hand and to ensure physical

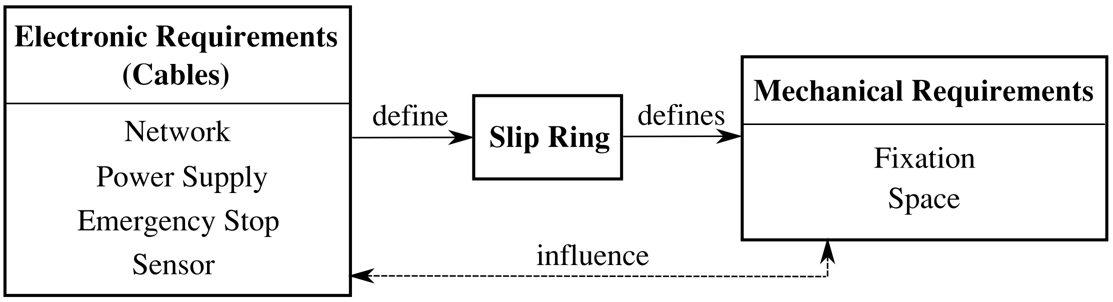


Figure 5.8.: Example of the electro-mechanical co-design: The slip ring has to fulfill electronic requirements and simultaneously it defines mechanical requirements. *Source:* (Rader et al., 2017) © 2017 IEEE.

feasibility and robustness on the other. A high level of integration can only be achieved if the design, selection and arrangement of the mechanical and electronic subcomponents are not performed separately but simultaneously. With regard to robustness, the complex cabling system resulting from the integration of so many sensors and other electronic subcomponents was of particular importance. As a result, an electro-mechanical co-design approach was pursued.

The electro-mechanical co-design started with first concepts for the mechanism and the electronic structure with a focus on the dependencies between both domains: installation space, fixation of all subcomponents and cabling. An example of a subcomponent that is highly dependent on both mechanics and electronics is the slip ring: On the one hand, it must provide all necessary cables for the output and the torque sensor, on the other hand, it requires sufficient installation space and is fixed by the output hollow shaft (Fig. 5.8).

Based on the first concepts, specific catalog components such as the motor, gearbox and sensors were chosen. Only after that, the exact electronic requirements could be specified and a slip ring could be chosen. Thereafter, structural parts such as the output shaft, which fixes the slip ring on one side, could be adjusted. Finally, each cable of the unit was inserted into the CAD model, considering each cable's bending radius and connectors. This led to further adjustments of the structural parts. In summary, it can be noted that the SAC unit ran through repeated iterations between mechanical and electronic design in its early design stage, especially because of the cabling concept. As a consequence, this exact model led to a highly integrated but still robust robotic component that could be assembled without unpleasant surprises.

For further details on the design and evaluation of the KIT SAC units please refer to Rader et al. (2017).

Table 5.3.: Sources from which knowledge on the design of SA units is gained.

Name	Source Type	Knowledge Acquisistion	Derived Solution Principles	Derived Components
KIT SAC unit L	Main source	Design, system analysis	All	All
KIT SAC unit M	Main source	Design, system analysis	All	All
KIT SAC unit S	Main source	Design, system analysis	All	All
ARMAR-4 SA unit leg	Supple-mentary	System analysis (CAD, tests, literature[1])	Compact drive, compact ouput, motor controller external, through bore cabling	No new data
WALK-MAN SEA (Size A)	Supple-mentary	System analysis (Literature[2])	Output two-sided, encoder-based torque sensor, Hall sensor	Kollmorgen motors

[1] (Asfour et al., 2013) 2 (Negrello et al., 2015)

Knowledge Sources

The presented KIT SAC units serve as the main sources of design knowledge that will be formalized in order to use it for an expert system. It was gained both through the actual design process and subsequent system analysis.

This knowledge was extended by analyzing the ARMAR-4 SA units (Asfour et al., 2013). Since the development of the SAC units is based on the design of the ARMAR-4 SA units, a part of the design knowledge overlaps. For example, they use many catalog components from the same manufacturers and series, including RoboDrive ILM motors, Harmonic Drives, AMS incremental encoders and THK cross roller bearings. The strain-gauge-based torque measurement is also similar. However, as Table 5.3 shows, the analysis of the ARMAR-4 SA units for the legs still allowed to extend the design knowledge by adding further solution principles. In particular, the ARMAR-4 SA units for the legs offer an alternative solution principle for arranging the components of the drive and output section in a compact way (Figure 5.9, left).

A further complementary source of knowledge is the IIT WALKMAN SEA in size A (Negrello et al., 2015). Besides adding Kollmorgen BLDC motors as an alternative to RoboDrive motors, this unit provides a new solution principle for arranging components. Thereby the output torque is transmitted back to the drive side via a long hollow shaft (Figure 5.9, right). A longer hollow shaft results in higher deflection under load, making it easier to measure torque with

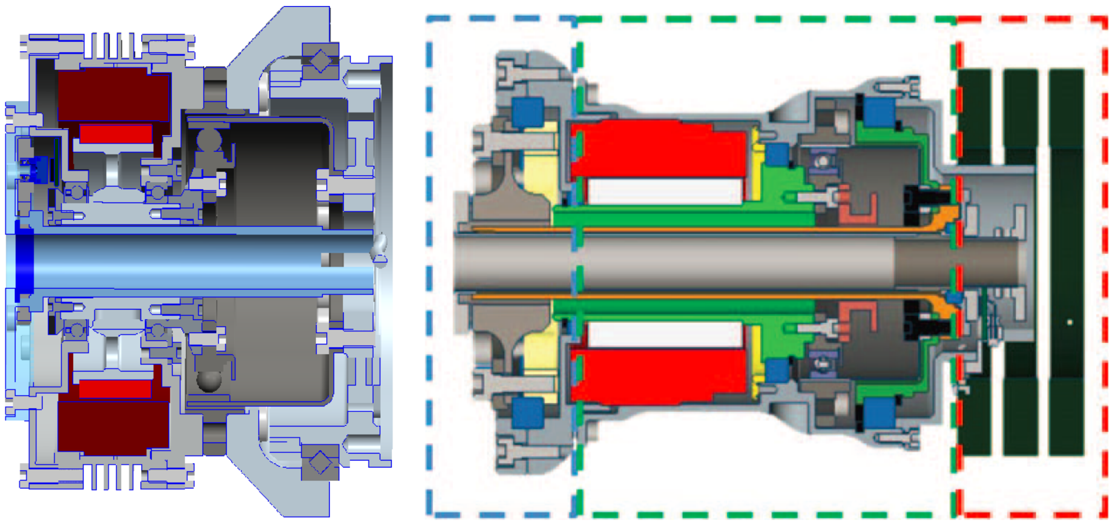


Figure 5.9.: Cross sections of SA units that serve as additional knowledge sources. Left: SA unit for the hip and legs of ARMAR-4 (Asfour et al., 2013). Right: SEA of WALK-MAN. *Source:* (Negrello et al., 2015) © 2015 IEEE.

two absolute encoders instead of strain gauges. In order to analyze this solution principle in more detail, it was re-modeled in CAD.

5.2.2. Formalization

Based on the design knowledge gained from the five SA/SAC units, specific models were created. By identifying the similarities and differences of the specific models, a generalized model for the design of SA units could be created, which is described below. First results on the generalized model of the SA units were presented in (Karrenbauer, Rader and Asfour, 2018).

Similarities and differences between robot components of the same type mainly result from the selected catalog components and solution principles. For SA units a total of 29 solution principles for 11 parameters (4 structural and 7 functional parameters) were identified.

Solution principles for structural parameters describe different possibilities for arranging catalog components and other subcomponents. They are depicted in Figure 5.10. In the sense of a morphological box, each SA unit is assigned exactly one solution principle for each of the four parameters *Drive Structure*, *Output Structure*, *Motor Controller* and *Cabling*. Each solution principle has different advantages and results in different spatial dimensions. For example, the drive bearings can be placed under (D2) instead of beside the motor (D1), resulting in a shorter length, but a smaller through bore for cabling.

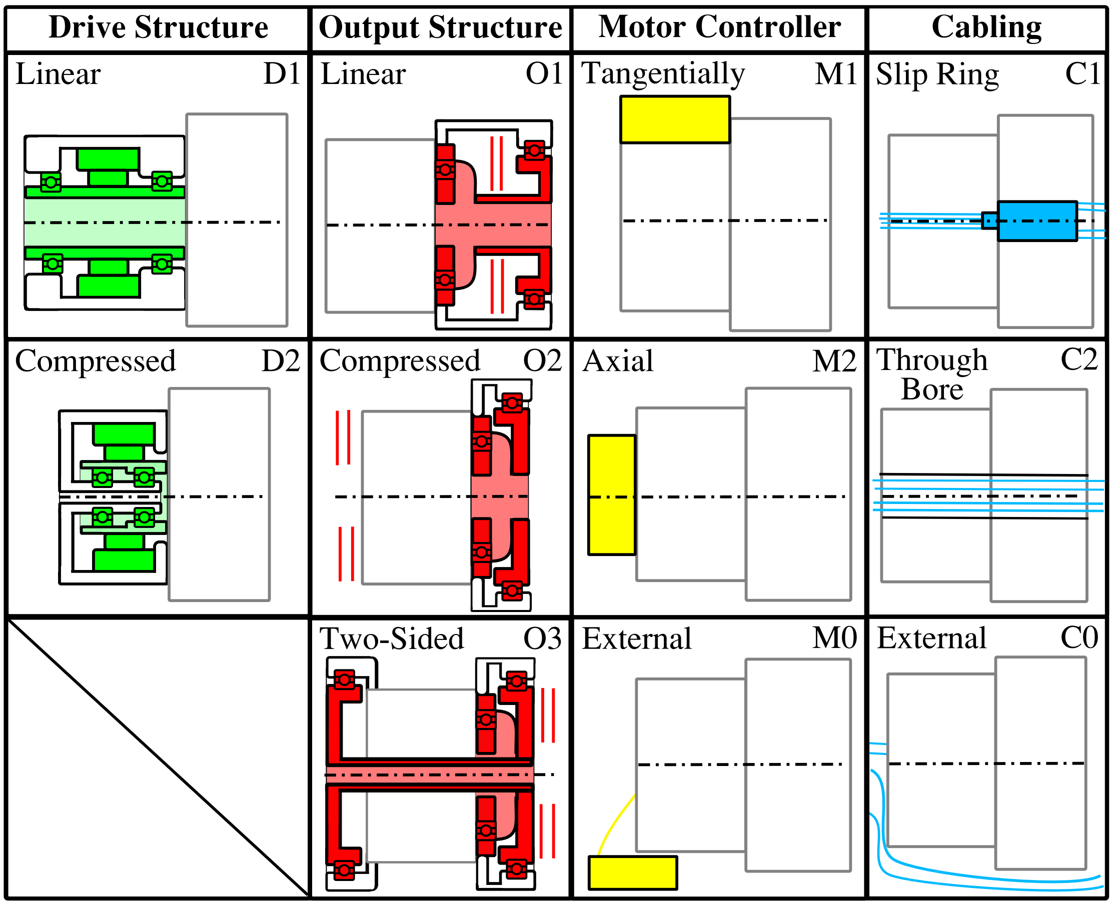


Figure 5.10.: Solution principles for the structural parameters of SA units.
Source: (Karrenbauer, Rader and Asfour, 2018) © 2018 IEEE.

Solution principles for functional parameters express how functions are realized and whether they are fulfilled at all. Table 5.4 represents the morphological box for the solution principles of the 7 functional parameters for SA units. While it is always an option not to realize a function at all (0), the other solution principles result in integrating additional catalog components or other subcomponents. The latter results in changed properties of the overall system. For example, the solution principle *Hide Cables Yes* (HC2) adds additional plastic covers to hide and protect cables and electronics, which increases the length, diameter, costs and weight of the SA unit. As with structural parameters, exactly one solution principle is chosen for each functional parameter. Thus it is also possible to classify solutions for SA units on a conceptual level by the chosen solution principles of the morphological boxes. For instance, the abbreviation (D1,O1,M1,C1,MF1,AE1,TS1,IM1,SB1,BR0,HC1) describes the design concept of the medium-sized KIT SAC unit (Figure 5.6).

Table 5.4.: Solution principles for functional parameters of SA units.

	Motor Feedback (MF)	**Absolute Encoder (AE)**	**Torque Sensor (TS)**	**IMU (IM)**	**Sensor PCB (SB)**	**Brake (BR)**	**Hide Cables (HC)**
0	None	None	None	None	None	None	No
1	Incremental Encoder	Yes	Strain-Gauge-Based	Yes	Yes	Yes	Yes
2	Hall Sensor		Encoder-Based				
3	Both		Both				

Apart from the solution principles, the selection of the catalog components is decisive for the properties of the SA unit. For the following 7 component types different catalog components are considered: *Motor, Gearbox, Brake, Drive Bearing, Output Bearing, Absolute Encoder* and *Slip Ring*. Important component types, of which only one catalog component is considered, are motor controllers, incremental encoders and the IMU. This can be explained by the fact that these components are not available in various sizes suitable for SA units.

The generalized model consists of a Multi-Stage Design Graph as well as rules and formulas that describe the transitions between the nodes. Figure 5.11 shows the Multi-Stage Design Graph for SA units. The sink node of the DAG represents the overall solution, the *SA unit* (dark violet). All catalog component types are represented by source nodes (dark violet). Solution principles are either represented by source nodes for their parameters (light violet) or different paths of the graph (D1, D2, O1, O2, O3). Start and end of the different paths are represented by light green nodes. All other nodes for partial solutions are shown in dark green. Table 5.5 lists what the abbreviations for the nodes of the partial solutions in Figure 5.11 stand for. As explained in subsection 3.3.3, each node has a counterpart node with the *Satisfied* prefix (white). To better illustrate the graph, these nodes are not represented in Figure 5.11, with a few exceptions for the *Motor-Gearbox Match*.

The Multi-Stage Design Graph for the SA units and all associated rules and formulas for the different solution principles are stored in a specific *HRC Design Ontology*: The *SA Unit Design Ontology*. This ontology is used by an expert system to support the design of SA units. As mentioned in subsection 3.4.2, other ontologies that are not robot component specific are also part of the knowledge base of the expert system, e.g. the *Component Ontology*.

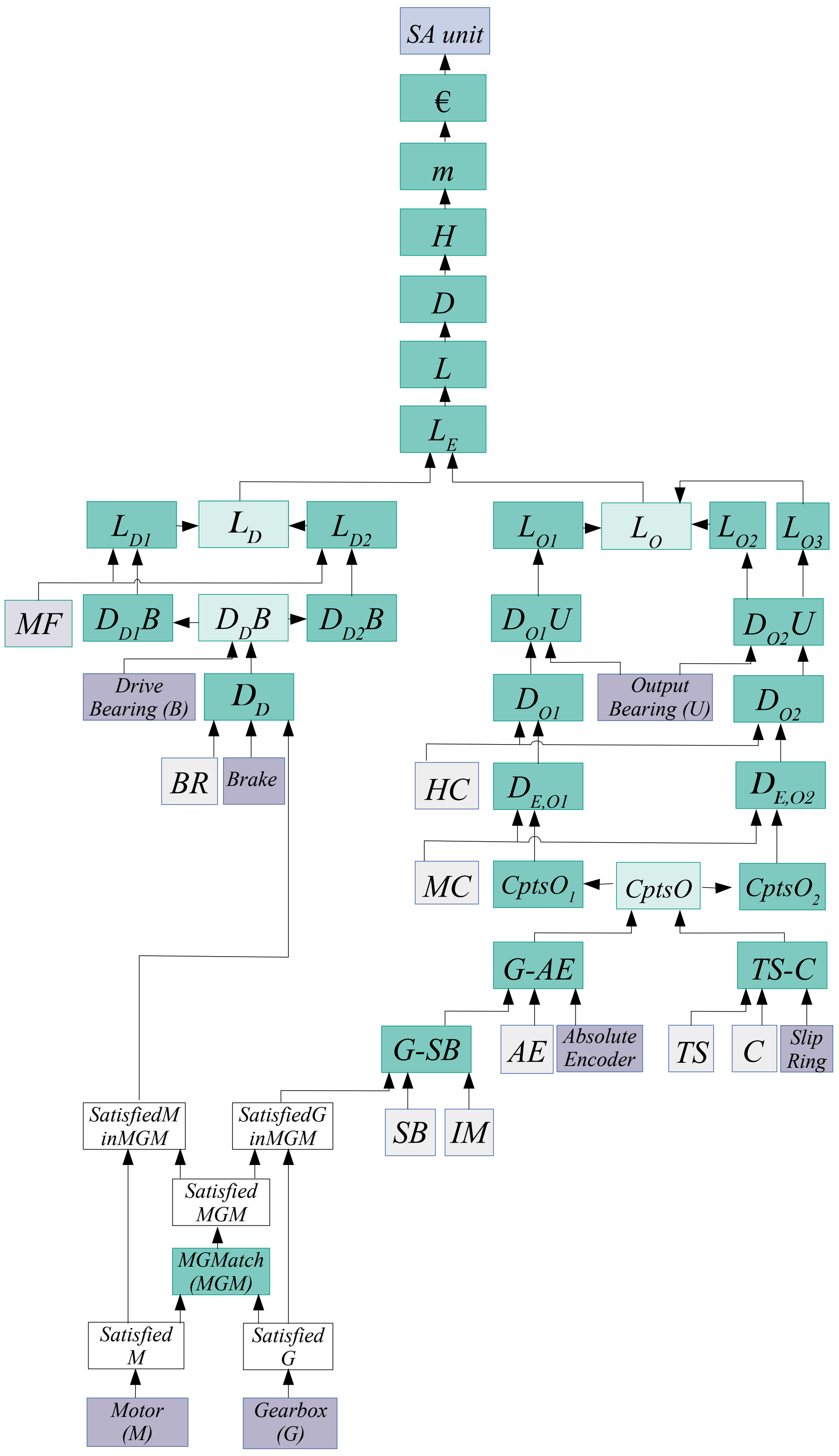


Figure 5.11.: Multi-Stage Design Graph for SA units.

Table 5.5.: Nodes of the Multi-Stage Design Graph for SA units.

Node	Description	Node	Description
€	Total costs of the SA unit	$G\text{-}SB$	Gearbox-Sensor-PCB Match
$CptsO$	Output-Components Match	H	Max. height of the SA unit
D	Max. diameter of the SA unit	L	Total length of the SA unit
D_D	Diameter of the drive section	L_D	Length of the drive section
D_DB	Diameter-Drive-Bearing Match	L_E	Length of the electronic section
$D_{E,O}$	Diameter of the electronic section	L_O	Length of the output section
D_O	Diameter of the output section	m	Total weight of the SA unit
D_OU	Diameter-Output-Bearing Match	MGM	Motor-Gearbox Match
$G\text{-}AE$	Gearbox-Absolute-Encoder Match	$TS\text{-}C$	Torque-Sensor-Cabling Match

5.2.3. Expert System

The ontological knowledge base resulting from the formalization process is used by an expert system for SA units. There follows a description of which requirements can be defined and how Multi-Stage Reasoning is performed by traversing the Multi-Stage Design Graph (Figure 5.11). First results on the expert system for SA units were presented in (Karrenbauer, Rader and Asfour, 2018).

Requirements Definition

To use the expert system, requirements for the SA unit must first be entered via the user interface. Table 5.6 lists which functional, performance and interface requirements can be defined. Most of the functional requirements can be mapped directly to functional parameters. In the case of functional parameters with more than two solution principles, the user can decide whether the function has to be fulfilled by any arbitrary means or by a specific solution principle. The performance requirements are defined by a minimum and/or maximum value. Allowed deviations and their weighting for rating functions are defined in the optimization tab (subsection 4.4.2). The interface requirements consider the electrical input and output.

Besides the *requirements*, the user also has the possibility to restrict the generation of design solutions by additional *constraints*. For example, specific solution principles for structural parameters can be defined. In expert mode it is also possible to *adjust parameters* of the formulas via UI: the motor-gearbox combination can be dimensioned for other load cases, materials for structural parts can be freely selected and costs can be adapted to current supplier offers.

Table 5.6.: Requirements, constraints and adjustable parameters of the expert system.

Requirements	
Functional	BR, MF, AE, TS, HC, SB, IM, continuous rotation (C)
Performance	Peak torque, speed, length, diameter, height, weight, costs
Interface	Input voltage, output current, other output cables (communication, emergency stop)
Constraints	
Structural Parameters	Drive Structure (D), Output Structure (O), Motor Controller Position (MC)
Motor-Gearbox Match	Do (not) allow small gearboxes with big motors
Adjustable Parameters (Expert Mode)	
Motor-Gearbox Match	Peak torque of motor (percentage of nominal or max. torque), peak torque of gearbox (repeated or momentary), use efficiency of gearbox and other devices
Structure	Wall thickness of drive and output structure
Material	Percentage of materials for structural parts (aluminum/ magnesium/steel alloys or test material with chosen density)
Costs	Costs of the structural parts, cover and cables

Based on the defined requirements and constraints as well as the ontological knowledge base, the Multi-Stage-Reasoning process for SA units is executed. It starts with the *Motor-Gearbox Matching*. Each of the equations presented in the following is represented in the knowledge base by SWRL rules, as described in subsection 3.4.3.

Motor-Gearbox Matching

The goal of this step is to find motor-gearbox combinations, which fulfill the performance requirements, in particular the maximum speed $n_{Req,max}$ and the peak torque $T_{Req,p}$. At first, both devices, the $Motors(M)$ and the $Gearboxes(G)$, are filtered separately and assigned to their satisfied counterpart *SatisfiedM* and *SatisfiedG* (Figure 5.12). Therefore, the mechanical limits of the gearbox, i.e. the maximum output speed $n_{G,max}$ and the peak torque $T_{G,p}$, have to satisfy the requirements:

$$n_{Req,max} \leq n_{G,max} \tag{5.1}$$

$$T_{Req,p} \leq T_{G,p} \tag{5.2}$$

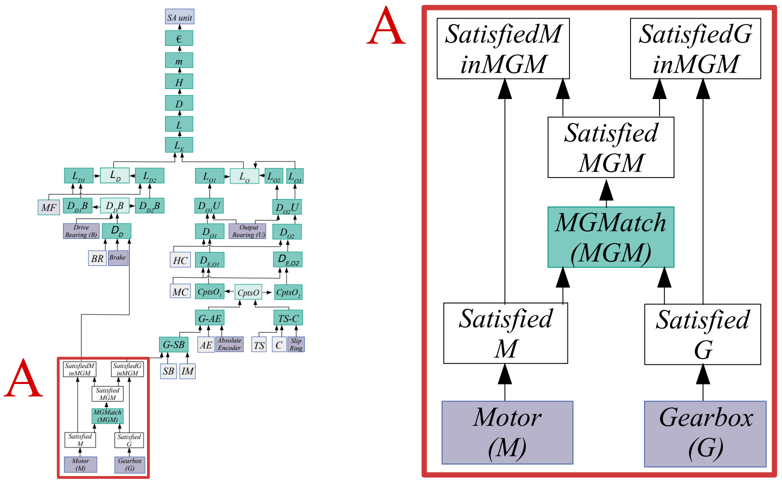


Figure 5.12.: Motor-Gearbox Matching (A) as part of the Multi-Stage Design Graph.

