THESIS ON MECHANICAL AND INSTRUMENTAL
ENGINEERING E32

MODEL BASED MECHATRONIC SYSTEMS MODELING METHODOLOGY IN CONCEPTUAL DESIGN STAGE

RAIVO SELL

TALLINN 2007

Faculty of Mechanical Engineering
Department of Mechatronics
TALLINN UNIVERSITY OF TECHNOLOGY

Dissertation was accepted for the defence of the degree of Doctor of Philosophy in Engineering Sciences on May 30, 2007

Supervisor:

Mart Tamre, Prof., PhD,
Department of Mechatronics,
Tallinn University of Technology

Opponents:

Mauri Airila, Prof., D.Sc. (Tech.),
Helsinki University of Technology, Finland

Elmo Pettai, PhD,
Tallinn University of Technology, Estonia

Defence of the thesis: August 28, 2007

Declaration: Hereby I declare that this thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for any degree or examination.

Raivo Sell, _____

ISSN 1406-4758
ISBN 978-9985-59-718-7

MASINA- JA APARAADIEHITUS

MUDELITEL BASEERUV MEHHATROONIKASÜSTEEMIDE MODELLEERIMISE METOODIKA KONTSEPTUAALSE PROJEKTEERIMISE FAASIS

RAIVO SELL

Contents

INTRODUCTION

Background

Mechatronics and robotics are one of the most rapidly developing fields in technology sector today, and it grows constantly. Many mechatronics applications, like home robots were quite rear ten years ago, but are now available for consumers. Mechatronics as an engineering domain itself is not very new at all. The term mechatronics was introduced by a Japanese engineer from Yasukawa Electric Company in 1969. Initially it reflected the merge of mechanics and electronics. In the mid-1980s the control and software part started to play more important role in mechatronics engineering and the scope of mechatronics was extended. Today, the term mechatronics encompasses a large array of technologies including all sub-domains of mechanics, electronics and control. Although all classical domains have its own elements, the boundaries between them have been misty and the software role has increased drastically. Almost every mechatronics product has more or less embedded software and control elements. Mechatronics gained its legitimacy in academic circles in 1996 with the publication of the first refereed journal: IEEE/ASME Transactions on Mechatronics. In the premier issue, the authors worked hard to define mechatronics. The selected definition was the following: "The synergistic integration of mechanical engineering with electronics and intelligent computer control in the design and manufacturing of industrial products and processes" [Mech96]. Authors suggested 11 topics that should fall, at least in part, under the general category of mechatronics [Mech96]:

- modeling and design
- system integration
- actuators and sensors
- intelligent control
- robotics
- manufacturing
- motion control
- vibration and noise control
- micro devices and optoelectronic systems
- automotive systems
- other applications

Of course there exist lots of concurrent definitions, but the idea is more or less the same.

The term robot however is much older. Introduced by the Czech writer Karel Čapek in 1921, it became very popular in science fiction books and movies. Nowadays the robotics is one of the true applications of mechatronics. As all other mechatronic systems the robots have became smarter and more sophisticated thanks to the growth of software and computing power.

In frame of all these developments the design and creation process of a mechatronics product has turned very complex. Market however demands more and more smart products with shorter development time and smaller cost. This puts very high demands on mechatronic system development process and conventional methods are not competitive any more. Engineering decisions in different design stages affect the final product without any doubt. However decisions made in early design stage affect the result much more than later ones. Although almost every engineer agrees with that, the early stage design is the most unsupported phase in the product development process. No common methodology and tools are exploited contrary to, for example, domain specific design, where lots of mature methods and tools are available and almost always used. The mechatronic system development is even more affected by the lack of early design support because the high level component integration from dissimilar domains is always needed. The later integration means that the system developer team must ensure the component and sub-system compatibility in all the development stages, including the early stage. Rapidly advancing technology automation market demands continuously new robots which are in their nature true mechatronic products. Accordingly, the design methodology is also urgently needed with the full support of early design and decision making process in open design space.