The filtering of the motors with a maximum speed $n_{M,max}$ and a peak torque $T_{M,p}$ can be described as follows:

$$n_{Req,max} \leq \frac{n_{M,max}}{i} = n_{Mi,max} \tag{5.3}$$

$$T_{Req,p} \leq T_{M,p}\ \eta_G\ \eta_{other}\ i = T_{Mi,p} \tag{5.4}$$

The motor filter considers all possible gearbox reductions i at an efficiency of the gearbox $\eta_G = 1$. For this purpose, the output speed $n_{Mi,max}$ and the peak torque $T_{Mi,p}$ of the motor at gear ratio i are calculated. After filtering both devices separately, the expert system checks each $MotorGearboxMatch(MGM)$:

$$n_{Req,max} \leq n_{Sol,max} = \min(n_{G,max}, n_{Mi,max}) \tag{5.5}$$

$$T_{Req,p} \leq T_{Sol,p} = \min(T_{G,p}, T_{Mi,p}) \tag{5.6}$$

In contrast to the motor filter, the efficiency $\eta_G < 1$ and ratio i of the respective gearbox are used. The result of the motor-gearbox matching is a set of $SatisfiedMGMs$ with performance parameters ($n_{Sol,max}, T_{Sol,p}$), which fulfill or surpass the requirements ($n_{Req,max}, T_{Req,p}$). It should be noted that the explanation of filtering only based on torques and speeds is exemplary. In order to be assigned to their satisfied counterpart, *Motors*, *Gearboxes* and *MotorGeraboxMatches* must further satisfy the other performance requirements: length, diameter, costs and weight. Since no catalog component or partial solution may be larger, heavier or more expensive than the overall system, rough filtering is carried out early on.

Selection of Other Catalog Components

Similar to the described motor-gearbox matching, the remaining catalog components, such as sensors and bearings, are selected in different nodes of the Multi-Stage Design Graph, based on rules described by mathematical expressions. Based on the requirements, a catalog component setup is determined for each motor-gearbox match and each solution principle for a structural parameter. For example, for each motor-gearbox match the system chooses two different drive bearing setups as two solution principles for the drive structure are implemented (D1, D2). Besides construction space and performance requirements, electrical requirements like the cabling of sensors are considered. For example, slip rings are chosen based on electrical requirements of the rotating sensors and following SA units, i.e. the necessary power supply and data lines for a bus. However, as a result of the slip ring choice, the gearboxes and structural options are filtered based on the necessary construction space.

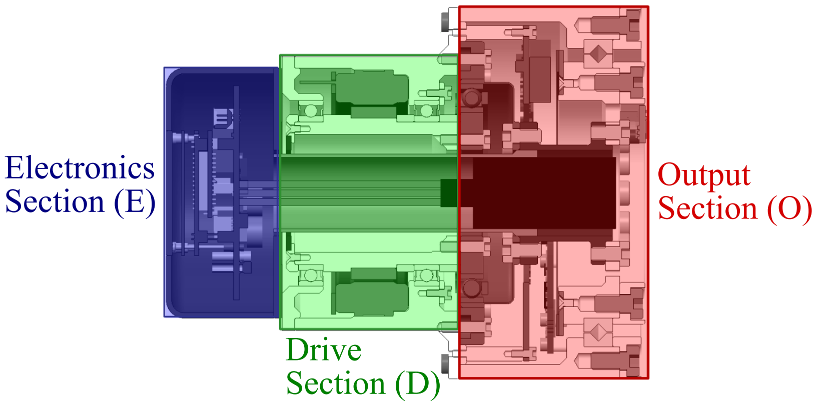


Figure 5.13.: The three sections of SA units.

Sections and Paths

Based on the selection of catalog components and solution principles, the spatial dimensions of the three cylindrical sections of the SA unit (Figure 5.13) can be determined: drive (D_D, L_D), output (D_O, L_O) and electronics (D_E, L_E). The structural parameters for the drive structure and the output structure, which determine the arrangement of the subcomponents, have a great influence on these spatial dimensions. Therefore they are modeled in the graph as different paths to which different rules are assigned.

Equation 5.7 and 5.8 present the calculation of the drive section length L_D for both drive structure options simplified for a better understanding by summarizing comparatively short lengths with $\epsilon_1, \epsilon_2, \epsilon_3$. As L_D mostly depends on the

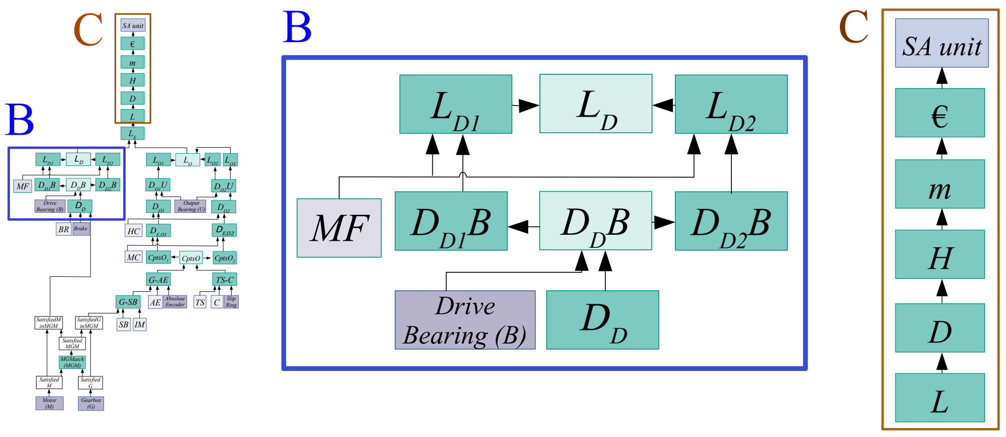


Figure 5.14.: Different paths (B) and final nodes (C) of the Multi-Stage Design Graph.

lengths of the motor-encoder-brake unit (L_{MHB}) and the drive bearings (L_{LB}, L_{FB}), the parallel arrangement (D2) results in a shorter length than the linear arrangement (D1). This is reflected in the equations by using the maximum instead of the sum. The rules described by Equation 5.7 and 5.8 are evaluated in the correspondent nodes L_{D1} and L_{D2} (B in Figure 5.14). They are based on the Diameter-Drive-Bearing Matches $D_{D1}B$ and $D_{D2}B$, in which the best fitting $DriveBearing(B)$ is selected depending on the solution principle (D1 and D2).

$$L_{D1} = L_{MHB} + L_{LB} + L_{FB} + \epsilon_1 \tag{5.7}$$
$$L_{D2} = \max(L_{MHB} + \epsilon_2,\ L_{LB} + L_{FB} + \epsilon_3) \tag{5.8}$$

Analogous to L_D, the remaining diameters and lengths of the three sections are determined for all solution principles of the structural parameters. Thereafter, the system determines the total length L and maximum diameter D of each possible SA unit combination (C in Figure 5.14):

$$L = L_D + L_O + L_E \tag{5.9}$$
$$D = \max(D_D, D_O, D_E) \tag{5.10}$$

Due to the placement of the motor controller (height H_{MC}) tangentially to the drive section (D_D), solution principle M1 requires the calculation of the SA unit's height H:

$$H = max\Big(\frac{D}{2} + \frac{D_D}{2} + H_{MC}, D\Big) \tag{5.11}$$

Until now, the weight of the SA unit m could only be calculated roughly as the sum of all catalog components $m_{CatalogComp}$. Now that the dimensions and all solution principles are known, cables m_{Cables}, covers m_{Covers} and structural parts $m_{Structure}$ can also be taken into account. Since structural parts for SA units are usually rotary parts, their volume $V_{Structure}$ is approximated by the volume of q hollow cylinders $V_{HollowCylinder,j}$. For each solution principle of a structural parameter a different parameterized formula set is used that depends on the dimensions of the catalog components and their interfaces. The thickness of the hollow cylinder parts depends mainly on the wall thicknesses t_w selected for the drive and output section. The density of the structural parts $\rho_{Structure}$ depends on the material, which the user can choose freely (Table 5.6). The weight for covers m_{Covers} is a constant and only considered if the user wishes to hide cables (H1).

$$m = \sum_{i=1}^{n} m_{CatalogComp} + m_{Structure} + m_{Covers} + m_{Cables} \tag{5.12}$$

$$m_{Structure} = \rho_{Structure} \cdot V_{Structure} = \rho_{Structure} \cdot \sum_{j=1}^{q} V_{HollowCylinder,j} \tag{5.13}$$

Finally, the net costs c of the overall system are calculated in euros (€). The costs of the structural parts $c_{Structure}$, covers c_{Covers} and cables c_{Cables} are based on the corresponding costs of the KIT SAC units. These default values can be adjusted via the UI.

$$c = \sum_{i=1}^{n} c_{CatalogComp} + c_{Structure} + c_{Covers} + c_{Cables} \tag{5.14}$$

The *SA units* that satisfy all requirements are presented to the user via the UI.

Scope of the System

The ontological knowledge base of the SA unit expert system comprises 980 OWL classes, 115 object properties, 216 data properties and 273 SWRL rules. Part of this knowledge base is the Component Ontology with approximately 500 catalog components, including 288 gearboxes and 68 motors. Taking into account compatibilities, there are 2162 valid motor-gearbox matches for SA units. They are suitable for a torque capacity range from 1.8 to $823\,\mathrm{Nm}$. Taking all 29 solution principles of the 11 parameters into account as well as other catalog components, the expert system is able to generate more than 100 million valid solutions for SA units. The exact calculation and an overview of the Component Ontology can be found in Appendix A and B.

5.2.4. Reproduction

In the following, the KIT SA/SAC units and state-of-the-art SA units are reproduced by the expert system. By reproducing the KIT SA/SAC units as accurately as possible, it can be demonstrated that the formalization process to create an expert system for SA units based on design knowledge was successful. This serves to validate the first part of the research question of this thesis, *how knowledge on robot design can be preserved*. The reproduction of further state-of-the-art SA units serves to evaluate the generalized model by showing how comprehensively it considers different SA units.

The properties of the SA units to be reproduced serve as performance requirements: spatial dimensions (length L, diameter D, height H), mechanical performance (peak torque T_p, maximum speed n_{max}) and weight m. Capabilities of the existing units, i.e. torque sensing, are specified by functional requirements. Whenever solution principles of the real units are known, they are specified by constraints. Otherwise, the expert system will consider any option to realize the capability. To measure how well solutions generated by the expert system reproduce existing units, the *Normalized Root-Mean-Square Deviation* (*NRMSD*) is calculated (subsection 4.4.2). The *NRMSD* takes only negative deviations from the required performance properties into account. In the case of SA units, negative deviations are larger spatial dimensions, a higher weight and lower mechanical performance. The smaller the *NRMSD*, the better the solution. If *NRMSD* $= 0$ there are no negative deviations, i.e. all performance requirements are fulfilled or surpassed. To rank solutions with the same *NRMSD*, the *Performance Index* (*PX*) is used as second criterion, which considers both positive and negative deviations. If *PX* $= 1$, the properties correspond exactly to the requirements. If *PX* > 1 the positive deviations outweigh, for *PX* < 1 it is the opposite.

Reproduction of the KIT SA/SAC Units

Table 5.7 lists results for the reproduction of the three KIT SAC units and the ARMAR-4 SA unit. There are two rows for each SA/SAC unit: The first row shows the requirements (Req) derived from the properties of the real SA units. The second row presents the properties of the best solution (Sol) with the least negative deviation, i.e. the smallest possible NRMSD. As demonstrated by the low *NRMSD* < 0.05 and the comparison between performance requirements and properties of the solutions, the KIT SA/SAC units could be reproduced

Table 5.7.: Reproduction of the KIT SA/SAC units.

			Performance Requirements / Properties							**Rating**	
SA Unit	**Size**		**L [mm]**	**D [mm]**	**H [mm]**	**T_p [Nm]**	**n_{max} [°/s]**	**m [kg]**	**Costs [€]**	**NRMSD**	**PX**
KIT SAC units[1]	L	Req	159	112	118	176	79	2.1			
		Sol	153	106	114	176	79	2.1	6151	0.000	1.02
	M	Req	113	112	118	123	131	1.8			
		Sol	114	103	112	123	131	1.7	4966	0.003	1.03
	S	Req	117	85	90	56	206	1.1			
		Sol	111	86	86	56	206	1.0	5039	0.005	1.03
ARMAR-4 SA units[2]	Leg	Req	84	112	112	157	174	1.4			
		Sol	83	113	113	157	174	1.5	4248	0.042	0.98

		Parameters for solution principles		**Catalog Comp.**	
Size		**Structural**	**Functional**	**Motor**	**Gearbox**
L	Req	D1 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1		
	Sol	D1 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x18	CPL25-160
M	Req	D1 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1		
	Sol	D1 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x10	CSD25-160
S	Req	D1 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1		
	Sol	D1 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	50x08	CSD20-160
Leg	Req	D2 O2 M0 C2	MF1 AE1 TS1 SB0 IM0 BR0 HC0		
	Sol	D2 O2 M0 C2	MF1 AE1 TS1 SB0 IM0 BR0 HC0	85x13	CPL25-100

		Catalog Components			
Size		**Slip Ring**	**Absolute Encoder**	**Drive Bearing**	**Output Bearing**
L	Req				
	Sol	AC7195	Aksim MBA/MRA 8	61808 2RU 2Z	THK RA 7008C
M	Req				
	Sol	SNE01-24X	Aksim MBA/MRA 8	61808 2RU 2Z	THK RA 7008C
S	Req				
	Sol	SRA-73805	Aksim MBA/MRA 7	61706 2RU 2Z	THK RA 5008C
Leg	Req				
	Sol	-	Aksim MBA/MRA 8	61807 2RU 2Z	THK RA 8008C

Abbreviations: Req = Requirements, Sol = Best solution (minimum $NRMSD$)
Settings for motor-gearbox matching: $T_{M,p} = 0.5\ T_{M,max}$ and $T_{G,p} = T_{G,repeated}$
Settings for material and structure: 100% aluminum alloys, wall thickness $t_w = 4$
Rating ($NRMSD$, PX): L, D, H, T_p, n_{max}, m are weighted equally ($w_i = 1$)
[1](Rader et al., 2017) [2](Asfour et al., 2013)

very well by the expert system. Torque and speed correspond exactly to the specifications, spatial dimensions and weight vary only minimally. The solutions tend to be slightly better ($PX > 1$), as some minor potential for improvement was identified and taken into account during the implementation of the expert system. The catalog components suggested by the expert system correspond exactly to the components of the existing units (subsection 5.2.1). Due to the successful reproduction it can be concluded that the design knowledge of the KIT SA/SAC units was successfully preserved and implemented.

Table 5.8.: Reproduction of state-of-the-art SA units for humanoid robotics.

			Performance Requirements / Properties							**Rating**	
SA Unit	**Size**		**L [mm]**	**D [mm]**	**H [mm]**	**T_p [Nm]**	n_{max} [°/s]	**m [kg]**	**Costs [€]**	**NRMSD**	**PX**
WALK-MAN SEA[3]	A	Req	150	110	110	270	84	2.0			
		Sol	147	113	113	299	109	2.1	4866	0.031	1.05
	B	Req	140	100	100	140	100	1.5			
		Sol	136	103	103	147	131	1.6	4669	0.039	1.04
	C	Req	100	60	60	56	68	0.7			
		Sol	86	80	80	54	210	0.9	2955	0.232	1.20
NREC Drive Joint[4]	NGT 200	Req	135	140	140	660	65	5.2			
		Sol	157	133	133	630	65	4.7	5495	0.068	1.00
	NGT 100	Req	131	112	112	360	89	3.0			
		Sol	137	113	113	304	85	2.6	4376	0.070	0.97
	NGT 50	Req	114	95	95	175	163	2.2			
		Sol	121	103	103	147	174	2.1	4368	0.088	0.95
	NGT 20	Req	91	77	77	50	182	1.0			
		Sol	108	80	80	54	210	1.1	3833	0.092	0.98
ANYdrive[5]	-	Req	95	90	120	40	684	1.0			
		Sol	89	93	113	41	548	1.1	4812	0.097	0.97
Gear Motor RD-HD[6]	50 x08	Req	78	97	97	28	330	0.9			
		Sol	77	80	80	28	480	0.6	2929	0.000	1.19
	70 x10	Req	100	128	128	92	132	2.6			
		Sol	83	113	113	96	348	1.4	3664	0.000	1.42
	85 x13	Req	111	141	141	176	108	3.7			
		Sol	99	133	133	278	195	3.1	3763	0.000	1.30

Abbreviations: Req = Requirements, Sol = Best solution (minimum $NRMSD$)
Rating ($NRMSD$, PX): L, D, H, T_p, n_{max}, m are weighted equally ($w_i = 1$)
See Appendix D for detailed settings and solution principles.
[3](Negrello et al., 2015) [4](Stentz et al., 2015) [5](ANYbotics, 2017) [6](TQ-Systems, 2016)

Reproduction of State-of-the-Art SA Units

Table 5.8 shows the results for the reproduction of 11 further state-of-the-art SA units for humanoid robots. In comparison to the KIT SA/SAC units, not all solution principles and catalog components are known for these units, so that various options are taken into account to realize capabilities or to arrange subcomponents. This also applies in part to the *WALK-MAN SEA* in size *A*, through whose analysis catalog component series (Kollmorgan motors) and new solution principles could be derived, but for which not all details are known.

However, Table 5.8 shows that the best solutions for 10 out of 11 SA units are very close to the real units ($NRMSD < 0.1$). The only negative exception is the reproduction of *WALK-MAN SEA* in size *C* with $NRMSD = 0.232$. The negative deviations result in particular from the higher diameter. As the volume increases with the diameter, the weight also increases, resulting in an overall higher NRMSD. To enable the expert system to better reproduce this unit in the future, a new solution principle for the output structure would have to be implemented, which selects and arranges the subcomponents at the output differently. In contrast, the *WALK-MAN SEA* in size *A* and *B* can be reproduced very well ($0.03 < NRMSD < 0.04$). The solutions for the *ANYdrive* and *NREC Drive Joints* offer small deviations ($0.06 < NRMSD < 0.10$), which can be explained by the use of different catalog components. For example, the *NREC Drive Joints* use smaller brakes, which results in a reduced total length. The solutions for the RoboDrive *Gear Motor RD-HD* units surpass the requirements ($NRMSD = 0$, $PX > 1$). The system uses the construction space to integrate motor-gearbox combinations with higher performance.

Reproduction Summary

In total, 15 SA units for humanoid robots were reproduced: 5 were used to derive the design knowledge for the expert system, the other 10 were unknown state-of-the-art units. 14 out of 15 SA units could be reproduced close to the real units ($NRMSD < 0.1$), including all 5 (partially) known and 9 unknown units. The generalized model of SA units has thus proven to be very precise, but can be slightly improved by new catalog components and solution principles.

5.2.5. Optimization

The reproduction of SA units demonstrated that the approach presented in this work is an answer to the first part of the research question (*How can expert* *knowledge on robot design be preserved?*). In the following it will be shown that the approach is also suitable for answering the second part of the research question (*How can it support future developments?*) by optimizing designs for SA units.

Most product development tasks are not original designs, but based on existing designs that are adapted or to which variants are designed (subsection 2.1.1). In this case, the three KIT SAC units serve as a basis for the expert system to find new, optimized solutions. For the same or a smaller installation space they will be optimized according to the different performance requirements (L, D, H, T_p, n_{max}, m, $Costs$) and the rating functions ($NRMSD$, PX). The allowed negative deviations relative to the existing SAC units are 5% for the spatial dimensions and 20% for the other performance requirements. The expert system is free to choose how capabilities are realized and how the subcomponents are arranged in the drive and output section. Considering the structural parts, default parameters are used for material (100% aluminum) and wall thickness ($t_w = 2$).

In the following, for each SAC unit a table is presented with the best optimizations results by all performance requirements and rating functions. If several solutions for the requirement to be optimized have the same result, PX is used as a second criterion for ranking. For comparison, the properties and rating functions for the reproduced SAC units (subsection 5.2.4) are given in the 1st column. Improvements of performance parameters are indicated in dependence on them, as smaller changes are more clearly visible. However, the performance requirements are derived from the properties of the existing SAC units, which are slightly different (Table 5.7).

Optimization of the KIT SAC Unit L

The optimized solutions differ from the existing unit by using other catalog components and solution principles. In the following, the changes for the different optimizations for SAC unit L (Table 5.9) are discussed:

- **Length (-10%):** The decrease can be achieved by a shorter motor from Kollmorgan (TBM 7615) and a more compressed arrangement of the subcomponents of the drive structure (D2).
- **Diameter (±0%), Height (-7%):** The diameter cannot be reduced due to the high torque requirements. However, the height can be reduced by placing the motor controller axially (M2) instead of tangentially (M1).
- **Peak Torque (±0%):** The unit is already torque-optimized.

Table 5.9.: Optimization of the KIT SAC unit L.

KIT SAC Unit L	**Performance Requirements / Properties**							**Rating**	
Optimization	**L [mm]**	**D [mm]**	**H [mm]**	**T_p [Nm]**	**n_{max} [°/s]**	**m [kg]**	**Costs [€]**	**NRMSD**	**PX**
Reproduced Unit L	**153**	**106**	**114**	**176**	**79**	**2.1**	**6151**	0.000	1.02
Length L	**137**	106	115	148	201	1.7	5725	0.061	1.27
Diam. D, Height H	150	**106**	**106**	148	201	1.7	5725	0.061	1.27
Peak Torque T_p	137	106	115	**176**	151	1.7	5725	0.000	1.20
Max. Speed n_{max}	137	106	115	148	**201**	1.7	5725	0.061	1.27
Weight m	137	113	115	153	131	**1.5**	6088	0.050	1.14
Costs	144	106	115	148	201	1.7	**5361**	0.061	1.27
NRMSD (e)	137	106	115	176	151	1.7	5725	**0.000**	1.20
PX, NRMSD (e)	137	106	115	148	201	1.7	5725	0.061	**1.27**

	Parameters for solution principles		**Catalog Comp.**	
Opt.	**Structural**	**Functional**	**Motor**	**Gearbox**
Rep.	D1 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x18	CPL25-160
L	D2 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CPL25-120
D,H	D2 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CPL25-120
T_p	D2 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CPL25-160
n_{max}	D2 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CPL25-120
m	D2 O2 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x10	CPL25-160
€	D2 O1 M1 C1	MF1 AE1 TS2 SB1 IM1 BR0 HC1	7615	CPL25-120
e	D2 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CPL25-160
PX, e	D2 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CPL25-120

Rating ($NRMSD$, PX): L, D, H, T_p, n_{max}, m, € are weighted equally ($w_i = 1$)

- **Max. Speed (+158%):** By selecting a faster motor (TBM 7615) and a lower gear ratio (i=120 instead of 160), the speed can be massively increased.
- **Weight (-29%):** The lower weight results from a more compressed sub-component arrangement (D2, O2), a lighter motor (ILM 70x10) and thinner walls for the structural parts ($t_w = 2$ mm instead of $t_w = 4$ mm).
- **Costs (-13%):** By using a cheaper motor (TBM 7615) and a solution principle to measure torque with encoders (TS2), the costs can be optimized.
- **NRMSD**: With the TBM 7615 motor, a thinner wall thickness ($t_w = 2$) and the compressed drive structure (D2), all properties of the unit can be improved with the exception of T_p and H.