The research object of this thesis is a model based mechatronics design methodology. The developed methodology deals with early design stage and supportive tools. The methodology is applied to the mobile robotic field as one of the most growing mechatronic application examples.
Very roughly all robotic applications can be divided as follows:
industrial robot – usually a manipulator type robot designed for a certain recurrent task. These kinds of robots are very widely used in automotive, electronic, assembling and many other industries. This is the most mature field of robotics;
humanoid robot – usually a bi-pedal walking machine, trying to simulate human behavior and locomotion. This type of robot is the common imagination of non-engineering people when talking about robots. They are much less useful than industrial robots and playing in most cases the entertainer role. A lot of research effort has been put in bi-pedal robot research and some success has been achieved but it is still far from a useful human partner. The most well known example is Honda ASIMO claimed as the most advanced bi-pedal humanoid in the world today [Honda06];
mobile robot – usually a wheeled autonomous or teleoperated vehicle. This robotic field has got a lot of attention from the last decade and many research and industrial results are in real use. The field is especially important for military domain where some of the application areas are antiterrorism and security. Mobile robots have got an explosive attention in home and service field even

more recently. This brings many new companies and research institutes to the robotic design and manufacturing field. Standardization and methodological design support is particularly important for the mobile robotic platform development.

The current thesis presents a general framework for mechatronics design methodology with emphasis to the early design. The implementation case is applied to mobile robotic section. The mobile robot section can also be divided into several sub-categories, depending on the purpose, size, locomotion schemas, etc. The methodology implementation can be extended with some restrictions to all these sub-categories, however the author has relied mostly on the mid size wheeled mobile robot as most common application. Most examples and implementations are based also on a hybrid wheeled mobile robot.

Actuality of the topic

The topic itself was selected based on the latest development of the mechatronics field and on a practical need and opportunities. As mechatronic product design has became more and more complex problem the need for new methodology has also steadily grown. From the other hand the increasing role of software in a mechatronic product sets additional requirements for the design methodology. This means that in a product development team software engineers have to work side-by-side with mechanical engineers and electrical engineer where the design methods are still quite different. Because of the boundaries of the software and hardware design being misty, the methodology of a system design must also be unified for all the sub-domains. This is especially challenging for conventional engineers who have been designed non-software related products.
Because of the complexity and time critical development, the early design has become an important design stage. Many recent papers and industry articles emphasize the growing importance of the early design and the need for development tools supporting the early design process [Balmelli06]. Another important concern is that the engineering design, especially mechatronic product design, is not a pure technical problem any more but a complex activity. There is a need to involve artifacts, people, environment, market, etc., where all the aspects have to be modeled in the same methodical way. The requirements in the area of the mechatronic system design, especially in robotic field where a control sub-system should be developed on a very high level, demand the quantities evaluation of different design alternatives. Existing tools like, for example, Unified Modeling Language (UML) have very wide modeling capabilities, however the early stage verification, conflict analyses and initial simulation are not currently covered. There are some early researches to fulfill this cap in

certain fields. System on Chip (SoC) is one of the most actively studied field and some approaches are presented in publications [Viehl06, VD06].
The modularization and model based engineering was adopted in software field many years ago and it is obtaining growing attention also in the system engineering field. This is the common trend adopted not only by engineers but also by team managers, researcher and corporate strategists in a number of industries. When the design is modularized the system or process elements are split up and assigned to a module according to the formal plan. Modularization enables an easier management of complex problems, performing parallel work and accommodating future uncertainty [BC04]. The modularity is extremely important also in mobile robotics design. The implementation example of this thesis is a service robot where the modularity is one of the key parameters of reducing cost and development time. Modularization effect in mobile robotics is opened in detail in doctoral thesis [Ylönen06].

One significant reason for developing new design methodology focused on mobile robots is that the market demand is drastically increased during last years. Many industry reports from Europe [EUROP] and North America evidence the growth of the robotic market [RIA105 & RIA205]. Robotic markets are set to grow quickly and become large economic sectors in their own right as well as providing the means for both manufacturing and service industries to become more effective [EUROP]. Most of the growth in interactive robots at present centers on those that can perform cleaning, security and human-interface tasks, states an industrial report from Japan [Jetro06]. The United Nations Economic Commission (UNEC) and International Federation of Robotics (IFR) estimate that the personal and service robotics market will more than double between 2005 and 2010, reaching $17.1B in 2010 (figure 1). The number of personal and service robots sold is expected to increase ten-fold between 2005 and 2010 according to the UNEC and IFR. Sales for domestic robots (vacuum cleaning, lawn mowing, window cleaning and other types) are expected to reach over million units, while sales for toy and entertainment robots will exceed more than million units.

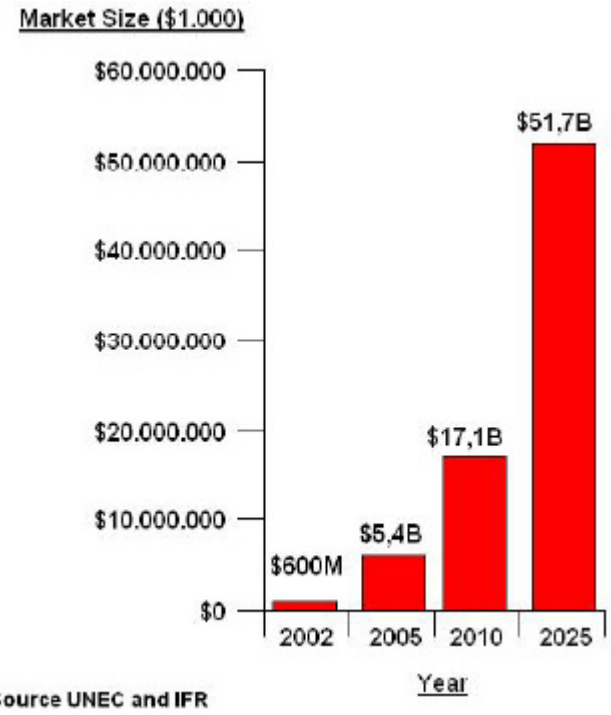

Figure 1 Personal & service market growth [WIRA]

The famous ICT guru Bill Gates estimates in the article in Scientific American [Gates07] that a new technological revolution will take place in the home robotics. However the concern is that the robotic industry and development tools are highly fragmented with only few common standards and platforms. Projects are complex, progress is slow, and practical applications are relatively rare. This refers again to the need for the complex system development methodologies and tools. The service robot development is not any more the niche of the big technological companies but has turned also to the activity of SME type companies, where a standard set of tools is especially valuable.

The practical need for this particular topic selection has also been driven by the author's home department (Department of Mechatronics, Tallinn University of Technology) last year activities. Several mobile robots related projects have been initiated in TUT as well as partner universities in Finland and Sweden. The methodology is partly implemented at demining robot design, which is a running development project in the Department of Mechatronics, Tallinn University of Technology. The methodology has been partly tested and developed based on Workpartner robot – the ongoing research project in Automation and System Technology Laboratory, Helsinki University of Technology.
Workpartner is the next generation interactive service robot for outdoor task. The ultimate goal is a highly adaptive service robot. Mobility is based on the

hybrid system which combines the benefits of both legged and wheeled locomotion to provide at the same time good terrain negotiating capability and good velocity range. The working tool is two-arm manipulator, simulating human behavior as much as possible [Ylönen06]. The Workpartner architecture is highly modular and common interfaces are defined between the modules. Workpartner is one of the case study subject concerned later on this thesis.

Figure 2 Workpartner service robot, TKK

The similar novel Unmanned Ground Vehicle (UGV) type mobile robotic platform development is running in the Department of Mechatronics at Tallinn University of Technology. Similarly to Workpartner described above the UGV is based on a modular design concept.