- **PX**: The result corresponds to the optimization in length and speed. A drawback is that the increase in speed is at the expense of torque (-16%).

Optimization of the KIT SAC Unit M

Table 5.10.: Optimization of the KIT SAC unit M.

KIT SAC Unit M	Performance Requirements / Properties							Rating	
Optimization	L [mm]	D [mm]	H [mm]	T_p [Nm]	n_{max} [°/s]	m [kg]	Costs [€]	NRMSD	PX
Reproduced Unit M	**114**	**103**	**112**	**123**	**131**	**1.7**	**4966**	**0.003**	**1.03**
Length L	**81**	113	119	110	242	1.5	4577	0.040	1.18
Diam. D, Height H	118	**103**	**103**	110	242	1.6	4590	0.043	1.16
Peak Torque T_p	99	113	119	**176**	151	1.6	4728	0.004	1.13
Max. Speed n_{max}	81	113	119	110	**242**	1.5	4577	0.040	1.18
Weight m	81	113	115	123	131	**1.4**	4953	0.003	1.08
Costs	108	103	114	110	242	1.6	**4226**	0.040	1.17
NRMSD	108	103	114	123	151	1.6	4226	**0.000**	1.08
PX, NRMSD	81	113	119	110	242	1.5	4577	0.040	**1.18**

	Parameters for solution principles		Catalog Comp.	
Opt.	Structural	Functional	Motor	Gearbox
Rep.	D1 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x10	CSD25-160
L	D2 O2 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CSD25-100
D, H	D2 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CSD25-100
T_p	D2 O2 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CPL25-160
n_{max}	D2 O2 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CSD25-100
m	D2 O2 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x10	CSD25-160
€	D2 O1 M1 C1	MF1 AE1 TS2 SB1 IM1 BR0 HC1	7615	CSD25-100
e	D2 O1 M1 C1	MF1 AE1 TS2 SB1 IM1 BR0 HC1	7615	CSD25-160
PX, e	D2 O2 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	7615	CSD25-100

Rating ($NRMSD$, PX): L, D, H, T_p, n_{max}, m, € are weighted equally ($w_i = 1$)

The optimization results for SAC unit M (Table 5.10) are as follows:

- **Length (-29%):** This shortening results from a more compact arrangement of the subcomponents of the drive (D2) and Output (O2) structure.
- **Diameter (±0%), Height (-8%):** While the diameter cannot be further optimized, a reduction of the height is possible by axial placement of the motor controller (M2). However, this slightly increases the length.

- **Peak Torque (+43%):** By using a more powerful motor (TBM 7615) and gearbox (CPL25-160), the same peak torque can be achieved as with the large SAC unit L. The integration of these longer subcomponents is only possible through compact arrangement (D2, O2).
- **Max. Speed (+77%):** A more powerful motor (TBM 7615) and a lower transmission ratio (i=100) result in a higher speed.
- **Weight (-18%):** In contrast to the other optimizations of SAC unit M, the unit keeps its lightweight motor. Weight is saved by more compact arrangement (D2, O2) and thinner walls of the structural parts ($t_w = 2$).
- **Costs (-15%):** Costs are saved by the cheaper motor (TBM 7615) and encoder-based torque measurement (TS2).
- **NRMSD**: Without negative deviations hardly any optimizations can be made. Small improvements can be achieved by choosing the TBM 7615 motor, a compact drive structure (D2), encoder-based torque measurement (TS2) and a reduced wall thickness ($t_w = 2$).
- **PX**: If optimized according to all criteria, the result corresponds to the optimization in length and speed: The TBM 7615 motor in combination with a lower gear ratio (i=100) increases the speed dramatically. The unit becomes shorter due to the more compact arrangement of the subcomponents (D2,O2). In addition, the wall thickness is also reduced ($t_w = 2$). A disadvantage is that diameter and height are slightly increased and the peak torque decreases slightly.

Optimization of the KIT SAC Unit S

For the SAC unit S (Table 5.11) the optimizations result in the following changes of catalog components, solution principles and other settings:

- **Length (-13%):** The length is shortened by changing the solution principles for two structural parameters: First, the subcomponents of the drive section are arranged more compactly (D2), and second, the motor controller is placed tangentially (M1) instead of axially.
- **Diameter (-5%), Height (-5%):** Diameter and height can be reduced by a lower wall thickness ($t_w = 2$). Apart from that they are already optimized.
- **Peak Torque (+55%):** By using the motor of the middle-sized unit (ILM 70x10) and a stronger gearbox (CPL20-120) the torque can be increased significantly. The additional space required results from a more compact arrangement of the subcomponents of the drive section (D2).

Table 5.11.: Optimization of the KIT SAC unit S.

KIT SAC Unit S	**Performance Requirements / Properties**							**Rating**	
Optimization	L **[mm]**	D **[mm]**	H **[mm]**	T_p **[Nm]**	n_{max} **[°/s]**	m **[kg]**	**Costs [€]**	**NRMSD**	**PX**
Reproduced Unit S	**111**	**86**	**86**	**56**	**206**	**1.0**	**5039**	**0.005**	**1.03**
Length L	**97**	82	90	56	206	0.9	5039	0.000	1.06
Diam. D, Height H	121	**82**	**82**	49	420	1.1	5242	0.052	1.14
Peak Torque T_p	121	82	82	**87**	175	1.1	5242	0.061	1.07
Max. Speed n_{max}	121	82	82	49	**420**	1.1	5242	0.052	1.14
Weight m	100	82	82	56	206	**0.9**	5039	0.000	1.07
Costs	121	82	90	45	180	1.2	**3936**	0.091	0.98
NRMSD	106	82	82	56	206	0.9	4611	**0.000**	1.07
PX, NRMSD	121	82	82	49	420	1.1	5242	0.052	**1.14**

	Parameters for solution principles		**Catalog Comp.**	
Opt.	**Structural**	**Functional**	**Motor**	**Gearbox**
Rep.	D1 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	50x08	CSD20-160
L	D2 O1 M1 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	50x08	CSD20-160
D, H	D2 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x10	CPL20-50
T_p	D2 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x10	CPL20-120
n_{max}	D2 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x10	CPL20-50
m	D2 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	50x08	CSD20-160
€	D2 O1 M1 C1	MF1 AE1 TS2 SB1 IM1 BR0 HC1	50x14	XSF50-100
e	D2 O1 M2 C1	MF1 AE1 TS2 SB1 IM1 BR0 HC1	50x08	CSD20-160
PX, *e*	D2 O1 M2 C1	MF1 AE1 TS1 SB1 IM1 BR0 HC1	70x10	CPL20-50

Rating ($NRMSD$, PX): L, D, H, T_p, n_{max}, m, € are weighted equally ($w_i = 1$)

- **Max. Speed (+104%):** This solution corresponds to the optimization for peak torque, with the exception of a significantly lower gear ratio of i=50. This reduces the peak torque slightly, but the speed is more than doubled.
- **Weight (-10%):** Weight can only be saved minimally through a compact subcomponent arrangement (D2) and reduced wall thickness ($t_w = 2$).
- **Costs (-22%):** This optimization is the most interesting, since it results from a concatenation of dependencies: By using encoders for cheap torque measurement (T2), the slip ring must have fewer wires than for strain-gauge-based measurement. This allows the use of a smaller slip ring (SRA-7306) that is compatible with more gears, including the low-cost

HAINA XSF-50-2-100. To achieve the peak torque despite a gear ratio of i=100, a stronger motor is required (ILM 50x14). Since these catalog components are larger, the motor controller must be placed tangentially (M1) and the components of the drive must be arranged more compactly (D2).

- **NRMSD**: Without negative deviations hardly any optimizations can be made. Small improvements can be achieved by a compact drive structure (D2), a reduced wall thickness ($t_w = 2$) and encoder-based torque measurement (TS2), which results in a smaller slip ring (SRA-7306).
- **PX**: The solution for the overall optimization according to all criteria corresponds to the results for the optimization according to speed and diameter: A larger motor (ILM 70X10) and a stronger gearbox (CPL20-50) with a much lower gear ratio. The necessary space results from a more compact arrangement (D2). The wall thickness is also reduced ($t_w = 2$). However, this solution slightly worsens the length, torque, costs and weight.

Optimization Summary

The optimization demonstrated how changing catalog components and solution principles can lead to significantly different properties of the SAC units. The most common changes were a compressed arrangement of the drive structure components (D2) and thinner walls for the SAC units. Depending on whether length (O2) or diameter (O1) was to be optimized, different arrangements for the subcomponents of the output structure were chosen. The solution principles for placing the motor controller offer the possibility to reduce the length (M1) or the height (M2). Encoder-based torque measurement (TS2) was mainly used to save costs compared to the strain-gauge-based solution principle (TS1). With its combination of high speed and torque, the Kollmorgan TBM 7615 motor has shown that it is suitable for optimizing actuators M and L, making it an interesting alternative to the RoboDrive ILM motors. However, the ILM motors were still preferred when optimizing the weight.

Of all the performance parameters, the speed could be optimized best. The large optimization potential compared to other parameters may be surprising, but can be explained as follows: In contrast to increased torque, a higher speed requirement does not result in a larger gearbox, but only a lower gear ratio and a larger motor. However, since motors of SA units are usually much smaller than the gearboxes, their dimensions do not have such a large influence on the overall dimensions of the SA unit. In addition, gearboxes with lower ratios also have a better efficiency, which means less mechanical power is lost.

The cost optimization for the small-sized SAC units demonstrated that the expert system also considers complex chains of dependencies for the selection of catalog components and solution principles. This is an example of a design solution generated by the expert system that a human expert would not necessarily have come up with. How difficult it is for a human to systematically find an optimal design solution for SA units is shown by the large number of possibilities. Taking all requirements into account, several hundred valid solutions were generated for each SAC unit (L:304/M:199/S:187).

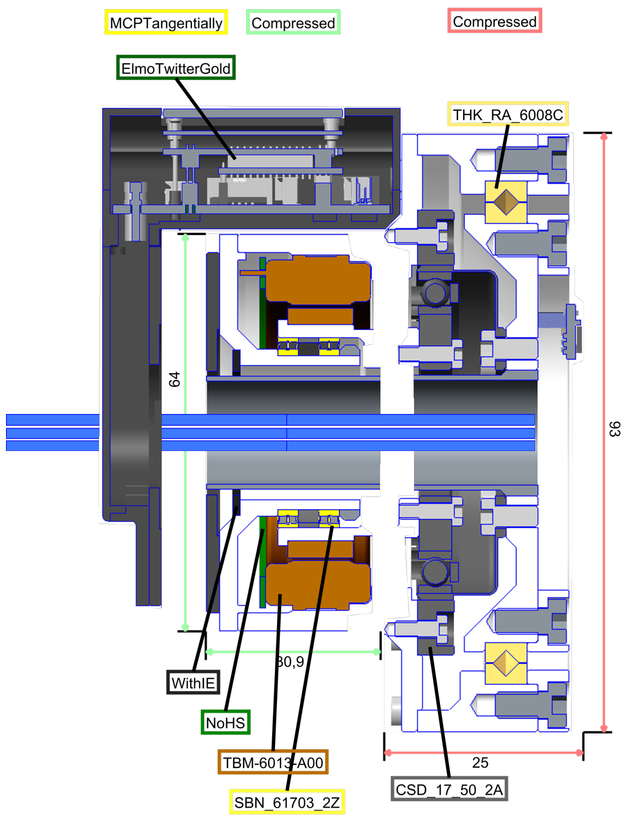
5.2.6. Novel Design

To demonstrate that the expert system is not only able to optimize existing solutions but also to generate solutions for novel designs, the development of a joint for elbow flexion/extension is shown as an example. The elbow joint should be designed so that it can be used for a humanoid robot with human-like proportions and capabilities. For this purpose the following requirements are defined:

- **Spatial dimensions:** To ensure that the elbow unit fits into an anthropomorphic arm, it must not be longer than **L=75 mm**. The diameter should not exceed **D=90 mm**. However, a slight increase on one side is possible (**H=110 mm**).
- **Peak torque:** The robot must be able to carry a load of 5 kg. The lever arm between the elbow axis and the load in the hand (or the Toolcenter point TCP) is assumed to be 30 cm (14.7 Nm). The weight of forearm and hand together is estimated to be 3 kg, with the center of gravity at half of the lever arm (4.4 Nm). This results in a necessary torque of 19.1 Nm, which, however, is estimated to be about **22 Nm**, taking into account dynamics.
- **Max. Speed:** The joint should be able to move at a speed of **400 °/s**.
- **Weight:** In order to make the arm not much heavier than a human arm, the weight of the SA unit should be less than **1 kg**.
- **Costs:** The net costs should not exceed **5000 €**.
- **Capabilities:** The functional requirements correspond to those of the SAC units, whereby the capabilities can be implemented as desired. However, no slip ring is required and the electronics do not have to be hidden.

Allowed negative deviations for all requirements are 5%. Based on these inputs, the expert system generates 8 solutions. The best solution (first criterion

SensorActuatorUnit



Subcomponents

Motor	TBM-6013-A00
Gearbox	CSD_17_50_2A
Brake	NoBrake
MotorController	ElmoTwitterGold
MotorControllerPosition	MCPTangentially
IncrementalEncoder	WithIE
HallSensor	NoHS
DriveBearing	SBN_61703_2Z
SlipRing	NoSlipRing
GearboxWaveGeneratorScrew	DIN7984_M3
GearboxCircularSplineScrew	DIN7984_M3
GearboxFlexSplineScrew	DIN7984_M4
AbsoluteEncoder	MBA7_MRA7D049AB025E00
SensorPCB	WithSPCB
IMU	WithIMU
TorqueSensor	StrainGaugeBased_SpokeWheel
OutputBearing	THK_RA_6008C
OutputBearingScrew	DIN7984_M5
DriveStructure	Compressed
OutputStructure	Compressed

Properties

NRMSD	0,0126
Performance Index (PX)	1,0862
Power Density	643,6423 W/dm^3
Peak Torque T_p	23 Nm
Maximal Speed n_max	516 °/s
Length L	74,5 mm
Length Electro	18,6 mm
Length Drive	30,9 mm
Length Output	25 mm
Diameter D	93 mm
Diameter Electro	60 mm
Diameter Drive	64 mm
Diameter Output	93 mm
Height H	100,5 mm
Volume Drive Structure	24,0852 cm^3
Volume Output Structure	98,2175 cm^3
Weight m	0,9567 kg
Costs c	4178,17 €

Figure 5.15.: Novel design for an elbow joint (PDF export of the expert system).

NRMSD, second PX) is shown in Figure 5.15. With the exception of the diameter, which is exceeded by 3%, the solution meets all requirements. This solution combines the design knowledge derived from the different sources: The compact arrangement of the ARMAR-4 SA units (D2,O2) is combined with the tangential placement of the motor controller (M1) and the electronic setup of the SAC units. The motor (TBM 6013) is from the same manufacturer (Kollmorgen) as the motor of the WALK-MAN SEA. Combined with a Harmonic Drive with low gear ratio (i=50), the speed requirement can even be exceeded (516 °/s).

Along with the optimization, this generation of a novel design serves to demonstrate that the approach presented in this thesis can also be an answer to the second part of the research question (*How can it support future developments?*).

5.3. Case Study 2: Robotic and Prosthetic Hands

To demonstrate that the approach presented in this thesis can be applied not only to SA/SAC units but also to other humanoid robotic components, a second case study is conducted on robotic and prosthetic hands. The main source is the knowledge gained through the design of the hands of ARMAR-6 (*KIT Robotic Hand V2*) and the *KIT Prosthetic Hand V2*.

This section presents the derived design knowledge, the formalization process and the resulting expert system for robotic and prosthetic hands. The expert system is then evaluated by reproducing, optimizing, and scaling the most current KIT Hands. Finally, the system is used to generate a novel design.

5.3.1. Design Knowledge

After a presentation of the co-development of robotic and prosthetic hands at KIT (Figure 5.16), the design knowledge that serves as a basis for the formalization process will be outlined.

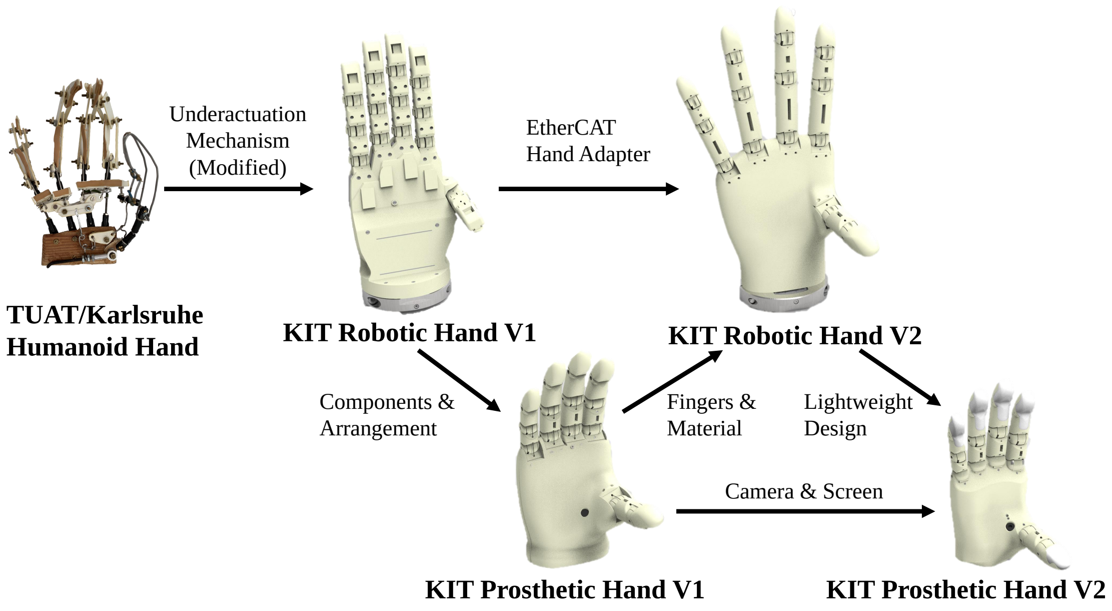


Figure 5.16.: The KIT Hands based on the original TUAT/Karlsruhe Humanoid Hand.

The KIT Hands: Co-Development of Robotic and Prosthetic Hands

As shown in Figure 5.16, several robotic and prosthetic hands have been developed at KIT in recent years. Thereby a co-development of robotic and pros-

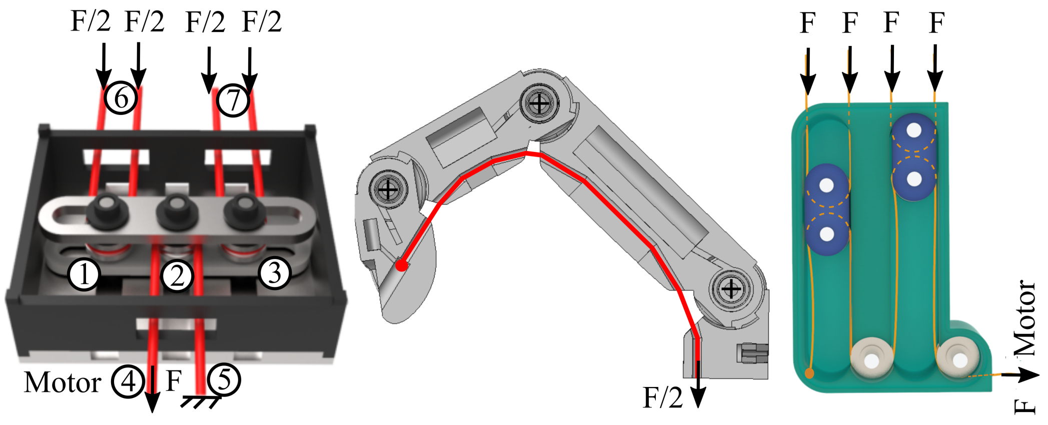


Figure 5.17.: Left: Rocker mechanism of the Prosthetic Hand V1. Adapted from *Source:* (Weiner et al., 2018) © 2018 IEEE.
Middle: Rope routing (red) of a robot hand finger (three joints).
Right: Double pulley mechanism of the Prosthetic Hand V2.

thetic hands was pursued, in which the experience gained from the development of both domains was used for new designs. It should be noted that the designs of the *TUAT/Karlsruhe Humanoid Hand* (Fukaya et al., 2000, 2013) and the *KIT Prosthetic Hand V1* (Weiner et al., 2018) are not contributions of this thesis. However, since all hand designs strongly influenced each other and since the *KIT Prosthetic Hand V1* serves as a supplementary knowledge source for the formalization and the expert system, they are also presented in the following.

The design of all KIT Hands originates from the *TUAT/Karlsruhe Humanoid Hand* developed by Fukaya et al. (2000, 2013) for the first ARMAR (Asfour et al., 2000). Its most distinctive feature is its whippletree couplings mechanism consisting of link-rods, link-plates and joints. This force distribution mechanism allows for an underactuated hand design: A single motor drives all finger and thumb joints. For the design of the KIT Robotic Hands and KIT Prosthetic Hands the original TUAT/Karlsruhe mechanism was modified. Instead of a single motor for all five fingers, they include two motors: one motor actuates the thumb, the other motor actuates the other four fingers using a whippletree couplings mechanism based on the TUAT/Karlsruhe Humanoid Hand. Instead of link-rods, Dyneema ropes are used. Figure 5.17 (left) illustrates this modified mechanism: There are three rope slings (red). Rope sling (6) leads to index and middle finger, rope sling (7) leads to ring and little finger. The lower rope sling is connected to the housing at point (5), while point (4) leads to the finger motor, on whose output shaft a rope pulley is mounted. If the motor rotates, the rope is pulled at point (4). The rocker (gray) moves downwards and is guided linearly over the slide bearings (1), (2) and (3) (black). As a result, loop (6) and (7) also move down and shorten the rope in the fingers, causing the

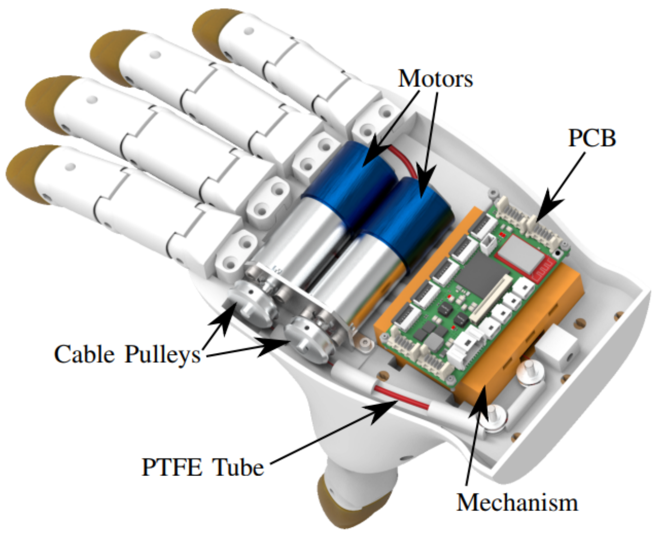


Figure 5.18.: Components of the KIT Prosthetic Hand V1. *Source:* (Weiner et al., 2018) © 2018 IEEE.

fingers to close (Figure 5.17, middle). If fingers block, the same effect as with the original TUAT/Karlsruhe mechanism occurs: The rocker tilts, allowing the other fingers to be closed further. To actuate the thumb, a second motor coils up another rope, which leads to the same effect: the thumb closes. Since ropes can only transmit tensile forces but not pressure, there must be a springback mechanism. For this purpose torsion springs are used in all finger joints.

Besides the modified underactuation mechanism and two encoder-motor-gearbox units with rope pulleys, all KIT Hands also include a PCB that features the motor controller, sensor interfaces and electronics for communication (Figure 5.18). After first functional prototypes, this setup was implemented for the design of the first hands of ARMAR-6: the *KIT Robotic Hand V1*. The hand has 14 joints (two per thumb, three per finger), which are driven by two motors. The fingers as well as the housing are made of ABS and are manufactured using the fused deposition modeling (FDM) 3D printing process. As mechanical and electrical interface to the robot arm, the SCHUNK hand adapter is used (subsection 5.1.3). EtherCAT serves as communication interface.

Based on this development, a male hand prosthesis, the *KIT Prosthetic Hand V1* (Weiner et al., 2018), was developed. Compared to the robotic hand, the number of joints was reduced from 14 to 10 (two per thumb and finger). The prosthesis proportions are based on a 50th percentile male according to DIN 33402 (Jürgens, 2004). The resulting dimensions (LxWxD) are 232x87x35 mm. Mechanical adjustments were mainly used to reduce friction and increase robustness. For example, two small ball bearings were used in all finger joints

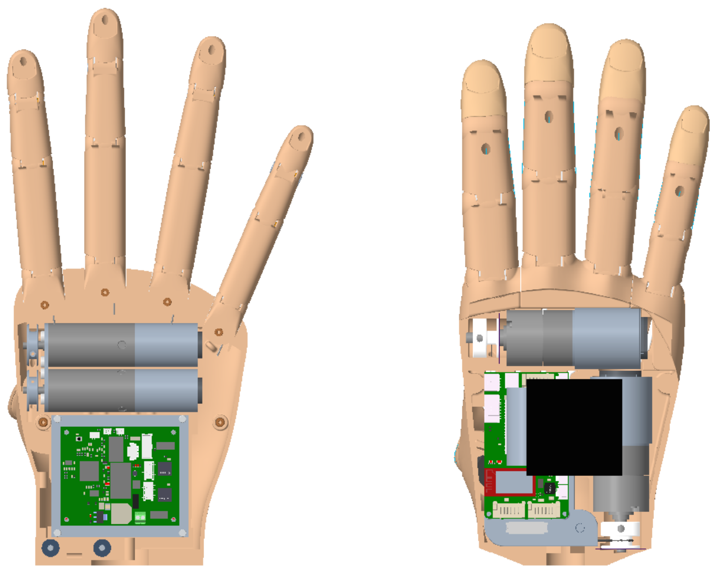


Figure 5.19.: Different motor arrangements of the KIT Hands.
Left: Parallel arrangement (KIT Robotic Hand V2).
Right: Vertical arrangement (KIT Prosthetic Hand V2).

instead of slide bearings. By using laser-sintered plastics (PA 2200), the robustness could be significantly increased. Furthermore, the sensor setup was extended: In addition to the motor encoders, the hand integrates a RGB camera, which is used for object recognition. Its image is shown on a display on the back of the hand.

The second version of the ARMAR-6 hand, the *KIT Robotic Hand V2*, was developed using the knowledge gained from previous developments: Like the KIT Robotic Hand V1, it has 14 joints, a SCHUNK hand adapter and electronics, which enable communication via EtherCAT. The 3D printing process for the manufacturing of the plastic parts (PA 2200) and the design of the fingers were adopted from the KIT Prosthetic Hand V1 to reduce friction and increase robustness. The hand is based on the proportions of a 50th percentile male, scaled up by a factor of 1.3 to fit the ARMAR-6 dual-arm system. This additional space was used to integrate a stronger motor-gearbox combination. Mechanically new on this hand is the use of different torsion springs in the finger joints. So that the distal joints close last, stronger springs are used in these joints than in the others. Furthermore, gloves are used to protect the hand and increase friction. With respect to sensors, a similar setup to the first robotic hand was used, with the addition of an IMU. And finally, the plastic parts of the hand were optimized much more for lightweight construction. For its large dimensions 281x113x37 mm the robotic hand is comparatively light (880 g).

This weight optimization was pursued further with the *KIT Prosthetic Hand V2*. Since it was designed as a prosthesis for a 50th percentile female, it is not only the smallest KIT Hand (194x77x28 mm) but also by far the lightest (388 g). Besides the reduced wall thickness of the plastic parts and the smaller dimensions, a new version of the TUAT/Karlsruhe mechanism also contributes to this weight reduction (Figure 5.17, right): The double pulley increases the transmission ratio by factor 2. Combined with a reduction of the diameter of the finger motor pulley from 16 to 8 mm, which corresponds to a further doubling of the transmission ratio (in total 4), the planetary gear can be reduced by one gear stage. This makes the encoder-motor-gearbox units shorter, lighter and cheaper. Further cost and weight savings result from manufacturing the housing parts from 3D-printed plastic instead of aluminum alloys. The smaller width of the mechanism allows the motors to be arranged vertically instead of parallel to each other (Figure 5.19). To further save space and weight, the knuckles are integrated directly into the palm housing. Compared to the KIT Prosthetic Hand V1, the sensor setup is extended by the IMU from the robotic hands and a distance sensor. In addition, the gripping properties of the prosthesis are improved by finger pads.

Knowledge Sources

The two latest KIT Hands are the main sources for the formalization process: the KIT Robotic Hand V2 and the KIT Prosthetic Hand V2 (Table 5.12). To parameterize the performance properties, the KIT Prosthetic Hand V1 is also analyzed. Since the hand designs are all based on previous developments, knowledge from earlier hand design is also indirectly taken into account.

Table 5.12.: Sources from which knowledge on the design of robotic and prosthetic hands is gained.

Name	**Source Type**	**Knowledge Acquisition**	**Derived Knowledge**
KIT Robotic Hand V2	Main source	Design, system analysis	All catalog components and solution principles
KIT Prosthetic Hand V2	Main source	Design, system analysis	All catalog components and solution principles
KIT Prosthetic Hand V1	Supplementary	System analysis (CAD, tests, literature[1])	Parameterization of spatial dimensions, weight and costs

[1] (Weiner et al., 2018)

5.3.2. Formalization

Based on specific models of the three KIT Hands, which serve as a source of knowledge, a generalized model was created. Thereby 30 solution principles for 13 parameters were identified: 7 functional, 2 structural and 4 kinematic parameters. The functional parameters represent optional catalog components or other optional subcomponents which can be part of a robotic/prosthetic hand, e.g. sensors (Table 5.13).

Table 5.13.: Solution principles for functional parameters of robotic/prosthetic hands.

	Functional						
	Glove	**Finger Pads**	**Hand PCB**	**IMU**	**Distance Sensor**	**Camera**	**Screen**
	(GL)	**(FP)**	**(HB)**	**(IM)**	**(DS)**	**(CA)**	**(SC)**
0	None	None	None	None	None	None	None
1	Yes	Yes	Yes	Yes	Yes	Yes	Yes

The two structural parameters (Table 5.14, right) have a strong influence on the arrangement of the components. The two solution principles for the structural parameter *motor arrangement* express whether the thumb motors are arranged *parallel* or *vertical* to the finger motors (Figure 5.19). *Modular* knuckles (*KIT Robotic Hand V2*) allow a quick exchange of the fingers, but space can be saved by *integrating* the knuckles into the palm housing (*KIT Prosthetic Hand V2*).

Unlike the model for SA units, different solution principles for kinematic parameters could be identified (Table 5.14, left). For example, robotic hands can be realized with fewer than *four* fingers plus thumb and it must be distinguished whether *two* or *three joints per finger* are realized. The number of *finger motors* can also be varied. And finally a *left* or *right* hand can be designed.

Table 5.14.: Solution principles for kinematic (left) and structural (right) parameters of robotic/prosthetic hands.

	Kinematic				**Structural**	
	Body Side (S)	**Finger Number (F)**	**Joints per Finger (J)**	**Finger Motors (FM)**	**Motor Arrangement (A)**	**Finger Knuckles (K)**
1	Right	One		One	Parallel	Modular
2	Left	Two	Two	Two	Vertical	Integrated
3		Three	Three	Three		
4		Four		Four		

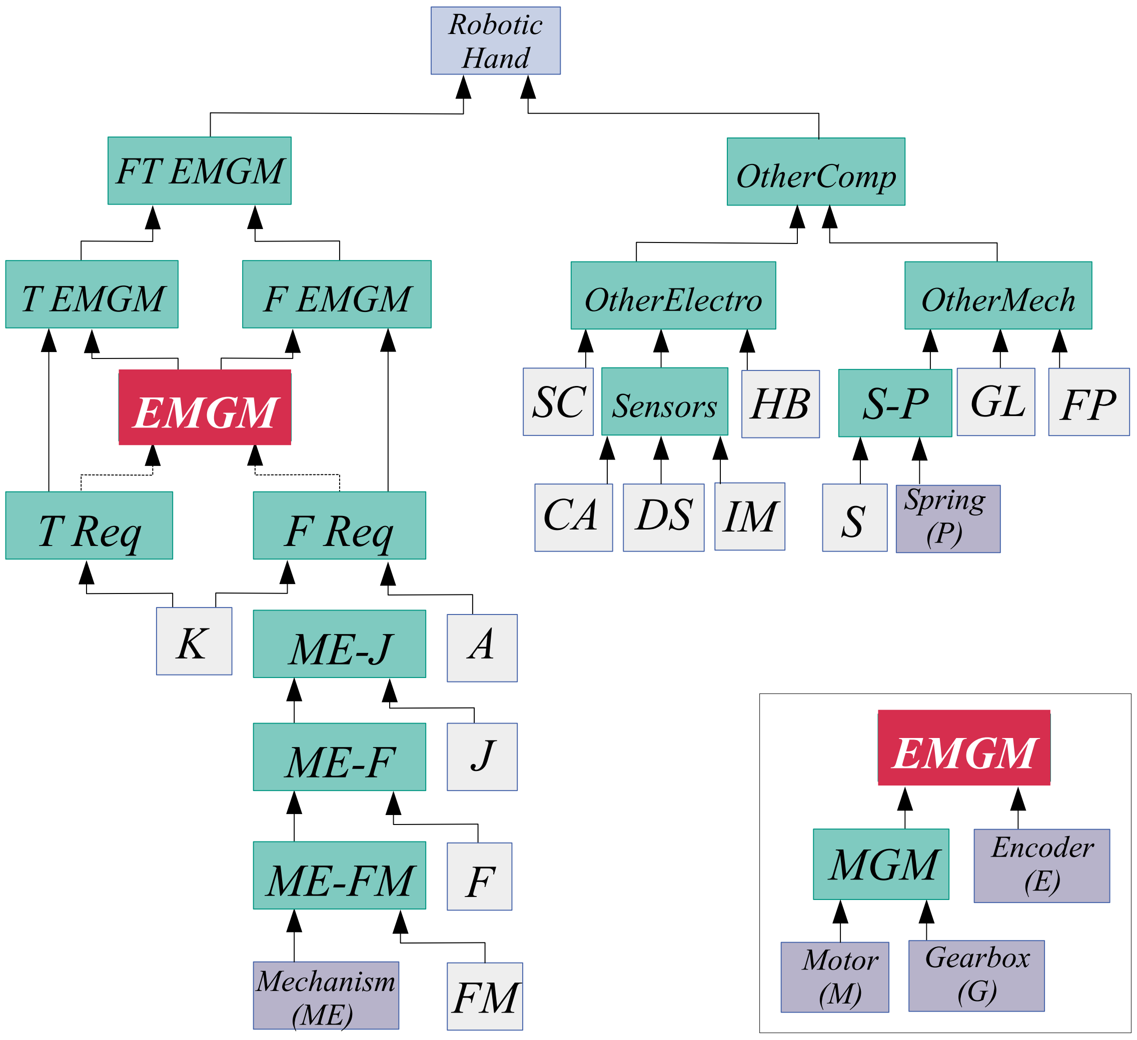


Figure 5.20.: Multi-Stage Design Graph for robotic/prosthetic hands.
Bottom right: Subgraph for Encoder-Motor-Gearbox Match (EMGM).
Adapted from (Förster, 2019).

Figure 5.20 shows the Multi-Stage Design Graph for robotic/prosthetic hands. The parameters for the solution principles are shown in light violet, the catalog component types and the resulting *Robotic Hand* in dark violet. *Mechanisms (ME)* are treated as catalog components for simplification, even though they are compositions of numerous small catalog components such as bearings, screws and pins. Nodes for partial solutions are shown in dark green. The explanations of the abbreviations can be found in Table 5.15. For each node shown in Figure 5.20, there is a satisfied counterpart (not illustrated for simplification).

A special node is the Encoder-Motor-Gearbox Match (*EMGM*), which is colored in dark red. It represents a subgraph for the EMGM (Figure 5.20, bottom right). The use of the subgraph results from the modular relationship between the

Table 5.15.: Nodes of the Multi-Stage Design Graph for robotic/prosthetic hands.

Node	Description	Node	Description
EMGM	Encoder-Motor-Gearbox Match	*MGM*	Motor-Gearbox Match
		OtherComp	Other subcomponents
F EMGM	EMGM for finger actuation	*OtherElectro*	Other electronic components
F Req	Finger EMGM requirements	*OtherMech*	Other mech. compononents
FT EMGM	Finger-Thumb-EMGM Match	*S-P*	Body-Side-Spring Match
ME-F	Finger-Mechanism-Match	*Sensors*	Sensor Setup
ME-FM	Motor-Mechanism Match	*T EMGM*	EMGM for thumb actuation
ME-J	Joints-Mechanism-Match	*T Req*	Thumb EMGM requirements

expert systems for robot/prosthetic hands and EMGM and is explained in the following subsection.

5.3.3. Expert System

Based on the formalization process, an expert system for robotic hands/prosthetic hands was created. This subsection presents its modular structure, the requirements definition, the reasoning and the scope of the system.

Modular Expert Systems

Robot components can be assigned to different hierarchical levels of a humanoid robot. If components of a higher level are built, it is necessary to use robot components of a lower level, which themselves already represent a composition of catalog components. In order to consider this relationship in the development of the expert systems, a modular approach is realized (Figure 5.21): Similar to users, high-level expert systems are able to execute low-level expert systems. For this purpose requirements resulting from the reasoning of the high-level expert system are used to get solutions from the low-level expert system. Based on these solutions, the reasoning of the high-level expert system is then continued. In case of the robotic/prostehtic hand expert system the EMGM system is used as a low-level expert system. If the reasoning reaches a node with a *hasChild* relationship to the node EMGM, the object property *useOntologyFor* in the *Robotic Hand Design Ontology* refers to the *EMGM Design Ontology*. A second working ontology is created for the EMGM, which is used for a separate multi-stage reasoning process. The Java code coordinates the data transfer between the two working ontologies. The EMGM expert system can be executed several times in succession if there are different sets of requirements.

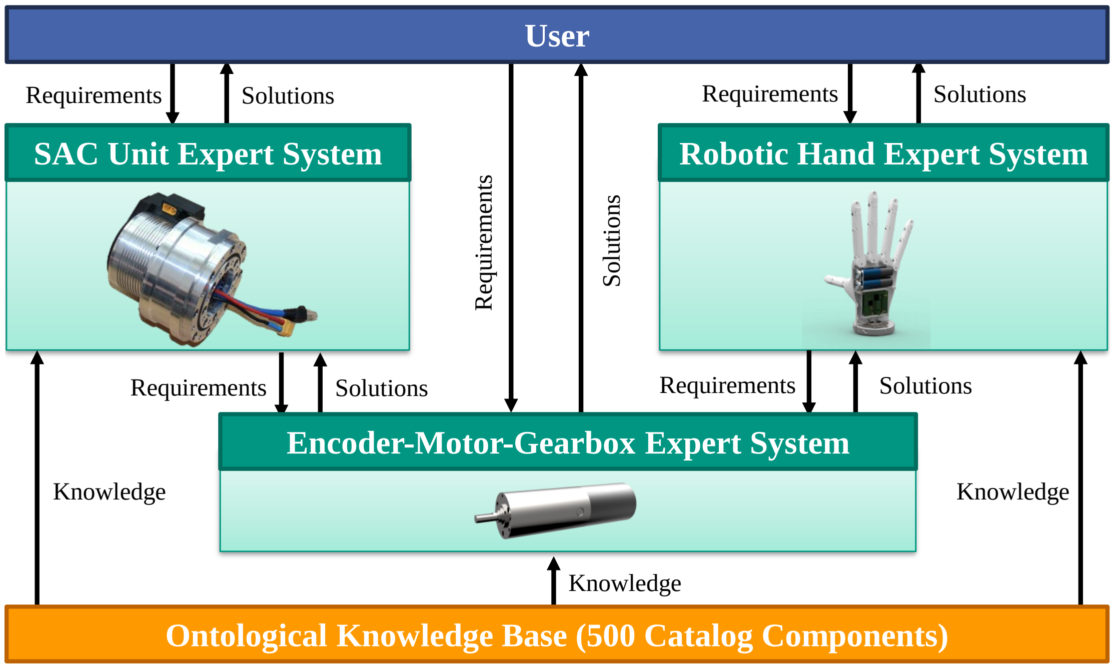


Figure 5.21.: Modularity of the expert systems.

With regard to the ontological knowledge base, the modular architecture presented in subsection 3.4.2 is used. Besides the upper ontologies, the *Abstract Ontology* and the *Component Ontology*, two specific *HRC Design Ontologies* are used: The *Robotic Hand Design Ontology* and the *EMGM Design Ontology*. In order to use data on human proportions, the *Robotic Hand Ontology* imports the *FMA Ontology* and the *Human Body Ontology*.

Requirements Definition

Similar to the expert system for SA/SAC units, requirements and constraints are first defined via the UI before the expert system is started (Table 5.16). A special requirement of the expert system for robotic/prosthetic hands is the requirement *User (Proportions)*. Instead of requiring the user to specify length, width and height of the hand in millimeters, the user specifies whether the hand proportions should correspond to those of a small, medium or large man or woman. Gender and percentiles in steps of 5 are freely selectable. Optionally, the proportions can be changed by a scaling factor if a large robotic hand is to be built, for example. The expert system decomposes these high-level requirements into performance requirements for the length, width and thickness of the hand. To do this, it uses the data from DIN 33402 that is stored in

Table 5.16.: Requirements, constraints and adjustable parameters of the expert system.

Requirements	
Functional	User (proportions)[1], GL, FP, HB, IM, DS, CA, SC
Performance	Force finger tip, force thumb tip, closing time finger, closing time thumb, weight, costs, encoder resolution
Interface	Input voltage, screw flange (number of screws and size)
Constraints	
Structural Parameters	Motor arrangement (A), finger knuckles (K)
Kinematic Parameters	Body side (S), finger number (F), joints per finger (J), finger motors (FM)
Mechanism	Selection of underactuation mechanism (optional)
Motor-Gearbox Match	Use same EMGM for fingers and thumb, exclude gearboxes with a ratio > 100 (adjustable), motor technology
Adjustable Parameters (Expert Mode)	
Dimensions	Additional length
Power transmission	Diameter of motor pulley (thumb and fingers), distance between rope and joint axes
Structure	Wall thickness of housing and fingers
Material	Material of structural parts (housing, fingers)

[1] Gender, percentile, scale

the *Human Body Ontology*. In addition to the outer dimensions of the hand, the data is also used to determine the segment lengths and widths of the fingers.

Mechanism Selection and Arrangement

The Multi-Stage Reasoning is based on the Multi-Stage Design Graph for robotic/ prosthetic hands (Figure 5.20).

First, depending on the selected number of finger motors (FM) and fingers (F), suitable mechanisms are selected to transmit the mechanical power from the finger motors to the fingers. As shown in Table 5.17, there are currently four different mechanisms (*ME0, ME1, ME2, ME3*) of which up to two alternatives can be used depending on the combination of number of fingers and finger motors. It is possible to use a mechanism more than once (number of identical mechanisms q_{Mech}). The length of the mechanisms depends on the number of joints (J), since more joints result in more rope having to be wound up.

After selecting the mechanism, the arrangement of the subcomponents is determined by solution principles for the structural parameters *Finger Knuckles (K)* and *Motor Arrangement (A)*. Since the catalog components of the EMGM,

Table 5.17.: Mechanism selection for finger actuation.

Mechanism for Finger Motors (ME)		**Finger Motors (FM) $q_{FingerMotor}$** 1	2	3	4
Finger Number (F) j_{Finger}	**1**	ME0 ($q_{Mech} = 1$)	-	-	-
	2	ME3 ($q_{Mech} = 1$)	ME0 ($q_{Mech} = 2$)	-	-
	3	-	-	ME0 ($q_{Mech} = 3$)	-
	4	ME1 ($q_{Mech} = 1$)	ME3 ($q_{Mech} = 2$)	-	ME0 ($q_{Mech} = 4$)
		ME2 ($q_{Mech} = 1$)			

ME0: Only Rope, ME1: Double Pulley, ME2: Rocker, ME3: Single Pulley
q_{Mech}: Number of identical mechanisms

i.e. motors, gearboxes and encoders, are still unknown at this point, sets of requirements for the spatial dimensions of the EMGM (maximum length L and diameter D) are calculated depending on the solution principles. For example, the maximum length of the EMGM for the fingers depends on the dimensions of the hand and the solution principles for the structural parameters.

Encoder-Motor-Gearbox-Match (EMGM)

In the Multi-Stage Design Graph for robotic/prosthetic hands two different types of EMGM are foreseen: One type of EMGM actuates the thumb, another type of EMGM the remaining fingers. Since the user does not directly specify the requirements for the EMGM, these requirements must be derived by logical decomposition of the high-level requirements for the hand. The transformation from the required force at the fingertip F_F to a peak torque requirement for an EMGM T_{EMGM} is described by Equation 5.15 to 5.18, which are implemented as SWRL rules.

$$F_{FingerInput} = (F_F \cdot L_{JointTip} + T_{Springs})/L_{RopeJoint} \tag{5.15}$$

$$F_{MechOutput} = F_{FingerInput} \cdot j_{Finger}/q_{Mech} \tag{5.16}$$

$$F_{Pu} = F_{MechOutput}/i_{Mech,x} \tag{5.17}$$

$$T_{EMGM} = F_{Pu} \cdot (D_{Pu} + D_{Rope})/2 \tag{5.18}$$

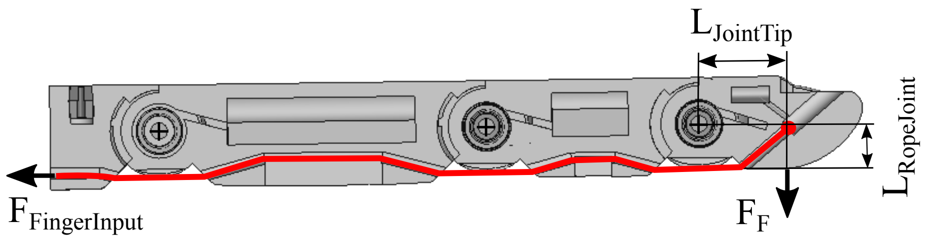


Figure 5.22.: Actuation of a finger with three joints. The rope is colored in red.