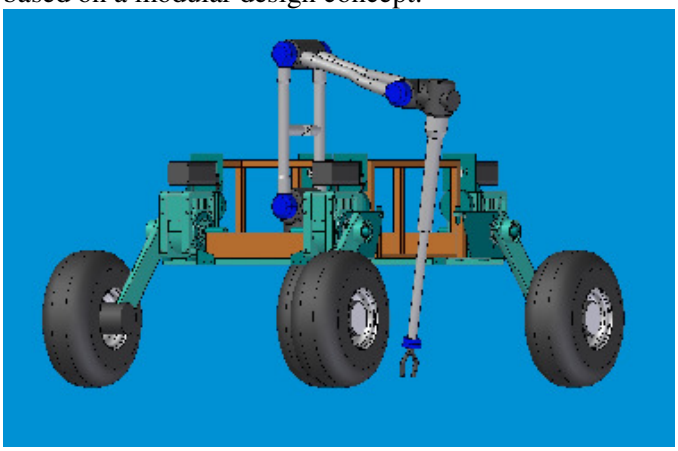

Figure 3 Demining UGV, TUT

Both described mobile robot platforms are highly complex mechatronics systems where the systematic development approach is unavoidable. Methodology developed on the frame of this thesis is applied on both robots. Workpartner is used as an analytical case and UGV as a design case. Generalization is made based on these real life cases and unified template system is proposed for mobile platform design in early stage.

Objectives

The main objectives of the doctoral research and the thesis are as follows:

To analyze existing product design approaches, methods and their suitability for mechatronic system design. To evaluate conventional methods originated from classical domains and new artificial intelligence based techniques exploited in conceptual design. Based on the evaluation, to create unified mechatronics system design process tree with the special focus on early stage, i.e. requirement- and concept engineering.

To create a general framework model for mechatronic design process focused on the early design stage. Utilizing the existing mechatronics and system engineering achievements, to develop an adopted design model corresponding in the best way to mechatronics design needs.

To develop a model based design tool applicable in practical designing and compatible with recent trends and activities in the robotic field in Europe. The design tool has to correspond to the practical needs of research and of industrial institutions as well as to be general enough to represent the developed methodology in generally understandable way.

To develop a problem based application example with guidelines for methodology implementation. The implementation example has to rely on a practical and well understood product from different viewpoints. At least two different approaches have to be used: the existing solution enabling the study of the links between the model and the developed system; the solution under development where the methodology is actually implemented and a real product is created. In addition several real examples have to be presented and a guideline provided to implement the methodology in different application domains.

To analyze the achieved results and define the further development guidelines. Analyzing the achieved results further developments must be pointed out to continue the research and integration with close research works. Further developments guidelines have to be provided and commented.

Scientific novelty

Scientific novelty involves the following:

- analysis of the design methodologies with the focus on early design in mechatronics domain;
- developed conceptual framework model (CFM) for semi-automated early design process;
- developed template based toolkit – Mobile Platform Toolkit (MPT) for practical robot applications;
- provided exampled guideline for further applications.

List of publications

- R. Sell, M. Tamre, Design templates for robot conceptual design, *AIM2007*, ETH Zürich, 2007 (accepted for publication, proceedings IEEE/ASME AIM2007 will be available in the Elsevier databases Engineering Index (EI), Compendex, Inspec, and IEEE Xplore).
- F. Christophe, R. Sell, E. Coatanéa and M. Tamre, System Modeling Combined with Dimensional Analysis for Conceptual Design, *Int. Workshop on Research & Education in Mechatronics*, Tallinn, 2007.
- R. Sell, T. Otto, Advanced E-Curriculum and Mobile Tools for Interdisciplinary Modular Study*, Int. Workshop on Research & Education in Mechatronics*, Tallinn, 2007.
- R. Sell, M. Tamre, An Environment Friendly Autonomous ATV - Practical Mechatronic Project*, Int. Workshop on Research & Education in Mechatronics*, Tallinn, 2007.
- R. Sell, M. Tamre, Hybrid Locomotion Of Autonomous Vehicle - UGV, Int. *Workshop on Research & Education in Mechatronics*, Tallinn, 2007.
- R. Sell, M. Tamre, M. Lehtla, A. Rosin, Conceptual Design Method for General Electric Vehicle, *Proc. of the Estonian Academy of Sciences Engineering*, 2007 (in press).
- R. Sell, Integration of V-model and SysML for advanced mechatronics system design, *Proc. of Int. Workshop on Research & Education in Mechatronics*, Annecy, 2005, pp. 276-280.
- R. Sell, Mechatronics System Design In Conceptual Stage, *4th Int. DAAAM Conference Industrial Engineering - Innovation As Competitive Edge For SME*, Tallinn, 2004, pp. 82-85.
- R. Sell, Mechatronics Design Process and Methologies, *11th Int. Power Electronics And Motion Control Conference EPE-PEMC,* Riga, 2004, pp. 6.20-6.25.
- R. Sell, New Object-Oriented Approach of Modelling Mechatronics System in Conceptual Stage, *5th Int. Workshop on Research and Education in Mechatronics*, Gliwice, 2004, pp. 75-80.