First of all, the force ($F_{FingerInput}$) is calculated with which the rope of the finger has to be pulled to achieve the desired force at the fingertip (F_F). The lever arms ($L_{JointTip}$ and $L_{RopeJoint}$) are required for the transmission ratio (Figure 5.22). They are derived from the selected hand portions according to DIN 33402. In addition, the torque due to the springs ($T_{Springs}$) in the joints is taken into account. This torque must also be overcome. Since no springs have been selected yet, the weakest springs are assumed at this point (best case assumption). The required force output of the mechanism ($F_{MechOutput}$) depends on the number of fingers to be actuated (j_{Finger}) and the number of mechanisms (q_{Mech}). The transmission ratio of the mechanism ($i_{Mech,x}$) determines the force of the rope that is wound up by the motor pulley (F_{Pu}). Finally, this rope force (F_{Pu}) is converted into the required torque of the EMGM (T_{EMGM}) by the radius of the rope pulley ($D_{Pu}/2$) plus half the thickness of the rope ($D_{Rope}/2$).

Just as the torque requirements of the EMGM are calculated from the finger force, the required maximum speed of the EMGM v_{EMGM} is derived from the closing time t_F of the finger joints (Equation 5.19 to 5.21). As auxiliary variables, first the speed at the output of the mechanism $v_{MechOutput}$ is calculated and then the speed at the motor rope pulley v_{Pu}.

$$v_{MechOutput} = (2 \cdot L_{RopeJoint} \cdot j_{FingerJoints})/t_F \tag{5.19}$$

$$v_{Pu} = v_{MechOutput} \cdot i_{Mech,x} \tag{5.20}$$

$$v_{EMGM} = (v_{Pu} \cdot 360°)/((D_{Pu} + D_{Rope}) \cdot \Pi) \tag{5.21}$$

Since the closing speed depends on how much rope must be wound up to close all joints, a higher number of finger joints $j_{FingerJoints}$ results in a higher speed requirement v_{EMGM}.

$$m_{Structure} = (V_{Palm} + V_{Fingers}) \cdot \rho_{Structure} \tag{5.22}$$

$$m_{EMGM} = (m - m_{Structure})/(q_{FingerMotor} + 1) \tag{5.23}$$

$$c_{EMGM} = c/(q_{FingerMotor} + 1) \tag{5.24}$$

At this point, the weight m_{EMGM} and cost c_{EMGM} requirements for the EMGM are roughly calculated using best-case assumptions. To determine m_{EMGM}, first the weight of the structural parts $m_{Structure}$ is calculated, whose volume is estimated by geometrical primitives: the palm of the hand is approximated by several cuboids (V_{Palm}), while hollow cylinders are used for the fingers ($V_{Fingers}$). The weight of the structural parts $m_{Structure}$ results from multiplying the resulting total volume by the density $\rho_{Structure}$ of the material. By subtracting the weight $m_{Structure}$ from the weight requirement m, the maximum weight that can be used for the catalog components is obtained. Divided by the number of finger motors $q_{FingerMotor}$ and an additional motor for the thumb (+1), this results in a rough estimation for the maximum weight of each EMGM m_{EMGM}. For a first cost estimation c_{EMGM}, the total costs c are divided by the number of motors for the fingers $q_{FingerMotor}$ and the thumb (+1).

Apart from torques T_{EMGM}, speeds v_{EMGM}, weights m_{EMGM} and costs c_{EMGM}, the maximum spatial dimensions of the EMGM (D_{EMGM},L_{EMGM}) are also calculated. They depend on the maximum spatial dimensions of the hand, the solution principles for the structural parameters and the selected mechanism. This results in a set of requirements for the EMGM of the thumb and the EMGM of the fingers, represented by the nodes *T Req* and *F Req* (Figure 5.20). For each EMGM requirement set the EMGM expert system is executed.

In the *FT EMGM* node the results for thumb and finger EMGM are combined. This step is used for the option *"Use same EMGM for fingers and thumb"* to exclude EMGM that are only usable for the actuation of the fingers or the thumb. Furthermore, it is also checked whether the specific combination of the thumb EMGM and the finger EMGM meets the maximum spatial dimensions of the palm.

Robotic Hand

Now the catalog components and solution principles will be considered, which have less influence on the hand dimensions (*OtherComp*). Through their selection, it is possible to calculate the performance properties (spatial dimensions, finger forces, closing times, costs, weight) of the *Robotic Hand* solutions exactly and exclude unsuitable design solutions. For this purpose, the formulas for determining the EMGM requirements are now applied in reverse: From torques of the EMGM T_{EMGM} the finger forces F_F are calculated, from speeds n_{EMGM} finger closing times t_F, etc... The resulting valid solutions for robotic/prosthetic hands are presented via the UI.

Scope of the System

The ontological knowledge base of the hand expert system comprises 1344 OWL classes, 97 object properties, 331 data properties and 258 SWRL rules. Furthermore, the hand expert system uses the EMGM expert system with its 848 OWL classes, 65 object properties, 145 data properties and 82 SWRL rules.

Both expert systems use the same *Component Ontology* as the expert system for SA units (Appendix A). However, the 288 gearboxes, 68 motors and 27 incremental encoders are not all compatible or suitable for hands: A total of 25,902 valid EMGM remain from 528,768. Since different EMGM can be used for the actuation of the fingers and thumb, there are about 671,000,000 valid Finger-Thumb-EMGM combinations. Considering the different mechanisms and the 30 solution principles, there are approximately 10^{13} valid solutions for robotic/prosthetic hands. In this, the dimensions of the hand are not considered, because there is no discrete number of solutions due to the continuous scalability of the proportions. The detailed calculation is presented in Appendix C.

5.3.4. Reproduction

To evaluate that the hand expert system is able to preserve design knowledge, the KIT Hands are reproduced, which serve as knowledge sources (subsection 5.3.1): the *KIT Prosthetic Hand V1*, the *KIT Prosthetic Hand V2* and the *KIT Robotic Hand V2*. The spatial dimensions (length L_H, width W_H, thickness T_H) and the weight of the hand (m) as well as the finger forces (F) and closing times (t) serve as performance requirements. Spatial dimensions are derived from the user high-level requirement that the hand proportions should correspond to a certain percentile and gender, e.g. the 50th percentile female. For finger forces and closing times a distinction is made between thumb (F_T, t_T) and index finger (F_F, t_F). The values of the index finger are exemplary for the four fingers, which all have the same closing time but slightly different forces due to their different lengths. The calculation of finger forces, as described in Equation 5.15 to 5.18, does not take friction into account, as this is dependent on the individual fabrication of each finger. For the calculation of the closing time, the nominal speed of the motor is used. Thus at low forces the closing time can be even higher than calculated. Nevertheless, measurements on the existing KIT Prosthetic Hand V1 as presented in Weiner et al. (2018) show that the finger force and closing time of the index finger (11.02 N, 1.3 s) correspond

Table 5.18.: Reproduction of the KIT Hands.

		Performance Requirements / Properties							Rating	
KIT Hand		L_H [mm]	W_H [mm]	T_H [mm]	F_F | F_T [N]	t_F | t_T [s]	m [kg]	Costs [€]	NRMSD	PX
Prosthetic	Req	232*	87*	35*	11 | 46	1.4 | 0.7	0.68			
Hand V1[1]	Sol	233	87	35	11 | 46	1.4 | 0.7	0.66	1241	0.06	0.98
Prosthetic	Req	194†	77†	28†	16 | 51	1.4 | 0.5	0.38			
Hand V2	Sol	194	77	28	16 | 51	1.4 | 0.5	0.37	908	0.03	0.99
Robotic	Req	281‡	113‡	37‡	35 | 50	1.5 | 0.5	0.88			
Hand V2[2]	Sol.	281	113	39	35 | 50	1.5 | 0.5	0.88	1268	0.01	1.00

	Parameters for Solution Principles			
	Kinematic	**Structure**	**Functional**	**Mechanism**
Req	S1 F4 J2 FM1	A1 K1	GL0 FP0 HB1 IM0 DS0 CA1 SC1	Rocker
Sol	S1 F4 J2 FM1	A1 K1	GL0 FP0 HB1 IM0 DS0 CA1 SC1	Rocker
Req	S1 F4 J2 FM1	A2 K2	GL0 FP1 HB1 IM1 DS1 CA1 SC1	Double Pulley
Sol	S1 F4 J2 FM1	A2 K2	GL0 FP1 HB1 IM1 DS1 CA1 SC1	Double Pulley
Req	S1 F4 J3 FM1	A1 K1	GL1 FP0 HB1 IM1 DS0 CA0 SC0	Rocker
Sol	S1 F4 J3 FM1	A1 K1	GL1 FP0 HB1 IM1 DS0 CA0 SC0	Rocker

	Adjustable Parameters			Interface		Catalog Components (EMGM)		
	L_{add}	D_{Pu} (F|T)	t_w (F|P)	U	Flange	Motor	Gearbox	Encoder
Req	24 mm	16|16 mm	8|2 mm	12 V	2xM4			
Sol	24 mm	16|16 mm	8|2 mm	12 V	2xM4	2224SR	201R86:1	IEH2-512
Req	6 mm	8|5 mm	2|2 mm	12 V	4xM4			
Sol	6 mm	8|5 mm	2|2 mm	12 V	4xM4	2224SR	201R23:1	IEH2-512
Req	10 mm	14|14 mm	4|2 mm	24 V	4xM6			
Sol	10 mm	14|14 mm	4|2 mm	24 V	4xM6	2232SR	22F71:1	IEH2-512

Abbreviations: Req. = Requirements, Sol. = Solution, F = Finger, T = Thumb ,
P = Palm, L_{add} = Additional Length, D_{Pu} = Pulley diameter, t_w = Wall thickness
Rating ($NRMSD$, PX): $L_H, W_H, T_H, F_F, F_T, t_F, t_T, m$ and € are weighted equally
* 50th percentile male † 50th percentile female ‡ 50th percentile male x 1.3
[1](Weiner et al., 2018) [2](Asfour et al., 2019b)

approximately to the calculation used by the expert system (10.96 N, 1.35 s). Besides performance requirements, functional requirements (e.g. integration of sensors), interface requirements (power supply, screw flange) and constraints (e.g. chosen solution principles such as 2 joints per finger) serve as input.

Table 5.18 shows that forces and closing times of the design solutions generated by the expert system (*Sol*) correspond exactly to the requirements (*Req*). This results from the fact that the design solutions use the same EMGM (Encoder-Motor-Gearbox match) and mechanisms as the existing KIT Hands. The spatial dimensions and weight of the design solutions, which were determined depending on the requirements (percentile, gender), catalog components and solution principles, are also close to the values of the existing KIT Hands. This results in a low *NRMSD*< 0.1 and a performance index of about *PX*≈ 1, demonstrating that the expert system can preserve the knowledge of the KIT Hands that serve as knowledge sources.

5.3.5. Optimization

In order to demonstrate that the expert system can not only reproduce knowledge but also use it for new developments, the KIT Hands are first optimized by different performance requirements. As performance requirements, the performance properties of the existing KIT Hands are used. Finger forces, closing times, costs and weight may deviate negatively by up to 20%. With regard to closing forces, the design is optimized by the closing time of the fingers (t_F), as the usually much shorter closing time of the thumb (t_T) does not bring a big advantage. Since the spatial dimensions of the hands are derived from human proportions, they must not fall below them, which would correspond to a positive deviation in the case of SAC units, for example. Negative deviations, i.e. exceeding the spatial dimensions, are allowed up to 5% by default. Since the existing KIT Prosthetic Hands have a negative deviation in thickness of +5 mm (V1) or +2 mm (V2), the corresponding deviations are also allowed during optimization. In contrast to the reproduction, the expert system can use both solution principles for the optimization with regard to the structural parameter *Motor Arrangement (A)*. Additionally, all known mechanisms, motor technologies (brushed/brushless) and two different EMGMs for the actuation of the thumb and the fingers may be used. With regard to wall thickness ($t_w = 2$) and pulley diameters ($D_{Pu,Finger} = 8\,\mathrm{mm}, D_{Pu,Thumb} = 5\,\mathrm{mm}$), the standard settings of the expert system are used, which have proven to be successful with the KIT Prosthetic Hand V2. The first criterion for the optimization is always given in the *Optimization* column of the tables. If no second criterion is specified (separated by a comma after the first criterion), PX is used. Improvements in performance properties are expressed as a percentage compared to the results of the reproduction in order to make smaller deviations more clearly visible.

Optimization of the KIT Prosthetic Hand V1

Table 5.19.: Optimization of the KIT Prosthetic Hand V1.

Prosthetic Hand V1	**Performance Requirements / Properties**							**Rating**	
Optimization	L_H **[mm]**	W_H **[mm]**	T_H **[mm]**	F_F | F_T **[N]**	t_F | t_T **[s]**	m **[kg]**	**Costs [€]**	**NRMSD**	**PX**
Reproduced	233	87	35	11 | 46	1.4 | 0.7	0.66	1241	0.06	
Finger force F_F	233	87	30	**36** | 45	1.5 | 0.2	0.48	941	0.04	2.03
Thumb force F_T	233	87	30	9 | **79**	0.3 | 0.4	0.44	1008	0.05	1.73
Closing time t_F, t_T	233	87	30	9 | 45	**0.3** | **0.2**	0.44	1005	0.05	2.13
Weight m	233	87	30	11 | 38	1.2 | 1.3	**0.38**	876	0.06	1.14
Costs	233	87	30	11 | 52	1.2 | 0.7	0.39	**857**	0.00	1.28
NRMSD (e)	233	87	30	34 | 52	1.1 | 0.7	0.42	881	**0.00**	1.51
PX, NRMSD (e)	233	87	30	9 | 45	0.3 | 0.2	0.44	1005	0.05	**2.13**

	Parameter	**Finger EMGM**		**Thumb EMGM**		
Opt.	**Structure**	**Motor**	**Gearbox**	**Motor**	**Gearbox**	**Mechanism**
Rep.	A1 K1	2224SR	201R 86:1	2224SR	201R 86:1	Rocker
F_F	A2 K2	2232BX4	22F 25:1	2232SR	201R 9.7:1	Double Pulley
F_T	A2 K2	2036B	201R 14:1	2232SR	201R 23:1	Double Pulley
t_F, t_T	A2 K2	2036B	201R 14:1	2232SR	201R 9.7:1	Double Pulley
m	A1 K2	2224SR	22EKV19:1	1717SR	17/1 81:1	Double Pulley
€	A1 K2	2224SR	22EKV19:1	2224SR	22EKV28:1	Double Pulley
e	A2 K2	2232SR	201R 23:1	2224SR	22EKV28:1	Double Pulley
PX, e	A2 K2	2036B	201R 14:1	2232SR	201R 9.7:1	Double Pulley

Kinematic/Functional parameters and interface correspond to reproduced hand.
Rating ($NRMSD$, PX): $L_H, W_H, T_H, F_F, F_T, t_F, t_T, m$ and € are weighted equally.

The optimizations of the Prosthetic Hand V1 as listed in Table 5.19 result from the following changes compared to the reproduced solution:

- **Finger force (+227%):** The significant increase in force results from the use of the *Double Pulley* mechanism ($i_{Mech} = 4$ instead of $i_{Mech} = 2$), a smaller pulley diameter (8 instead of 16 mm) and a more powerful brushless DC motor (2232BX4). A smaller gear ratio (i=25) is used to meet the closing time requirements. The necessary space for the bigger motor is saved by using a different motor arrangement (A2) and integrated knuckles (K2).

- **Thumb force (+72%):** This is realized by a bigger thumb motor (2232SR).
- **Closing time fingers (-76%), thumb (-74%):** To reduce the closing time of the fingers, the brushless DC motor 2036B with significantly higher nominal speed (11,430 instead of 4,300 rpm) is used and combined with a gearbox with a low gear ratio (i=14). The thumb uses a larger motor (2232SR) with lower gear ratio (i=9.7).
- **Weight (-42%):** The weight of the plastic parts is reduced by using less finger wall thickness and by integrating the knuckles into the palm of the hand (K2). Further weight is saved by a smaller thumb motor (1717SR) and the use of the lighter *Double Pulley* mechanism, which allows a smaller gearbox of the finger motor (i=19) due to its higher gear ratio.
- **Costs (-31%):** This reduction results from the cheaper *Double Pulley* mechanism and a different, cheaper gearbox series (22EKV instead of 201R).
- **NRMSD:** By reducing the hand thickness to 30 mm, there is no longer a negative deviation ($NRMSD = 0$). This is achieved by the thinner *Double Pulley* mechanism.
- **PX:** The result for overall optimization (PX) corresponds to optimization by closing time t_F. This indicates that the closing time has the greatest optimization potential.

Optimization of the KIT Prosthetic Hand V2

With regard to the Prosthetic Hand V2 the following optimizations result (Table 5.20):

- **Finger force (+143%):** This is only achieved by a stronger motor (2232SR).
- **Thumb force (+16%):** By using a slightly higher gear ratio of the thumb gearbox (i=28 instead of i=23) the thumb force can be increased at the expense of thumb closing speed (0.7 s instead of 0.5 s).
- **Closing time fingers (-66%), thumb (±0%):** A stronger motor (2232SR) with a lower gear ratio (i=9.7 instead of i=23) leads to a reduction of the finger closing time.
- **Weight (-8%):** A slight reduction can be achieved by weaker motors (1628B and 1717SR), but this reduces forces and increases closing times.

Table 5.20.: Optimization of the KIT Prosthetic Hand V2.

Prosthetic Hand V2	**Performance Requirements / Properties**							**Rating**	
Optimization	L_H **[mm]**	W_H **[mm]**	T_H **[mm]**	F_F | F_T **[N]**	t_F | t_T **[s]**	m **[kg]**	**Costs [€]**	**NRMSD**	**PX**
Reproduced	194	77	28	16 | 51	1.4 | 0.5	0.37	908	0.03	
Finger force F_F	199	77	28	**38** | 51	1.1 | 0.5	0.39	922	0.03	1.36
Thumb force F_T	199	79	28	16 | **59**	0.5 | 0.7	0.38	899	0.03	1.36
Closing time t_F, t_T	199	77	28	16 | 51	**0.5** | **0.5**	0.39	919	0.03	1.39
Weight m	203	77	26	15 | 43	1.7 | 1.3	**0.34**	1017	0.09	0.97
Costs	194	79	28	16 | 59	1.4 | 0.7	0.36	**888**	0.03	1.15
NRMSD (e)	194	77	28	16 | 51	1.4 | 0.5	0.37	908	**0.03**	1.18
PX, NRMSD (e)	199	77	28	16 | 51	0.5 | 0.5	0.39	919	0.03	**1.39**

	Parameter	**Finger EMGM**		**Thumb EMGM**		
Opt.	**Structure**	**Motor**	**Gearbox**	**Motor**	**Gearbox**	**Mechanism**
Rep.	A2 K2	2224SR	201R 23:1	2224SR	201R 23:1	Double Pulley
F_F	A2 K2	2232SR	201R 23:1	2224SR	201R 23:1	Double Pulley
F_T	A2 K2	2232SR	201R9.7:1	2224SR	22EKV28:1	Double Pulley
t_F, t_T	A2 K2	2232SR	201R9.7:1	2224SR	201R 23:1	Double Pulley
m	A2 K2	1628B	17/1 50:1	1717SR	17/1 81:1	Double Pulley
€	A2 K2	2224SR	201R 23:1	2224SR	22EKV28:1	Double Pulley
e	A2 K2	2224SR	201R 23:1	2224SR	201R 23:1	Double Pulley
PX, e	A2 K2	2232SR	201R9.7:1	2224SR	201R 23:1	Double Pulley

Kinematic/Functional parameters and interface correspond to reproduced hand.
Rating (*NRMSD*, *PX*): $L_H, W_H, T_H, F_F, F_T, t_F, t_T, m$ and € are weighted equally.

- **Costs (-2%):** This small reduction is achieved by using the cheaper gearbox series (22EKV) for the thumb.

- **NRMSD:** The result corresponds to the reproduced, existing Prosthetic Hand V2. Hence, no improvement can be achieved without worsening another performance property. The result is Pareto optimal.

- **PX:** The result corresponds to the optimization by closing time, which has the highest optimization potential. However, this makes the hand 5 mm longer.

Table 5.21.: Optimization of the KIT Robotic Hand V2.

Robotic Hand V2	**Performance Requirements / Properties**							**Rating**	
Optimization	L_H **[mm]**	W_H **[mm]**	T_H **[mm]**	F_F | F_T **[N]**	t_F | t_T **[s]**	m **[kg]**	**Costs [€]**	**NRMSD**	**PX**
Reproduced	281	113	39	35 | 50	1.5 | 0.5	0.88	1268	0.01	
Finger force F_F	281	113	39	**85** | 50	1.8 | 0.3	0.59	955	0.05	1.77
Thumb force F_T	281	113	39	84 | **97**	1.7 | 1.1	0.61	971	0.03	1.37
Closing time t_F, t_T	281	113	39	32 | 50	**0.7** | **0.3**	0.57	949	0.03	1.75
Weight m	281	113	39	56 | 53	1.3 | 1.6	**0.52**	937	0.03	1.20
Costs	281	113	39	56 | 40	1.3 | 1.6	0.53	**916**	0.07	1.18
NRMSD, PX	281	113	39	37 | 50	0.8 | 0.3	0.57	1058	**0.00**	1.74
PX, NRMSD	281	113	39	84 | 50	1.7 | 0.3	0.57	952	0.03	**1.78**

	Parameter	**Finger EMGM**		**Thumb EMGM**		
Opt.	**Structure**	**Motor**	**Gearbox**	**Motor**	**Gearbox**	**Mechanism**
Rep.	A1 K1	2232SR	22F 71:1	2232SR	22F 71:1	Rocker
F_F	A$*$ K2	2232SR	22F 25:1	2232SR	201R 14:1	Double Pulley
F_T	A$*$ K2	2232SR	201R 23:1	2232SR	22/7 43:1	Double Pulley
t_F, t_T	A$*$ K2	2232SR	201R9.7:1	2232SR	201R 14:1	Double Pulley
m	A$*$ K2	2232SR	26AK16:1	1724SR	17/1 68:1	Double Pulley
€	A$*$ K2	2232SR	26AK16:1	2224SR	22EKV69:1	Double Pulley
e, PX	A$*$ K2	2036B	201R 23:1	2232SR	201R 14:1	Double Pulley
PX, e	A$*$ K2	2232SR	201R 23:1	2232SR	201R 14:1	Double Pulley

Kinematic/Functional parameters and interface correspond to reproduced hand.
Rating (*NRMSD*, *PX*): $L_H, W_H, T_H, F_F, F_T, t_F, t_T, m$ and € are weighted equally.
"$*$": Both solution principles result in the same performance properties.

Optimization of the KIT Robotic Hand V2

The optimizations for the Robotic Hand V2 (Table 5.21) result from the following changes:

- **Finger force (+141%):** Since the motor is already the largest in the series, it cannot be replaced to increase the performance. Instead, the increase is achieved by a different effect: By using the *Double Pulley* mechanism with a transmission ratio of 4 instead of 2, the maximum radial bending force that may be applied to the gearbox shaft is increased by a factor of 2. This allows the mechanical power of the motor to be better exploited.

- **Thumb force (+95%):** By using a smaller pulley diameter and a different gearbox (22/7 43:1) the thumb force is increased while the closing time is longer.
- **Closing time fingers (-53%), thumb (-49%):** Lower gear ratios (finger: i=9.7, thumb: i=14) reduce the closing times. The maximum forces are barely reduced, since the radial bending forces are the decisive limitation of the gearbox.
- **Weight (-41%):** The strong weight reduction is achieved by lighter plastic parts (integrated knuckles K2, reduced finger wall thickness), the lighter *Double Pulley* mechanism and a weaker EMGM for the thumb (1724SR motor with 17/1 68:1 gearbox).
- **Costs (-28%):** In addition to the cheaper *Double Pulley* mechanism, costs are saved on the EMGM: The thumb motor is reduced (2224SR) and cheaper gearboxes are used (26AK 16:1 and 22EKV 69:1).
- **NRMSD:** The solution shows that closing times, weight and costs can be reduced without negative deviations. This is achieved by a different finger EMGM (2036B with 20/1R 23:1), a lower thumb gearbox ratio (i=14), the *Double Pulley* mechanism and smaller pulley diameters.
- **PX:** The result for the overall optimization (PX) is similar to that for the NRMSD optimization. Only the finger motor is replaced (2232SR instead of 2036B), resulting in lower costs and higher finger forces at the expense of the finger closing time.

A noteworthy aspect of the Robotic Hand optimization is that half of the design solutions use the parallel (A1) and the other half the vertical motor arrangement (A2). This can be explained by the comparatively large construction space, whereby EMGMs are not excluded due to their length or diameter, regardless of their arrangement.

Optimization Summary

Even though there are many similarities regarding the optimization of individual performance properties, e.g. increasing the finger force by using a stronger motor and a different gear ratio, the optimization potential of the KIT Hands differs significantly, which is reflected by the number of valid solutions: 1045 (Prosthetic Hand V1), 23 (Prosthetic Hand V2), 3120 (Robotic Hand V2). The Robotic Hand V2 and the Prosthetic Hand V1 both benefit from the use of the

Double Pulley mechanism, the smaller pulley diameter, the integration of the knuckles and the reduced finger wall thickness as used in the Prosthetic Hand V2. Based on the current expert knowledge, the performance properties of the Prosthetic Hand V2 can only be improved if others are worsened. The ratio between closing time and finger force can be adjusted very precisely, which can be explained by the large number of gearboxes in the Component Ontology.

5.3.6. Scaling of a Design

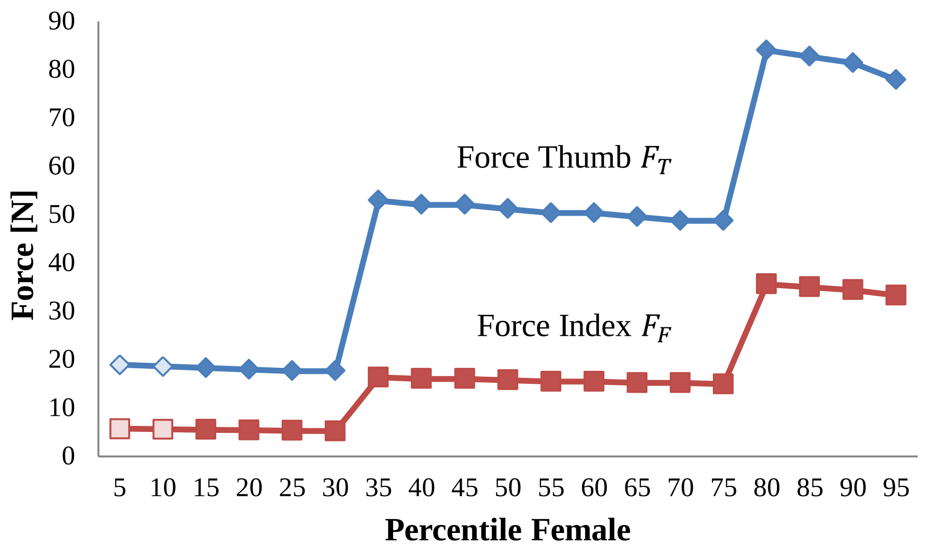
Many design tasks consist of creating variant designs for an existing technical system (subsection 2.1.1). This includes realizing systems in other sizes. For this reason, the ability of the expert system to scale existing solutions is shown in the following. This is made possible by the parameterized formulas and other design rules as well as the large scope of the Component Ontology.

The KIT Prosthetic Hand V2 serves as the hand to be scaled. For different human proportions of male and female hands, the design solution is sought that achieves maximum thumb and finger force. Thumb and finger forces are not specified as requirements, nor are costs and weight. For the closing times, the finger closing time of the Prosthetic Hand V2 (1.4 s) is the maximum. Functional/interface requirements, solution principles and adjustable parameters correspond to the reproduction settings for the Prosthetic Hand V2. Thus, the same EMGM is used for thumb and finger actuation. The only exception is the wall thickness of the palm, which is reduced to 1 mm in order to consider results that are as tight as possible. Proportions and closing times may deviate negatively by a maximum of 5%. The thumb force is the first optimization criterion. If the thumb forces are identical, the finger force is used as the second criterion, the NRMSD as the third criterion and the weight as the fourth criterion.

Table 5.22.: Motors and gearboxes for the scaled KIT Prosthetic Hand V2 (Figure 5.23) depending on gender and percentile.

Percentile Female	**5-30**	**35-75**	**75-95**
Motor (Faulhaber)	1724SR	2224SR	2232SR
Gearbox (Faulhaber)	15/10 15:1	20/1R 23:1	20/1R 23:1

Percentile Male	**5**	**10-95**
Motor (Faulhaber)	1724SR	2232SR
Gearbox (Fauhaber)	15/10 15:1	20/1R 23:1



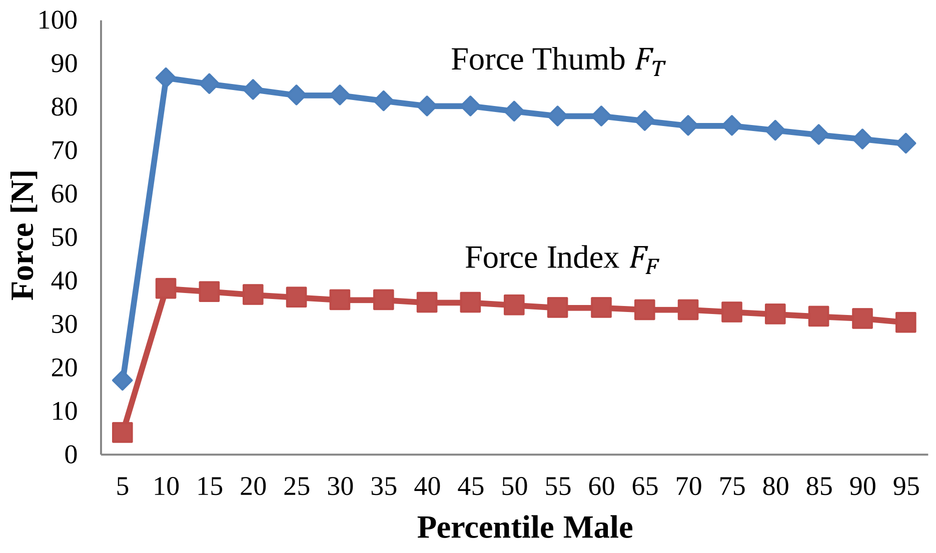


Figure 5.23.: Scalability of the KIT Prosthetic Hand V2. The figures illustrate the maximum forces of the index finger F_F and the thumb F_T with respect to spatial dimensions based on human proportions (percentiles and gender).

Figure 5.23 shows the maximum finger and thumb forces for male and female hand proportions in 5-percentile increments. Table 5.22 lists the motors and gearboxes used by the expert system. For the 5th and 10th percentile female there are only solutions if the hand thickness is exceeded by 10% (10th percentile), respectively 20% (5th percentile). The graph for the female percentiles shows that the force progression can be described approximately as a step function. The jumps result when changing to a more powerful motor, which can only be fitted if the hand is big enough. For example, the finger force is about 5-6 N at hand proportions up to the 30th percentile female and then jumps to about 16 N at the 35th percentile by changing from motor 1724SR to 2224SR. Another motor change occurs between the 75th to the 80th percentile. The graph for male proportions has only a single jump between the 5th and 10th percentile. After that jump the most powerful motor of the series (2232SR) is used. Besides the jumps, another effect can also be observed , which becomes particularly clear when examining the graph of the male proportions: Between the jumps, the force values for fingers and thumb drop slightly. This can be explained by the fact that with the same EMGM, the proportions and thus the length of the finger segments nevertheless increase. Since the finger segments act as a lever arm for the torque providing the finger force, the finger force decreases with increasing proportions if no stronger EMGM is used.

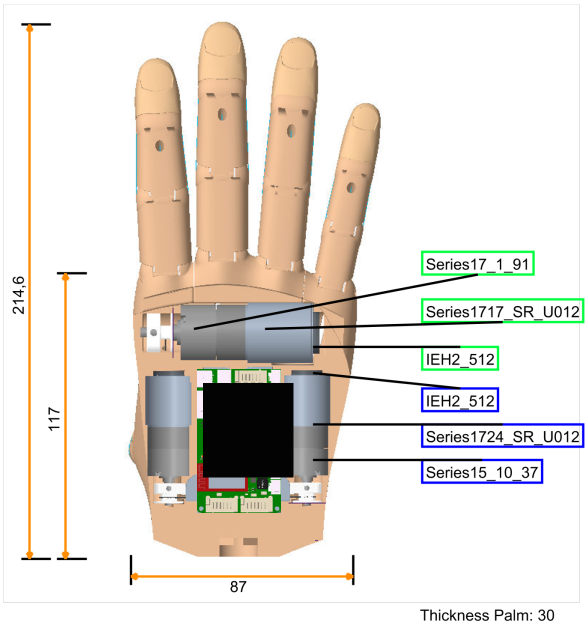
5.3.7. Novel Design

By optimizing and scaling the KIT Hands, it was shown that the expert system is suitable to systematically support future developments of adaptive and variant designs. The third and rarer type of design task are original designs, i.e. novel designs (subsection 2.1.1). While limited by the existing knowledge base, novel designs are still possible to a certain extent through new combinations of solution principles and catalog components.

The goal of the novel design presented in the following is a right (S1) prosthetic hand for an average-sized male user (50th percentile male) with 10 degrees of freedom (J2). The four fingers (F4) shall be actuated by two motors (FM2), the thumb by a third. Different EMGMs can be used for finger and thumb actuation. The minimum finger force is 10 N, the thumb should have about four times this force (40 N). It should be possible to close the hand within 1.4 s. The weight of the hand should not exceed 400 g. Low costs are not a requirement at this stage. As equipment an IMU (IM1), a distance sensor (DS1), a camera (CA1), a screen (SC1) and fingerpads (FP1) are desired. For safety reasons, the power supply must not exceed 12 V. The electronics should be integrated in the hand if possible (HB1) and the motors must be brushed. Apart from that, the default settings are used, which correspond to the settings of the KIT Prosthetic Hand V2. Negative deviations of 5% are allowed, whereby the design solution with the least negative deviation is searched for ($NRMSD \rightarrow 0$). The second criterion is the total optimum of all properties (PX).

The expert system generates 8 solutions for these requirements, of which the best is presented in Figure 5.24. The mechanism used is the *Single Pulley Mechanism (ME3)*, which was derived from the *Double Pulley Mechanism (ME1)* when

RoboticHand



Subcomponents

Motor	Series1724_SR_U012
Gearbox	Series15_10_37
Encoder	IEH2_512
MotorThumb	Series1717_SR_U012
GearboxThumb	Series17_1_91
EncoderThumb	IEH2_512
MechanismMotor	Mech3SinglePulley
Camera	OV2640
DistanceSensor	VL53L1X
IMU	BNO055
PCB	HandPCB
Screen	DD-128128FC-6A

Properties

NRMSD	0,0105
Performance Index (PX)	1,0072
Force finger tip F (index finger)	10,4284 N
Closing time t (finger)	1,4374 s
Number of finger motors (excl. thumb)	2
Force finger tip F (thumb)	42,3923 N
Closing time t (thumb)	1,4172 s
Number of thumb motors	1
Length Hand	214,6 mm
Length Palm	117 mm
Width Palm	87 mm
Thickness Palm	30 mm
Scale factor	1
Side of the body	Right
Number of joints	2
Number of fingers without thumb	4
Weight m	0,4032 kg
Motor arrangement	Vertical
Costs c	1140,7815 €
VAT (Mehrwertsteuer)	Exclude

Figure 5.24.: Novel design for a prosthetic hand (50th percentile male) with three motors (PDF export of the expert system).

creating the knowledge base. Since it drives two fingers each instead of four, it is used twice. Due to the additional mechanism and finger motor, the expert system must use motors with a smaller diameter (1717SR and 1724SR) compared to the reproduced KIT Hands to meet the required hand width. However, since a lesser degree of underactuation also means that less rope has to be wound up per motor, the closing time is reduced at constant motor speed. By selecting a gearbox with a ratio of i=37, the expert system is able to achieve a finger force of more than 10 N, which is only slightly less than the finger forces of the Prosthetic Hand V1 and V2. The closing times and the weight are slightly above the requirement (1.4 s, resp. 400 g), but within the allowed negative deviation of 5%. Overall this results in a very small negative deviation ($NRMSD \approx 0.01$) from the requirements. The novel design matches the proportions of a 50th percentile male hand better than the existing Prosthetic Hand V1, although it integrates an additional motor.

5.4. Summary and Review

This chapter described the evaluation of the approach presented in this thesis using the example of the humanoid robot ARMAR-6.

First the development of the mechatronic components of ARMAR-6 was presented, most notably highly integrated sensor-actuator-controller (SAC) units for robot joints. Furthermore, this section also served to motivate why the expert knowledge gained during the successful development of this complex, high-performance humanoid robot is worth preserving.

Thereafter, two case studies were conducted on the two most highly integrated components of ARMAR-6: SAC units for robot joints and robotic hands. It was shown how knowledge gained through the design of these robot components and analysis of related work can be formalized and used as the knowledge base of an expert system. The resulting expert systems use an ontological knowledge base that comprises approximately 500 catalog components. Furthermore, they take different solution principles for conceptual design parameters into account that describe functional, structural and kinematic options. The expert system for SA units considers 29 solution principles for 11 parameters, while the hand expert system includes 30 solution principles for 13 parameters. This results in more than 10^8 valid SA/SAC units and 10^{13} valid robotic/prosthetic hands. To demonstrate the expert systems' capabilities, existing designs were reproduced, optimized and scaled. Furthermore, novel designs were generated.

In addition to their high scope and level of detail, the presented expert systems differ from most related works by their application. Instead of supporting the design of comparatively small, low-cost robots or other mechatronic systems for educational use, they preserve design knowledge on components of a high-performance humanoid robot.

6. Conclusion

This thesis aimed to find an answer to the following research question:

> *How can expert knowledge on the design of humanoid robot components be preserved in order to support future developments?*

For this purpose, a systematic approach was presented which can be divided into three successive steps. First, design knowledge on humanoid robot components is gained through robot design and analysis of related work. Second, this knowledge is formalized resulting in an ontological knowledge base that allows for automated reasoning. Third, this ontological knowledge base is used by an expert system to generate design solutions based on user requirements.

The presented formalization process makes it possible to *preserve expert knowledge on the design of humanoid robot components*. By using this knowledge as part of an expert system, it can be further used to *support future developments*. This was evaluated through case studies on components of the humanoid robot ARMAR-6: sensor-actuator-controller (SAC) units and robotic hands.

6.1. Scientific Contributions of the Thesis

The presented approach and its evaluation result in the three main scientific contributions of this thesis, which will be summarized in the following.

Formalization of Expert Knowledge on Humanoid Robot Design

The formalization process is the first contribution of this work. But to carry it out, design knowledge must first be acquired. As described in chapter 3, this knowledge is gained through own designs and related work. It was explained why it is important to use own designs as the main source of knowledge for complex, highly integrated systems, while knowledge from related work may be used as a supplement. In addition, it was described which knowledge has to be gained by system analysis to perform the formalization process.

The formalization process was then explained step by step. Based on the system analysis a specific model is created for each specific robot component serving as knowledge source. Therefore, SysML Block Definition Diagrams (BDD) are used - a modeling tool from systems engineering, which allows to represent relationships between subcomponents of a technical system. This is combined with rules and formulas that describe relationships between the subcomponents and the properties of the overall system.

Based on the specific models, a generalized model for the robot component type is induced. The generalized model is also modeled using a SysML BDD. In contrast to the specific models, however, the generalized model is represented by a hierarchical structure tree. This means that the robot component is decomposed hierarchically into subsystems up to catalog component types. Furthermore, formulas and rules are parameterized. In order to consider not only similarities but also differences in the design of a robot component type, different solution principles for decisive conceptual design parameters are identified similar to morphological boxes. Examples for such solution principles are different catalog component arrangements and possibilities to realize functionalities. They are represented in the BDD by different paths or leaf nodes and differ from each other by different formulas and rules. To ensure that the generalized model not only considers the structure of a robot component but also the requirements, the BDD is first transformed into a directed, acyclic graph (DAG). Blocks representing catalog components and partial solutions are transformed into nodes, the rules and formulas describing the transition between the blocks are represented by edges. The DAG is structured in a way that it describes the bottom-up design of the robot component type, starting with the source nodes: catalog components and solution principles. To model when requirements are met, each node of the DAG is followed by a counterpart node expressing that the requirements are satisfied. Adding requirement checks after each node has the advantage that unsuitable catalog components, partial solutions and solution principles can be excluded as early as possible. The resulting DAG, the Multi-Stage Design Graph, therefore contains the necessary knowledge to describe the requirements-oriented design of a humanoid robot component type.

Finally, it was explained how the Multi-Stage Design Graph and knowledge on catalog components of manufacturers is transformed into an ontological knowledge base. The modular ontology architecture allows to use some ontologies for all expert systems (e.g. the Component Ontology), while very specific ontologies that describe the Multi-Stage Design Graph to a robot component type are only used if needed.

The described formalization process ultimately serves to make design knowledge usable for automated reasoning. To achieve this, methods of product development (morphological boxes) and systems engineering (SysML) are combined with knowledge representation (graphs, ontologies, rules). The resulting combination is suitable for preserving both procedural and domain-specific knowledge at a high level of detail, as required for the design of highly integrated robot components.

Expert System to Support the Design of Humanoid Robot Components

The second contribution of this thesis is an expert system that supports the design of humanoid robot components. It is based on the ontological knowledge base resulting from the formalization process. In chapter 4 the other two main components of the expert system architecture were presented, the inference engine and the user interface.

The user interface allows the user to define requirements, constraints, permitted deviations and weights for the rating functions. It further allows to start the inference engine, which uses a novel Multi-Stage-Reasoning Process that combines ontological reasoning with a graph-based approach. The process is based on a bottom-up traversal of the Multi-Stage Design Graph resulting from the formalization process: Partial solutions composed of catalog components and solution principles are generated. Based on requirement checks, these partial solutions are either discarded or combined in a bottom-up way to finally generate valid overall design solutions. These design solutions are visualized by the user interface and can be ranked by different properties and rating functions.

By automatically selecting and arranging catalog components, the expert system can support large parts of the design process. Compared to related work in the field of automated design of robotic and mechatronic design (section 2.3), it avoids typical simplifications such as the use of modules instead of catalog components or the adaptation of only a single, parameterized solution.

Evaluation Using the Example of the Humanoid Robot ARMAR-6

The evaluation of the presented approach using the example of components for the collaborative humanoid robot ARMAR-6 is the third contribution of this work. This includes the design of mechatronic components for ARMAR-6, most notably sensor-actuator-controller (SAC) units for robot joints, which were designed in three sizes. Each SAC unit integrates a motor, a gearbox and a large sensor setup as well as communication and control electronics. This leads to a high degree of integration, even compared to related work (section 2.4).

After the presentation of ARMAR-6 (section 5.1), the approach presented in this thesis was evaluated through two case studies on the robot component types of ARMAR-6 with the highest degree of integration: SA/SAC units for robot joints (section 5.2) and robotic/prosthetic hands (section 5.3). For both case studies it was described how the design knowledge on existing robot components was formalized and used to develop an ontology-based expert system. The expert system for SA units takes 29 solution principles for 11 conceptual design parameters into account, including 4 parameters that describe different subcomponent arrangements. The expert system for robotic/prosthetic hands considers 30 solution principles for 13 parameters, including 2 parameters describing different subcomponent arrangements. Both expert systems use a common Component Ontology with data on about 500 catalog components. Taking all valid combinations of catalog components and solution principles into account, the expert systems are able to generate more than 10^8 solutions for SA/SAC units and 10^{13} solutions for robotic/prosthetic hands.

To demonstrate that the expert systems are capable of preserving knowledge, the five SA/SAC units and the three hands that served as knowledge sources were successfully reproduced. For the SA unit expert system it was further demonstrated that the system can accurately reproduce 9 of 10 state-of-the-art SA/SAC units. This showed that the generalized model and the Component Ontology are extensive enough to cover most rotary SA/SAC unit designs.

To show that the expert systems are not only capable of preserving knowledge but also of supporting future developments, existing designs were optimized according to different performance requirements. In the case of the KIT Prosthetic Hand V2, the design was also scaled for different hand sizes. While some optimizations did not surprise, there were also examples of generated solutions that a human engineer would not necessarily have come up with. This can be explained by the complex chain of dependencies for the selection of catalog components and solution principles as well as the multitude of possible combinations. Finally, it was demonstrated that the expert systems are also capable of generating novel design solutions for completely new requirements. The systems were able to achieve this goal by combining solution principles and catalog components from different knowledge sources in a novel way.

The case studies highlighted a further aspect in which the systematic approach and the resulting expert systems differ from most related work. Instead of having an educational application or creating small, low-cost, 3D printed robots, the field of application are highly integrated components for high-performance humanoid robots like ARMAR-6.

6.2. Discussion and Future Work

The systematic approach presented in this thesis and the resulting expert systems are first contributions to simplify future developments of humanoid robots. However, since this new field offers a lot of potential, this thesis can be seen as a starting point for future research. In the following, possible future work is discussed based on the limitations of the current approach.