- M. Tamre, A. Kask, R. Sell, P. Leomar, Cognition through Multy-Domain Practical Project, *The Eighth Symphosium on Machine Design*, Oulu, 2003, p. 278.
- R. Sell, P.Leomar, Methodologies on the Mechatronics Domain, *The Eighth Symphosium on Machine Design*, Oulu, 2003, pp. 53-59.
- R. Sell, T. Mart, Component Based Mechatronics Modelling, *Proc. of ICOM2003, Int. Conf. on Mechatronics 2003, Loughborough, GB. (Ed.) R M Parkin, A Al-Habaibeh, M R Jackson.* London: Professional Engineering Publishing, 2003, pp. 111-116.
- M. Tamre, R. Sell, A. Kask, M. Grimheden, The Mechatronics Collaboration Project between KTH and TTU (Sweden-Estonia). *Proc. of Int. Workshop on Research & Education in Mechatronics*, TUD, Lyngby, 2002.
- R. Sell, Real-Time Mechatronics Measurement System In Virtual Environment, *3rd Int. DAAAM Conference*, Tallinn, 2002, pp. 58-61.

Outline of the thesis

This thesis applies to the model based design methodology for mechatronic systems. It contains an introduction, overview of the developed methodology and implementation.

INTRODUCTION
The introduction chapter covers the background of mechatronic and robotic developments, methodologies and design specifics. Trends and surveys of mechatronics and robotics market are presented with recent expectations and estimations from industry leaders. The objectives of the thesis are summarized and scientific novelty described. In addition there is a list of publications, an acknowledgement and abbreviations used in the thesis.

PRODUCT DESIGN
The product design chapter provides an overview of the design process and stages in it. Special attention is paid to the requirement design and conceptual design stages due to the fact that the thesis is orientated towards early design of mechatronic systems. Several most common abstract modeling techniques and methods like bond graphs, Petri nets and Hybrid automata are briefly covered including their usability for early design of a mechatronic system.

STATE OF THE ART
The state of the art chapter covers recent developments of mechatronics design and artificial intelligence utilization for early design. Main widely recognized complex system engineering methods and tools are studied. The main focus is on Mechatronics Design Guideline VDI2206, developed by the VDI, and a very new System Modeling Language (SysML) developed by the SysML Partners,

adopted by the Object Management Group (OMG). In addition artificial intelligence technique based applications for product design are studied and selected researches covered.

DESIGN FRAMEWORK
The design framework chapter is a theoretical work of this thesis. The chapter introduces a new model for the early stage of mechatronic system design. The proposed model adopts the V-model from VDI2206 and SysML based toolkit as a tool for conceptual design of mechatronic system. The Conceptual Framework Model (CFM) is proposed utilizing application specific toolkit and template libraries. As a practical approach the Mobile Platform Toolkit (MPT) is developed. The toolkit embraces template libraries, design models and SysML profile.

IMPLEMENTATION PROCESS
The implementation process chapter is a practical implementation of the developed toolkit, introduced in the previous chapter. The chapter starts with a short guideline for implementing the developed framework follow an application example modeled with the developed toolkit. The model examples include two different applications showing the different ways to apply the developed toolkit. Two different approaches of linking the model with simulation are described.