Design of High-Level Components

The presented case studies, SA/SAC units and robotic hands, support the design of mid-level components of a humanoid robot. A logical next step would be to extend the expert system for high-level components such as a dual-arm system or a complete humanoid robot. Since higher-level components consist of mid-level and low-level components, a modular expert system framework is advantageous. A first step towards this modular framework has already been taken with the development of the robotic/prosthetic hand expert system that is able to use the EMGM expert system similar to a user. Likewise, for a dual-arm expert system, it would make sense to use the SA/SAC units expert system. However, since there are more and more possible combinations as the system level increases, the dual-arm expert system would benefit from an effective filter mechanism, which reduces the number of solutions provided by the SA/SAC unit expert system. This filter could be based on the existing rating functions, Pareto optimality or heuristic approaches. In addition to modular structures, the development of a high-level component such as a dual-arm system would also benefit from coupling the expert system with other software, e. g. a program for kinematics simulation or a parametrized CAD model to determine masses and inertias precisely.

Integration of the Knowledge of Different Design Groups

With few exceptions, the knowledge base currently used by the expert systems is based on knowledge gained during the development of humanoid robots and similar mechatronic systems at KIT. For future extensions of the expert system it would be advantageous to take greater account of the expert knowledge of other research institutions, ideally in close cooperation with the designers. Ultimately, the robot design expert system could also develop into an open source project. In this context, it would be advantageous if new components could be added more easily by anyone. Although the ontology can be extended comparatively easily by the Protégé editor, a special GUI would be ideal, which specifies what information needs to be entered for a new catalog component without requiring robot developers to become familiar with Protégé.

Flexible Adjustment

One of the major limitations of the approach is that the expert system is restricted to its knowledge base. Thus, when generating new design solutions, only combinations of known solution principles and catalog component series can be generated. If a new concept is to be used spontaneously, it must first be integrated into the knowledge base in order to make full use of the expert system. Although this problem probably cannot be completely solved, it can be reduced. The adjustable parameters in the GUI have proven to be a practical way to take new concepts into account. For example, the SA/SAC unit expert system makes it possible to set the wall thickness and any desired material density for the structural parts. This way it can be tested whether it is worthwhile to switch to new materials. A possible new extension for the robotic/prosthetic hand expert system could be the possibility to specify a new underactuation mechanism directly in the GUI by its dimensions, ratio, weight and costs.

Requirements Definition

The expert system requires the specification of mostly technical requirements. But the identification of requirements itself requires a certain degree of expert knowledge. To make the expert system accessible to everyone, including novices, it would be helpful to start with high-level requirements or stakeholder expectations. In the case of a dual-arm system, these could be tasks that the system has to perform. An intuitive user interface could also help in this context. One possibility would be for the user to express the tasks and other high-level requirements verbally. With the help of speech recognition this could serve as input for the expert system. In case of missing specifications, the system uses default parameters or asks the user.

Appendix

A. Scope of the Ontological Knowledge Base

Table .1.: Scope of the ontological knowledge bases of the individual expert systems

Expert System	OWL Classes	Object Properties	Data Properties	SWRL Rules
SA Unit	980	115	216	273
EMGM	848	65	145	82
Robotic Hand	1344	97	331	258

Table .2 lists all 497 catalog components which are implemented in the *Component Ontology* in its current state. The Component Ontology is used by all expert systems and therefore comprises components for encoder-motor-gearbox combinations (EMGM), robotic hands and SA units.

Table .2.: Catalog components of the Component Ontology with manufacturers

Component	Manufacturer	Count
Gearbox	Harmonic Drive, HAINA, Faulhaber	288
Motor	TQ RoboDrive ILM, Kollmorgen TBM, Faulhaber	68
Increm. Encoder	AMS, Faulhaber	27
Drive Bearing	SBN	23
Output Bearing	THK, SBN EZO	23
Slip Ring	Senring, Moog	20
Screws and Nuts	Freely selectable (Standardized DIN-ISO)	18
Springs	Gutekunst, Febrotec	6
Brake	TQ RoboDrive	4
Absolute Encoder	Renishaw RLS AksIM	3
Miscellaneous	Bosch, Elmo, Densitron, Bola	17

B. Valid Solutions SA Units

$$\begin{aligned} n_{SAunits} = &(n_M \cdot n_{BR*} - n_{incomp,MBR}) \cdot (n_G \cdot (n_{SR} + n_{C*} - 1) - n_{incomp,GSR}) \\ &\cdot n_{D*} \cdot n_{O*} \cdot n_{M*} \cdot n_{MF*} \cdot n_{AE*} \cdot n_{TS*} \cdot n_{IM*} \cdot n_{SB*} \cdot n_{HC*} \end{aligned}$$

The number of valid SA units $n_{SAunits}$ depends on the number of solution principles and catalog components. There are a total of 29 solution principles, divided into 11 parameters: Drive Structure ($n_{D*} = 2$), Output Structure ($n_{O*} = 3$), Motor Controller ($n_{M*} = 3$), Cabling ($n_{C*} = 3$), Motor Feedback ($n_{MF*} = 4$), Absolute Encoder ($n_{AE*} = 2$), Torque Sensor ($n_{TS*} = 4$), IMU ($n_{IM*} = 2$), Sensor PCB ($n_{SB*} = 2$), Brake ($n_{BR*} = 2$) and Hide Cables ($n_{HC*} = 2$).

With regard to catalog components, only the number of valid motors ($n_M = 23$), gearboxes ($n_G = 94$) and slip rings ($n_{SR} = 20$) is important. Through the rules, for each combination of motors, gearboxes, slip rings and solution principles, exactly one catalog component is selected for each other type required. Since not all gearboxes are compatible with all slip rings ($n_{incomp,GSR} = 1038$) and there is no suitable brake for every motor ($n_{incomp,MBR} = 13$), incompatible solutions must be subtracted. This results in $n_{SAunits} = 156,625,920$ valid SA units.

C. Valid Solutions Robotic/Prosthetic Hands

$$\begin{aligned}
n_{MGM} =& n_M \cdot n_G - n_{incomp,MG} \\
n_{EMGM} =& n_{MGM} \cdot n_E - n_{incomp,ME} \\
n_{Hands} =& (n_{EMGM})^2 \cdot n_{ME,FM,F} \cdot \prod_{i \in U} n_i \\
U =& \{S*, J*, A*, K*, GL*, FP*, HB*, IM*, DS*, CA*, SC*\}
\end{aligned}$$

The number of valid design solutions for robotic/prosthetic hands depends on the number of motors, gearboxes, encoders, mechanisms and solution principles. The hand expert system can use $n_M = 45$ motors and $n_G = 194$ gearboxes, which theoretically results in 8,730 motor-gearbox matches, of which $n_{incomp,MG} = 5,646$ are not compatible. The remaining $n_{MGM} = 3,084$ valid MGM can be combined with $n_E = 26$ encoders. Of the 80,184 EMGMs, however, $n_{incomp,ME} = 54,282$ are not compatible, so that a total of $n_{EMGM} = 25,902$ valid EMGMs can be used.

The hand can integrate two different EMGM for the fingers and the thumb. This results in $(n_{EMGM})^2 = 670,913,604$ possible combinations. As shown in Table 5.17, there are $n_{ME,FM,F} = 8$ valid possibilities to use mechanisms (ME) for the hand, which are selected depending on the number of fingers (F) and finger motors (FM). For the remaining 11 parameters there are two solution principles each, resulting in $n_{Hands} = \prod_{i \in U} n_i = 2^{11}$ possibilities to combine these solution principles.

In total, this results in $n_{Hands} = 10,992,248,487,936 \approx 10^{13}$ valid design solutions for robotic/prosthetic hands. In this, the dimensions of the hand are not considered, because there is no discrete number of solutions due to the continuous scalability of the proportions.

D. Reproduction State-of-the-Art SA Units (Details)

Table .3.: Reproduction of state-of-the-art SA units (details)

SA unit			Parameters for solution principles	
Source	Size		Structural	Functional
WALK-MAN SEA[3] †	A	Req	D∗ O3 M2 C2	MF2 AE1 TS2 SB1 IM0 BR0 HC0
		Sol	D2 O3 M2 C2	MF2 AE1 TS2 SB1 IM0 BR0 HC0
	B	Req	D∗ O3 M2 C2	MF2 AE1 TS2 SB1 IM0 BR0 HC0
		Sol	D2 O3 M2 C2	MF2 AE1 TS2 SB1 IM0 BR0 HC0
	C	Req	D∗ O3 M0 C0	MF2 AE1 TS2 SB0 IM0 BR0 HC0
		Sol	D2 O3 M0 C0	MF2 AE1 TS2 SB0 IM0 BR0 HC0
NREC Drive Joint[4] †	NGT 200	Req	D∗ O3 M0 C1	MF1 AE1 TS2 SB0 IM0 BR1 HC0
		Sol	D2 O3 M0 C1	MF1 AE1 TS2 SB0 IM0 BR1 HC0
	NGT 100	Req	D∗ O3 M0 C1	MF1 AE1 TS2 SB0 IM0 BR1 HC0
		Sol	D2 O3 M0 C1	MF1 AE1 TS2 SB0 IM0 BR1 HC0
	NGT 50	Req	D∗ O3 M0 C1	MF1 AE1 TS2 SB0 IM0 BR1 HC0
		Sol	D2 O3 M0 C1	MF1 AE1 TS2 SB0 IM0 BR1 HC0
	NGT 20	Req	D∗ O3 M0 C1	MF1 AE1 TS2 SB0 IM0 BR1 HC0
		Sol	D2 O3 M0 C1	MF1 AE1 TS2 SB0 IM0 BR1 HC0
ANYdrive[5]	-	Req	D∗ O∗ M1 C2	MF∗ AE1 TS∗ SB1 IM0 BR0 HC1
		Sol	D2 O2 M1 C2	MF2 AE1 TS1 SB1 IM0 BR0 HC1
Gear Motor RD-HD[6]	50 x08	Req	D∗ O∗ M0 C2	MF∗ AE1 TS0 SB0 IM0 BR0 HC0
		Sol	D1 O2 M0 C2	MF2 AE1 TS0 SB0 IM0 BR0 HC0
	70 x10	Req	D∗ O∗ M0 C2	MF∗ AE1 TS0 SB0 IM0 BR0 HC0
		Sol	D2 O2 M0 C2	MF1 AE1 TS0 SB0 IM0 BR0 HC0
	85 x13	Req	D∗ O∗ M0 C2	MF∗ AE1 TS0 SB0 IM0 BR0 HC0
		Sol	D2 O2 M0 C2	MF1 AE1 TS0 SB0 IM0 BR0 HC0

[3](Negrello et al., 2015) [4](Stentz et al., 2015) [5](ANYbotics, 2017) [6](TQ-Systems, 2016)

Abbreviations: Req = Requirements, Sol = Best solution
"∗"= Any option except from "none"; see subsection 5.2.2 for solution principles
Settings for motor-gearbox matching: $T_{M,p} = 0.5\ T_{M,max}$ and $T_{G,p} = T_{G,repeated}$, unless source (†) utilizes $T_{G,p} = T_{G,momentary}$ for gearbox selection
Settings for material and structure: 100% aluminum alloys, wall thickness $t_w = 2$

List of Figures

List of Tables

List of Algorithms

Acronyms

AI Artificial intelligence

BDD Block definition diagram
BLDC Brushless DC (direct current)
BMBF Bundesministerium für Bildung und Forschung (German Federal Ministry of Education and Research)

CAD Computer-aided design
CMU Carnegie Mellon University
CORA Core Ontology for Robotics and Automation

DAG Directed acyclic graph
DARPA Defense Advanced Research Projects Agency
DIN Deutsches Institut für Normung (German Institute for Standardization)
DLR Deutsches Zentrum für Luft- und Raumfahrt
DoF Degree of Freedom

EMGM Encoder-Motor-Gearbox Match

FMA Foundational Model of Anatomy

GUI Graphical user interface

HRC Humanoid robot component
HRI Human-robot interaction

IIT Istituto Italiano di Tecnologia (Italian Institute of Technology)
IMU Inertial Measurement Unit
ISO International Organization for Standardization

KIT Karlsruher Institut für Technologie (Karlsruhe Institute of Technology)

MGM Motor-Gearbox Match

NASA National Aeronautics and Space Administration
NRMSD Normalized Root-Mean-Square Deviation

OWL Web Ontology Language

PCB Printed circuit board
PX Performance Index

RRLab Robotics Research Lab TU Kaiserslautern

SA Sensor-actuator
SAC Sensor-actuator-controller
SCARA Selective Compliance Assembly Robot Arm
SE Systems engineering
SEA Series Elastic Actuator
SUMO Suggested Upper Merged Ontology
SWRL Semantic Web Rule Language
SysML Systems Modeling Language

TUAT Tokyo University of Agriculture and Technology

UI User interface
UML Unified Modeling Language

VDI Verein Deutscher Ingenieure (Association of German Engineers)

Bibliography

Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI communications*, 7(1):39–59. *Cited on page 22.*

Abburu, S. (2012). A Survey on Ontology Reasoners and Comparison. *International Journal of Computer Applications*, 57(17). *Cited on pages 27 and 28.*

Abraham, A. (2005). Rule-based Expert Systems. *Handbook of Measuring System Design*. *Cited on page 20.*

Agrawal, V. P., Gupta, S., and Kohli, V. (1991). Computer aided robot selection: The 'multiple attribute decision making' approach. *International Journal of Production Research*, 29(8):1629–1644. *Cited on pages 30, 42, and 43.*

Albu-Schäffer, A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. (2007). The DLR Lightweight Robot - Design and Control Concepts for Robots in Human Environments. 34(5):376–385. *Cited on pages 45, 46, and 47.*

Álvarez, A. and Ritchey, T. (2015). Applications of General Morphological Analysis. *Acta Morphologica Generalis*, 4(1). *Cited on page 15.*

ANYbotics (2017). ANYdrive robot joint: Integrated, robust, torque controllable. `http://www.anybotics.com`. *Cited on pages 45, 46, 118, and 164.*

Asfour, T. (2003). *Sensomotorische Bewegungskoordination zur Handlungsausführung eines humanoiden Roboters*. Ph.D. thesis, Department of Informatics, University of Karlsruhe. *Cited on page 94.*

Asfour, T., Berns, K., and Dillmann, R. (2000). The Humanoid Robot ARMAR: Design and Control. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 7–8. IEEE. *Cited on page 129.*

Asfour, T., Dillmann, R., Vahrenkamp, N., Do, M., Wächter, M., Mandery, C., Kaiser, P., Kröhnert, M., and Grotz, M. (2019a). The Karlsruhe ARMAR Humanoid Robot Family. In Goswami, A. and Vadakkepat, P., editors, *Humanoid Robotics: A Reference*, pages 337–368. Springer. *Cited on page 92.*

Asfour, T., Kaul, L., Wächter, M., Ottenhaus, S., Weiner, P., Rader, S., Grimm, R., Zhou, Y., Grotz, M., Paus, F., Shingarey, D., and Haubert, H. (2018). ARMAR-6: A Collaborative Humanoid Robot for Industrial Environments. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 447–454. IEEE. *Cited on pages 5 and 98.*

Asfour, T., Schill, J., Peters, H., Klas, C., Bucker, J., Sander, C., Schulz, S., Kargov, A., Werner, T., and Bartenbach, V. (2013). ARMAR-4: A 63 DOF Torque Controlled Humanoid Robot. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 390–396. IEEE. *Cited on pages 45, 46, 47, 95, 99, 105, 106, and 117.*

Asfour, T., Wächter, M., Kaul, L., Rader, S., Weiner, P., Ottenhaus, S., Grimm, R., Zhou, Y., Grotz, M., and Paus, F. (2019b). ARMAR-6: A High-Performance Humanoid for Human-Robot Collaboration in Real-World Scenarios. *IEEE Robotics & Automation Magazine*, 26:108–121. *Cited on pages 5, 92, 93, 96, 98, and 142.*

Auerbach, J. E. and Bongard, J. C. (2011). Evolving Complete Robots with CPPN-NEAT: The Utility of Recurrent Connections. In *13th Annual Conference on Genetic and Evolutionary Computation*, pages 1475–1482. ACM. *Cited on pages 33 and 34.*

Baader, F., Horrocks, I., and Sattler, U. (2008). Description Logics. *Foundations of Artificial Intelligence*, 3:135–179. *Cited on pages 21 and 22.*

Baccelliere, L., Kashiri, N., Muratore, L., Laurenzi, A., Kamedula, M., Margan, A., Cordasco, S., Malzahn, J., and Tsagarakis, N. G. (2017). Development of a Human Size and Strength Compliant Bi-Manual Platform for Realistic Heavy Manipulation Tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5594–5601. IEEE. *Cited on pages 46 and 47.*

Ben-Gal, I. (2008). Bayesian Networks. *Encyclopedia of statistics in quality and reliability*, 1. *Cited on page 23.*

Bhatia, P., Thirunarayanan, J., and Dave, N. (1998). An expert system-based design of SCARA robot. *Expert Systems with Applications*, 15(1):99–109. *Cited on pages 35, 39, 40, and 42.*

Bongard, J. (2010). The Utility of Evolving Simulated Robot Morphology Increases with Task Complexity for Object Manipulation. *Artificial Life*, 16(3):201–223. *Cited on pages 33 and 34.*

Borràs, J., Heudorfer, R., Rader, S., Kaiser, P., and Asfour, T. (2018). The KIT Swiss Knife Gripper for Disassembly Tasks: A Multi-Functional Gripper for Bimanual Manipulation with a Single Arm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4590–4597. IEEE. *Cited on page 95.*

Borst, W. N. (1997). Construction of Engineering Ontologies for Knowledge Sharing and Reuse (PhD Thesis, University of Twente, Enschede). *Cited on page 23.*

Boubekri, N., Sahoui, M., and Lakrib, C. (1991). Development of an expert system for industrial robot selection. *Computers & Industrial Engineering*, 20(1):119–127. *Cited on pages 20, 30, 42, and 43.*

Buchanan, B., Sutherland, G., and Feigenbaum, E. A. (1968). *HEURISTIC DENDRAL: a Program for Generating Explanatory Hypotheses in Organic Chemistry*. Stanford University Stanford. *Cited on page 16.*

Campbell, M. I., Cagan, J., and Kotovsky, K. (1999). A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment. *Research in Engineering Design*, 11(3):172–192. *Cited on pages 31, 32, 42, and 43.*

Canaday, B., Zapolsky, S., and Drumwright, E. (2017). Interactive, Iterative Robot Design. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1188–1195. IEEE. *Cited on pages 35 and 36.*

Chen, Y., Feng, P., He, B., Lin, Z., and Xie, Y. (2006). Automated Conceptual Design of Mechanisms Using Improved Morphological Matrix. *Journal of Mechanical Design*, 128(3):516–526. *Cited on pages 30, 31, 42, and 43.*

Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2013). Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding. *ACM SIGEVOlution*, 7(1):11–23. *Cited on page 33.*

Chew, M., Issa, G. F., and Shen, S. N. T. (1991). Expert System for Robot Hand Design Using Graph Representation. In *CAD/CAM Robotics and Factories of the Future'90*, pages 466–471. Springer. *Cited on pages 33, 42, and 43.*

Chiou, S.-J. and Sridhar, K. (1999). Automated conceptual design of mechanisms. *Mechanism and Machine Theory*, 34(3):467–495. *Cited on pages 30, 31, 42, and 43.*

Clancey, W. J. (1985). Heuristic Classification. *Artificial Intelligence*, 27(3):289–350. *Cited on pages 16, 17, and 167.*

Craft, M., Howsman, T., O'Neil, D., and Howell, J. T. (2002). Evolutionary Design and Simulation of a Tube Crawling Inspection Robot. *Cited on page 34.*

DARPA (2015). DARPA Robotics Challenge Finals 2015. `https://archive.darpa.mil/roboticschallenge/`. *Cited on page 45.*

Datta, R. and Deb, K. (2011). Optimizing and Deciphering Design Principles of Robot Gripper Configurations Using an Evolutionary Multi-Objective Optimization Method. Technical report, India, Technical report. *Cited on page 33.*

Davis, R., Shrobe, H., and Szolovits, P. (1993). What Is a Knowledge Representation? *AI Magazine*, 14(1):17. *Cited on page 19.*

Delligatti, L. (2014). *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Addison-Wesley. *Cited on pages 12, 13, and 14.*

Desai, R., McCann, J., and Coros, S. (2018a). Assembly-aware Design of Printable Electromechanical Devices. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, pages 457–472. ACM. *Cited on pages 35, 37, 38, and 39.*

Desai, R., Safonova, M., Muelling, K., and Coros, S. (2018b). Automatic Design of Task-specific Robotic Arms. *arXiv preprint arXiv:1806.07419*. *Cited on pages 35 and 38.*

Desai, R., Yuan, Y., and Coros, S. (2017). Computational Abstractions for Interactive Design of Robotic Devices. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1196–1203. IEEE. *Cited on pages 35, 37, and 39.*

Diftler, M. A., Mehling, J. S., Abdallah, M. E., Radford, N. A., Bridgwater, L. B., Sanders, A. M., Askew, R. S., Linn, D. M., Yamokoski, J. D., Permenter, F. A., Hargrave, B. K., Platt, R., Savely, R. T., and Ambrose, R. O. (2011). Robonaut

2 - The First Humanoid Robot in Space. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2178–2183. IEEE. *Cited on page 47.*

Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. G. (2015). Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI*, 2:4. *Cited on page 34.*

El-Nakla, S. (2012). Case-Based Expert System to Support Electronics Design in Mechatronic System. In *Applied Mechanics and Materials*, volume 229, pages 2793–2797. Trans Tech Publ. *Cited on pages 32 and 42.*

Englsberger, J., Werner, A., Ott, C., Henze, B., Roa, M. A., Garofalo, G., Burger, R., Beyer, A., Eiberger, O., Schmid, K., and Albu-Schäffer, A. (2014). Overview of the Torque-Controlled Humanoid Robot TORO. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 916–923. IEEE. *Cited on page 47.*

Erdman, A. G., Thompson, T., and Riley, D. R. (1986). Type Selection of Robot and Gripper Kinematic Topology Using Expert Systems. *The International Journal of Robotics Research*, 5(2):183–189. *Cited on pages 33, 42, and 43.*

Eriksson, T. and Ritchey, T. (2002). Scenario Development using Computerised Morphological Analysis. In *Adapted from papers presented at the cornwallis and winchester international OR conferences.—England*. *Cited on page 15.*

Förster, D. (2019). *Expertensystem für die Entwicklung von Roboterhänden und Prothesen*. Bachelor thesis, Karlsruhe Institute of Technology (KIT). Supervisor: S. Rader. *Cited on page 134.*

Frutiger, D. R., Bongard, J. C., and Iida, F. (2002). Iterative Product Engineering: Evolutionary Robot Design. In *International Conference on Climbing and Walking Robots*, pages 619–629. Professional Engineering Publishing. *Cited on page 34.*

Fukaya, N., Asfour, T., Dillmann, R., and Toyama, S. (2013). Development of a Five-Finger Dexterous Hand without Feedback Control: the TUAT/Karlsruhe Humanoid Hand. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4533–4540. IEEE. *Cited on pages 96 and 129.*

Fukaya, N., Toyama, S., Asfour, T., and Dillmann, R. (2000). Design of the TUAT/Karlsruhe Humanoid Hand. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1754–1759. IEEE, IEEE. *Cited on pages 96 and 129.*

Geilinger, M., Poranne, R., Desai, R., Thomaszewski, B., and Coros, S. (2018). Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels. *ACM Transactions on Graphics (TOG)*, 37(4):160. *Cited on pages 35 and 37.*

Görz, G. (1993). *Einführung in die künstliche Intelligenz*. Addison-Wesley Bonn, 1 edition. *Cited on pages 2 and 16.*

Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., and Sattler, U. (2008). OWL 2: The next step for OWL. *Web Semantics*, 6(4):309–322. *Cited on page 24.*

Gruber, T. (1993). A Translational Approach to Portable Ontology Specifications. Knowledge Systems Laboratory technical Report KSL 92-71. *Cited on page 23.*

Ha, S., Coros, S., Alspach, A., Bern, J. M., Kim, J., and Yamane, K. (2018). Computational Design of Robotic Devices From High-Level Motion Specifications. *IEEE Transactions on Robotics*, 34(5):1240–1251. *Cited on pages 35 and 38.*

Haarslev, V., Hidde, K., Möller, R., and Wessel, M. (2012). The RacerPro Knowledge Representation and Reasoning System. *Semantic Web Journal*, 3(3):267–277. *Cited on page 27.*

Haidegger, T., Barreto, M., Gonçalves, P., Habib, M. K., Ragavan, S. K. V., Li, H., Vaccarella, A., Perrone, R., and Prestes, E. (2013). Applied ontologies and standards for service robots. *Robotics and Autonomous Systems*, 61(11):1215–1223. *Cited on page 26.*

Harmonic Drive AG (2016). Harmonic Drive General Catalogue. `http://harmonicdrive.de`. *Cited on pages 45 and 46.*

HEBI Robotics (2019). X-Series Actuator - Technical Specifications. `http://www.hebirobotics.com`. *Cited on pages 45 and 46.*

Hiller, J. and Lipson, H. (2012). Automatic Design and Manufacture of Soft Robots. *IEEE Transactions on Robotics*, 28(2):457–466. *Cited on pages 33 and 34.*

Hornby, G. S., Lipson, H., and Pollack, J. B. (2003). Generative Representations for the Automated Design of Modular Physical Robots. *IEEE Transactions on Robotics and Automation*, 19(4):703–719. *Cited on pages 33 and 34.*

Horridge, M. and Bechhofer, S. (2011). The OWL API: A Java API for OWL Ontologies. *Semantic Web Journal*, 2(1):11–21. *Cited on page 77.*

Horridge, M., Knublauch, H., Rector, A., Stevens, R., and Wroe, C. (2004). A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0. *University of Manchester*. *Cited on page 24.*

Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a Web Ontology Language. *Web Semantics*, 1(1):7–26. *Cited on page 24.*

Hu, H., Liu, D.-y., and Du, X.-y. (2004). Semi-automatic hardware design using ontologies. In *International Conference on Control, Automation, Robotics and Vision Conference (ICARCV)*, volume 2, pages 792–797. IEEE. *Cited on page 65.*

IEEE Robotics and Automation Society (2015). *IEEE Standard Ontologies for Robotics and Automation*. Institute of Electrical and Electronics Engineers, New York. *Cited on pages 25 and 26.*

Jakus, G., Milutinović, V., Omerović, S., and Tomažič, S. (2013). Knowledge Representation. In *Concepts, Ontologies, and Knowledge Representation*, pages 47–62. Springer. *Cited on pages 19, 21, 22, 23, and 27.*

Juarez, A., Bartneck, C., and Feijs, L. (2011). Using Semantic Technologies to Describe Robotic Embodiments. In *International Conference on Human-Robot Interaction*, pages 425–432. *Cited on pages 26 and 65.*

Jürgens, H. W. (2004). *Erhebung anthropometrischer Maße zur Aktualisierung der DIN 33 402-Teil 2*. Wirtschaftsverl. NW, Verlag für Neue Wiss. *Cited on pages 70 and 130.*

Kapurch, S. J. (2010). *NASA Systems Engineering Handbook*. Diane Publishing. *Cited on pages 2, 10, 11, 12, 59, and 60.*

Karrenbauer, O., Rader, S., and Asfour, T. (2018). An Ontology-Based Expert System to Support the Design of Humanoid Robot Components. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 532–539. IEEE. The first two authors contributed equally to this work. *Cited on pages 50, 67, 75, 106, 107, and 110.*

Karsak, E. E. (2007). Expert Decision System for Robot Selection. *Wiley Encyclopedia of Computer Science and Engineering*, pages 1–11. *Cited on pages 30, 42, and 43.*

Keller, S., Hausmann, R., Kressner, L., and Koenig, A. (2016). An approach of a computerized planning assistant to the system design of collaborative robot installations. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, [Piscataway, New Jersey]. IEEE, IEEE. *Cited on pages 22, 30, 42, and 43.*

Kinova Robotics (2015). K-Series Technical specifications. `http://www.kinovarobotics.com`. *Cited on page 46.*

Komoto, H. and Tomiyama, T. (2012). A framework for computer-aided conceptual design and its application to system architecting of mechatronics products. *Computer-Aided Design*, 44(10):931–946. *Cited on pages 31 and 42.*

Koulouriotis, D. E. and Ketipi, M. K. (2014). Robot evaluation and selection Part A: an integrated review and annotated taxonomy. *The International Journal of Advanced Manufacturing Technology*, 71(5-8):1371–1394. *Cited on page 30.*

Laugis, J. and Vodovozov, V. (2008). Expert System for Electric Drive Design. In *IEEE International Power Electronics and Motion Control Conference*, pages 1017–1019. IEEE. *Cited on pages 35, 39, 40, 42, and 43.*

Lee, W.-P. (1998). An Evolutionary System for Automatic Robot Design. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 3477–3482. IEEE. *Cited on page 34.*

Leger, C. et al. (1999). *Automated Synthesis and Optimization of Robot Configurations: An Evolutionary Approach*. Carnegie Mellon University USA. *Cited on page 33.*

Li, W., Li, Y., Wang, J., and Liu, X. (2010). The process model to aid innovation of products conceptual design. *Expert Systems with Applications*, 37(5):3574–3587. *Cited on page 31.*

Lipson, H. and Pollack, J. B. (2000). Towards Continuously Reconfigurable Self-Designing Robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1761–1766. IEEE. *Cited on pages 33 and 34.*

Maher, M. L., Sriram, D., and Fenves, S. J. (1984). Tools and Techniques for Knowledge Based Expert Systems for Engineering Design. *Advances in Engineering Software (1978)*, 6(4):178–188. *Cited on pages 18, 20, and 21.*

Megaro, V., Thomaszewski, B., Nitti, M., Hilliges, O., Gross, M., and Coros, S. (2015). Interactive Design of 3D-Printable Robotic Creatures. *ACM Transactions on Graphics (TOG)*, 34(6):216. *Cited on pages 35 and 38.*

Mehta, A., DelPreto, J., and Rus, D. (2015). Integrated Codesign of Printable Robots. *Journal of Mechanisms and Robotics*, 7(2):021015. *Cited on pages 35, 37, 38, and 39.*

Meixner, A., Hazard, C., and Pollard, N. (2019). Automated Design of Simple and Robust Manipulators for Dexterous In-Hand Manipulation Tasks using Evolutionary Strategies. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. *Cited on pages 33 and 34.*

Minsky, M. (1975). A Framework for Representing Knowledge. *The Psychology of Computer Vision*. *Cited on page 21.*

Myung, S. and Han, S. (2001). Knowledge-based parametric design of mechanical products based on configuration design method. *Expert Systems with Applications*, 21(2):99–107. *Cited on pages 35, 39, 40, and 42.*

Nagori, V. and Trivedi, B. (2014). Types of Expert System: Comparative Study. *Asian Journal of Computer and Information Systems*, 2(02). *Cited on pages 20 and 21.*

Negrello, F., Garabini, M., Catalano, M. G., Malzahn, J., Caldwell, D. G., Bicchi, A., and Tsagarakis, N. G. (2015). A Modular Compliant Actuator for Emerging High Performance and Fall-Resilient Humanoids. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 414–420. IEEE. *Cited on pages 45, 46, 47, 105, 106, 118, and 164.*

Nilsson, A., Muradore, R., Nilsson, K., and Fiorini, P. (2009). Ontology for Robotics: a Roadmap. In *The International Conference on Advanced Robotics (ICAR)*. *Cited on page 65.*

Northover, S. and Wilson, M. (2004). *SWT: The Standard Widget Toolkit, Volume 1*. Addison-Wesley Professional. *Cited on page 84.*

Novák, V., Perfilieva, I., and Mockor, J. (1999). *Mathematical Principles of Fuzzy Logic*, volume 517. Springer Science & Business Media. *Cited on page 23.*

Noy, N. F., Fergerson, R. W., and Musen, M. A. (2000). The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In Dieng, R. and Corby, O., editors, *Knowledge engineering and knowledge management*, volume

1937 of *Lecture notes in computer science. Lecture notes in artificial intelligence*, pages 17–32. Springer, Berlin and London. *Cited on pages 65 and 72.*

Nygaard, T. F., Samuelsen, E., and Glette, K. (2017). Overcoming Initial Convergence in Multi-Objective Evolution of Robot Control and Morphology Using a Two-Phase Approach. In *European Conference on the Applications of Evolutionary Computation*, pages 825–836. Springer. *Cited on page 34.*

O'Connor, M., Knublauch, H., Tu, S., Grosof, B., Dean, M., Grosso, W., and Musen, M. (2005). Supporting Rule System Interoperability on the Semantic Web with SWRL. In Gil, Y., editor, *The Semantic Web ISWC*, volume 3729 of *Lecture notes in computer science, 0302-9743*, pages 974–986. Springer Berlin Heidelberg, New York and Springer, 2005. Great Britain. *Cited on pages 24 and 64.*

Øhrstrøm, P., Andersen, J., and Schärfe, H. (2005). What Has Happened to Ontology. In *International Conference on Conceptual Structures*, pages 425–438. Springer. *Cited on page 23.*

Olier, E. (1985). Conceptual Design for Space Robots Using Expert Systems. *IFAC Proceedings Volumes*, 18(16):145–150. *Cited on pages 32 and 42.*

Pahl, G., Beitz, W., Feldhusen, J., and Grote, K.-H. (2007). *Engineering Design: A Systematic Approach*. Springer. *Cited on pages 1, 7, 8, 9, 10, 12, 15, and 48.*

Pease, A., Niles, I., and Li, J. (2002). The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications. In *Working notes of the AAAI-2002 workshop on ontologies and the semantic web*, volume 28. *Cited on page 25.*

Pfeifer, R., Iida, F., and Bongard, J. (2005). New Robotics: Design Principles for Intelligent Systems. *Artificial Life*, 11(1-2):99–120. *Cited on page 33.*

Pham, D. T. and Tacgin, E. (1991). DBGRIP: A learning expert system for detailed selection of robot grippers. *The International Journal of Production Research*, 29(8):1549–1563. *Cited on pages 30 and 42.*

Rader, S., Kaul, L., Fischbach, H., Vahrenkamp, N., and Asfour, T. (2016). Design of a High-Performance Humanoid Dual Arm System with Inner Shoulder Joints. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 523–529. IEEE. *Cited on pages 92, 94, and 95.*

Rader, S., Kaul, L., Weiner, P., and Asfour, T. (2017). Highly Integrated Sensor-Actuator-Controller Units for Modular Robot Design. In *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1160–1166. IEEE. *Cited on pages 5, 44, 92, 95, 99, 100, 102, 104, and 117.*

Radford, N. A., Strawser, P., Hambuchen, K., Mehling, J. S., Verdeyen, W. K., Donnan, S., Holley, J., Sanchez, J., Nguyen, V., Bridgwater, L., et al. (2015). Valkyrie: NASA's First Bipedal Humanoid Robot. 32(3):397–419. *Cited on pages 45, 46, and 47.*

Ramos, F., Olivares-Alarcos, A., Vázquez, A. S., and Fernández, R. (2017). What Can Ontologies Do for Robot Design? In *Iberian Robotics Conference*, pages 465–476. Springer. *Cited on pages 26 and 65.*

Ramos, F., Vázquez, A. S., Fernández, R., Olivares-Alarcos, A., Schlenoff, C., Balakirsky, S., and Christensen, H. (2018). Ontology based design, control and programming of modular robots. *Integrated Computer-Aided Engineering*, 25(2):173–192. *Cited on pages 26, 35, 38, 39, 42, and 65.*

Ritchey, T. (1998). General Morphological Analysis. In *16th Euro Conference on Operational Analysis*. *Cited on page 15.*

Robotis (2015). Dynamixel Pro. `http://www.robotis.com`. *Cited on pages 45, 46, and 47.*

Roehr, T. M., Harnack, D., Lima, O., Wöhrle, H., and Kirchner, F. (2019). Introducing q-rock : Towards the automated self-exploration and qualification of robot behaviors. In *Poster at the ICRA 2019 Workshop on Robot Design and Customization: Opportunities at the Intersection of Computation and Digital Fabrication. International Conference on Robotics and Automation (ICRA 2019)*, pages 1–2. *Cited on page 41.*

Römmerman, M., Kuhn, D., and Kirchner, F. (2009). Robot Design for Space Missions Using Evolutionary Computation. In *IEEE Congress on Evolutionary Computation*, pages 2098–2105. IEEE. *Cited on page 34.*

Rosse, C. and Mejino, J. L. V. (2008). The Foundational Model of Anatomy Ontology. In *Anatomy Ontologies for Bioinformatics*, pages 59–117. Springer. *Cited on page 66.*

Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice-Hall series in artificial intelligence. Prentice-Hall, Upper Saddle River, NJ, 3. edition. *Cited on pages 16, 22, and 23.*

Russell, S. J. and Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited,. *Cited on page 24.*

Samuelsen, E. and Glette, K. (2015). Real-World Reproduction of Evolved Robot Morphologies: Automated Categorization and Evaluation. In *European Conference on the Applications of Evolutionary Computation*, pages 771–782. Springer. *Cited on page 34.*

Saravanan, R., Ramabalan, S., Ebenezer, N. G. R., and Dharmaraja, C. (2009). Evolutionary multi criteria design optimization of robot grippers. *Applied Soft Computing*, 9(1):159–172. *Cited on page 33.*

Schulz, A., Sung, C., Spielberg, A., Zhao, W., Cheng, R., Grinspun, E., Rus, D., and Matusik, W. (2017). Interactive robogami: An end-to-end system for design of robots with ground locomotion. *The International Journal of Robotics Research*, 36(10):1131–1147. *Cited on pages 35, 37, and 39.*

Schunk (2011). Montage- und Betriebsanleitung Leichtbaumodul Typ PRL. `https://schunk.com`. *Cited on pages 45 and 46.*

Schütz, S., Mianowski, K., Kotting, C., Nejadfard, A., Reichardt, M., and Berns, K. (2016). RRLAB SEA — A Highly Integrated Compliant Actuator with Minimised Reflected Inertia. In *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 252–257. IEEE. *Cited on page 44.*

Schütz, S., Nejadfard, A., Mianowski, K., Vonwirth, P., and Berns, K. (2017). CARL — A Compliant Robotic Leg Featuring Mono- and Biarticular Actuation. In *IEEE/RAS International Conference on Humanoid Robotics (Humanoids)*, pages 289–296. IEEE. *Cited on page 44.*

SENSODRIVE (2020). Senso-joints: Torque-controlled actuator. `https://www.sensodrive.de`. *Cited on pages 45 and 46.*

Shiakolas, P. S., Koladiya, D., and Kebrle, J. (2002). Optimum Robot Design Based on Task Specifications Using Evolutionary Techniques and Kinematic, Dynamic, and Structural Constraints. *Inverse Problems in Engineering*, 10(4):359–375. *Cited on page 33.*

Shortliffe, E. H. (1974). MYCIN: a rule-based computer program for advising physicians regarding antimicrobial therapy selection. Technical report, Stanford University California Department of Computer Science. *Cited on page 16.*

Sims, K. (1994). Evolving 3D Morphology and Behavior by Competition. *Artificial life*, 1(4):353–372. *Cited on page 33.*

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5(2):51–53. *Cited on pages 27 and 77.*

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., et al. (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255. *Cited on page 65.*

Spielberg, A., Araki, B., Sung, C., Tedrake, R., and Rus, D. (2017). Functional Co-Optimization Of Articulated Robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5042. IEEE. *Cited on pages 35 and 36.*

Staab, S. and Studer, R. (2004). *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, Berlin and London. *Cited on page 24.*

Stentz, A., Herman, H., Kelly, A., Meyhofer, E., Haynes, G. C., Stager, D., Zajac, B., Bagnell, J. A., Brindza, J., Dellin, C., George, M., Gonzalez-Mora, J., Hyde, S., Jones, M., Laverne, M., Likhachev, M., Lister, L., Powers, M., Ramos, O., Ray, J., Rice, D., Scheifflee, J., Sidki, R., Srinivasa, S., Strabala, K., Tardif, J.-P., Valois, J.-S., Weghe, J. M. V., Wagner, M., and Wellington, C. (2015). CHIMP: The CMU Highly Intelligent Mobile Platform. 32(2):209–228. *Cited on pages 45, 46, 47, 118, and 164.*

Studer, R., Benjamins, V. R., and Fensel, D. (1998). Knowledge Engineering: Principles and Methods. *Data & Knowledge Engineering*, 25(1-2):161–197. *Cited on page 23.*

Styczynski, Z. A., Rudion, K., and Naumann, A. (2017). *Einführung in Expertensysteme: Grundlagen, Anwendungen und Beispiele aus der elektrischen Energieversorgung*. Springer-Verlag. *Cited on pages 17, 20, and 21.*

Tanev, I., Ray, T., and Buller, A. (2005). Automated Evolutionary Design, Robustness, and Adaptation of Sidewinding Locomotion of a Simulated Snake-Like Robot. *IEEE Transactions on Robotics*, 21(4):632–645. *Cited on page 33.*

TQ-Systems (2016). RD50/70/85-HD Gear motors. `https://www.tq-group.com`. *Cited on pages 45, 46, 118, and 164.*

Tripathi, K. P. (2011). A Review on Knowledge-based Expert System: Concept and Architecture. *IJCA Special Issue on Artificial Intelligence Techniques-Novel Approaches & Practical Applications*, 4:19–23. *Cited on pages 16, 18, and 19.*

Tsagarakis, N. G., Laffranchi, M., Vanderborght, B., and Caldwell, D. G. (2009). A Compact Soft Actuator Unit for Small Scale Human Friendly Robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4356–4362. IEEE. *Cited on page 47.*

Tsarkov, D. and Horrocks, I. (2006). FaCT++ Description Logic Reasoner: System Description. In Furbach, U. and Shankar, N., editors, *Automated reasoning*, volume 4130 of *Lecture notes in computer science. Lecture notes in artificial intelligence, 0302-9743*, pages 292–297. Springer, Berlin and London. *Cited on page 27.*

Vahrenkamp, N., Wächter, M., Kröhnert, M., Welke, K., and Asfour, T. (2015). The Robot Software Framework ArmarX. *it-Information Technology*, 57(2):99–111. *Cited on page 98.*

Vanderborght, B., Albu-Schäffer, A., Bicchi, A., Burdet, E., Caldwell, D. G., Carloni, R., Catalano, M., Eiberger, O., Friedl, W., Ganesh, G., Garabini, M., Grebenstein, M., Grioli, G., Haddadin, S., Hoppner, H., Jafari, A., Laffranchi, M., Lefeber, D., Petit, F., Stramigioli, S., Tsagarakis, N., Damme, M. V., Ham, R. V., Visser, L. C., and Wolf, S. (2013). Variable Impedance Actuators: A Review. 61(12):1601–1614. *Cited on page 47.*

VDI (1993). *VDI 2221 Methodik zum Entwicklen und Konstruieren technischer Systeme und Produkte*. VDI Verlag. *Cited on pages 10 and 12.*

Vila-Rosado, D. N. and Domınguez-López, J. A. (2006). A MATLAB toolbox for the optimal design of robot manipulators using evolutionary techniques. *International Conference on Mechatronic Technology (ICMT)*. *Cited on page 33.*

Vo-Gia, L., Kashiri, N., Negrello, F., Tsagarakis, N. G., and Caldwell, D. G. (2014). Development of a 7DOF Soft Manipulator Arm for the Compliant Humanoid Robot COMAN. In *IEEE/RAS International Conference on Robotics and Biomimetics*, pages 1106–1111. IEEE. *Cited on page 47.*

Šaša Bastinos, A. and Krisper, M. (2013). Multi-criteria decision making in ontologies. *Information Sciences*, 222:593–610. *Cited on pages 28, 77, 78, and 90.*

Wagner, W. P. (2017). Trends in expert system development: A longitudinal content analysis of over thirty years of expert system case studies. *Expert Systems with Applications*, 76:85–96. *Cited on pages 16, 19, 20, 43, and 65.*

Wahl, J.-C., Sartor, M., Paredes, M., et al. (2003). A General Framework for Automated Conceptual Design of One DOF Mechanisms. In *International Conference on Engineering Design (ICED), Stockholm*, pages 17–18. *Cited on pages 30 and 31.*

Wang, Q., Chen, X., Yin, Y., and Lu, J. (2014). Ontology-Based Coupled Optimization Design Method Using State-Space Analysis for the Spindle Box System of Large Ultra-Precision Optical Grinding Machine. In *IEEE International Conference on Enterprise Systems*, pages 273–278. IEEE. *Cited on pages 35, 39, and 42.*

Weiner, P., Starke, J., Hundhausen, F., Beil, J., and Asfour, T. (2018). The KIT Prosthetic Hand: Design and Control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3328–3334. IEEE. *Cited on pages 129, 130, 132, 141, and 142.*

Whitman, J. and Choset, H. (2019). Task-Specific Manipulator Design and Trajectory Synthesis. *IEEE Robotics and Automation Letters*, 4(2):301–308. *Cited on pages 35 and 36.*

Wu, M.-C., Lo, Y.-F., and Hsu, S.-H. (2008). A fuzzy CBR technique for generating product ideas. *Expert Systems with Applications*, 34(1):530–540. *Cited on pages 33 and 42.*

Yang, B.-S., Han, T., and Kim, Y.-S. (2004). Integration of ART-Kohonen neural network and case-based reasoning for intelligent fault diagnosis. *Expert Systems with Applications*, 26(3):387–395. *Cited on page 23.*

Zheng, C., Eynard, B., Qin, X., Li, J., Bai, J., Gomes, S., and Zhang, Y. (2019). A requirement-driven architecture definition approach for conceptual design of mechatronic systems. *Integrated Computer-Aided Engineering*, (Preprint):1–22. *Cited on page 31.*

Ziglar, J., Williams, R., and Wicks, A. (2017). Context-Aware System Synthesis, Task Assignment, and Routing. *arXiv preprint arXiv:1706.04580. Cited on page 32.*

Zu, Y., Xiao, R., and Zhang, X. (2009). Automated conceptual design of mechanisms using enumeration and functional reasoning. *International Journal of Materials & Product Technology*, 34(3):273. *Cited on pages 30, 31, 42, and 43.*

Zwicky, F. (1948). Morphological Astronomy. *The Observatory*, 68:121–143. *Cited on page 14.*

Zwicky, F. and Wilson, A. G. (1967). New Methods of Thought and Procedure. *Springer Verlag*. *Cited on page 14.*