Recommendations and future development guides are given with the conclusions.

The author has over 15 scientific papers published in the pre-reviewed international conference collections. Six articles are referred on the international databases, including ISI Web of Science/Proceedings.

Acknowledgement

The greatest appreciation goes to my family for their support and understanding throughout the period of thesis compilation and my absence when realizing the fellowship in ETH Zürich in Switzerland and TKK in Finland.
I gratefully acknowledge the assistance of Professor Mart Tamre and my colleagues from the Department of Mechatronics. Special thank to the Professor Arne Halme from TKK for giving the opportunity to finish my thesis in his laboratory.
The work has been carried out with the support of the Estonian Science Foundation Grant No 5908 and the Estonian Ministry of Education & Science Grant No 0142506s03.

Abbreviations

AC	Alternating Current
act	Activity Diagram
ADC	Analog Digital Converter
AI	Artificial Intelligent
ANN	Artificial Neural Network
ANSI	American National Standard Institution
AP233	The draft ISO Standard for exchanging systems engineering data
ASME	American Society Of Mechanical Engineers
bdd	Block Definition Diagram
BG	Bond Graph
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CFM	Conceptual Framework Model
CLAWAR	Climbing and Walking Robots
CPU	Central Processing Unit
DC	Direct Current
DOA	Dead on Arrival
DoDAF	The Department of Defense (DoD) Architecture Framework
EOD	Explosive Ordnance Disposal
EUROP	European Robotic Platform
FEM	Finite Element Method
FoS	Families of Systems
GA	Genetic Algorithms
GP	Genetic Programming
GPS	Global Positioning System
HMI	Human Machine Interface
I2C	Inter-Integrated Circuit
ibd	Internal Block Diagram
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IFR	International Federation of Robotics
INCOSE	International Council on Systems Engineering
ISO	International Standard Organization
LCA	Life Cycle assessment
MatLab	Matrix Laboratory
MC	Micro Controller
MDA	Model Driven Architecture
MPT	Mobile Platform Toolkit
NBC	Nuclear Biological Chemical
OMG	Object Management Group
par	Parametric Diagram
PLM	Product Lifecycle Management

QFD	Quality Function Deployment
req	Requirement Diagram
RS232	Serial Interface
seq	Sequence Diagram
sm	State Machine Diagram
SoC	System on Chip
SoS	Systems of Systems
STEP	Standard for the Exchange of Product Model Data (ISO 10303)
SysML	System Modeling Language
UGV	Unmanned Ground Vehicle
UML	Unified Modeling Language
UNEC	United Nations Economic Commission
VDI	Verein Deutscher Ingenieure
XMI	XML Metadata Interchange
XML	Extensible Markup Language

1 PRODUCT DESIGN

1.1 Design process

Product design process in general can be described as a set of certain phases, needed to be passed despite the diversity of the products even in a single domain. These phases can be defined in various ways and are sometimes related to the problem domain. Nevertheless the phases described in different sources [Ullman02, PBFG07, UE03, Hubka96, Cross89, French99] overlap quite often and the differences are in most cases formal. For example, according to Ullman [Ullman02], the mentioned phases are specification definition, conceptual design, product design and product support phase (figure 4).

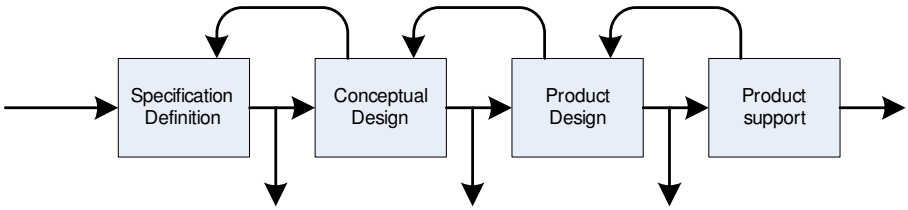

Figure 4 Product design phases, according to Ullman (simplified)

Another widely acknowledged engineering design source - Pahl & Beitz [PB97] defines the design process with four phases as follows:
- product planning and clarifying the task,
- conceptual design,
- embodiment design,
- detail design.

A simplified structure is shown in figure 5. A more detailed graphical schema can be found in [PBFG07].

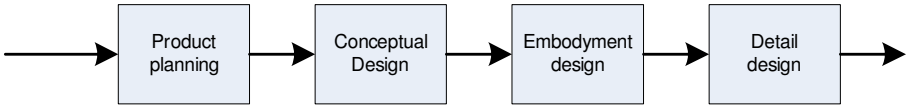

Figure 5 Product design phases, according to Pahl & Beitz (simplified)

The first two phases are almost identical, but the last two are slightly different in these definitions. The input to the embodiment design is similar to the previous

definition of the design concept. The output of this phase is according to Pahl & Beitz, a technical description of the future product, often technical layouts, schemas, drawings, general arrangements or other documents depending on the particular domain and industry. Although the phase listing does not include the product support and utilization phase, today's product design can not leave out these phases. Moreover the disposal and recycling have been turned to be very important issues in modern design and any applied methodology should have to support it.

Discussing the design science, Hubka [Hubka96] has proposed the following main phases for engineering design procedure:

- requirement,
- conceptual design – conceptualizing,
- layout design – embodying,
- detailing.

A simplified schema is shown in figure 6.

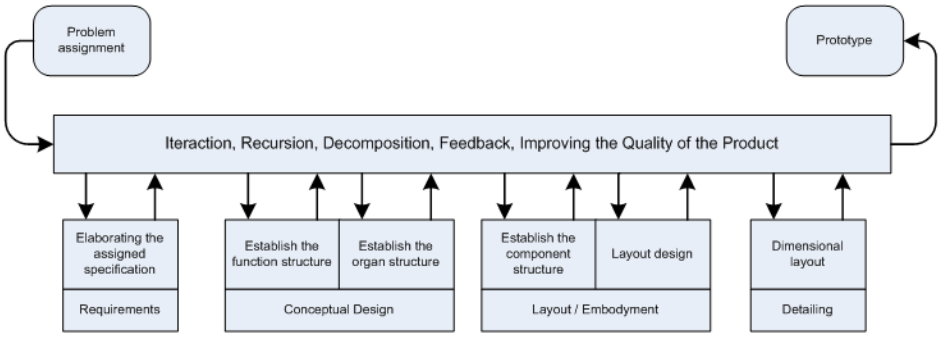

Figure 6 Product design phases, according to Hubka (simplified)

When designing complex systems there are many important relationships which have to be granted through the different design phases as well as engineering domains. Although the previously described design processes may seem trivial they consist many inner activities and the links between the phases are usually much more complex than seen in the simple figures. As also seen in figure 6 there are many keywords, i.e. interaction; recursion, decomposition, feedback, improvement of quality, representing the interactivity between phases. A novel approach covering the system design process including described activities is described in [Pettai05]. In complex and cross-domain products modularization is another key factor which has got much attention. Several examples include also the mobile robotic applications [Ylönen06, STLR07, STL05, HZ07]. The relationship analysis is often carried out by the support of design structure

matrix (DSM) [SA04]. DSM is a matrix representation of a complex system. DSM lists all constituent subsystems or activities and the corresponding information exchange and dependency patterns. The purpose is to provide a quick overview of relationships and dependences between activities. Modularization is often combined with dependence trees to follow all relationships through the design process. It is also important to mention that nowadays the design process needs more advanced tools and techniques to support the product development through the whole design process. It means that we have a product model in all the design phases and this model is improved all the time as the design advances but is also verified and validated against the requirements and reality. What is required and what is feasible is one of the important issues to test before the actual design can start. This brings forth the importance of requirement engineering and later validation against the requirement model. It is especially important in mechatronics because due to the overlapping domains the feasibility is not always easily recognized. With increasing complexity these aspects must be treated with great respect otherwise, the failure of later redesign stage will cost a lot.

As this thesis deals with mechatronics oriented design process focused on the early design the following design process schema (shown in figure 7) is adopted as a generalization and intersection of the previously described definitions.

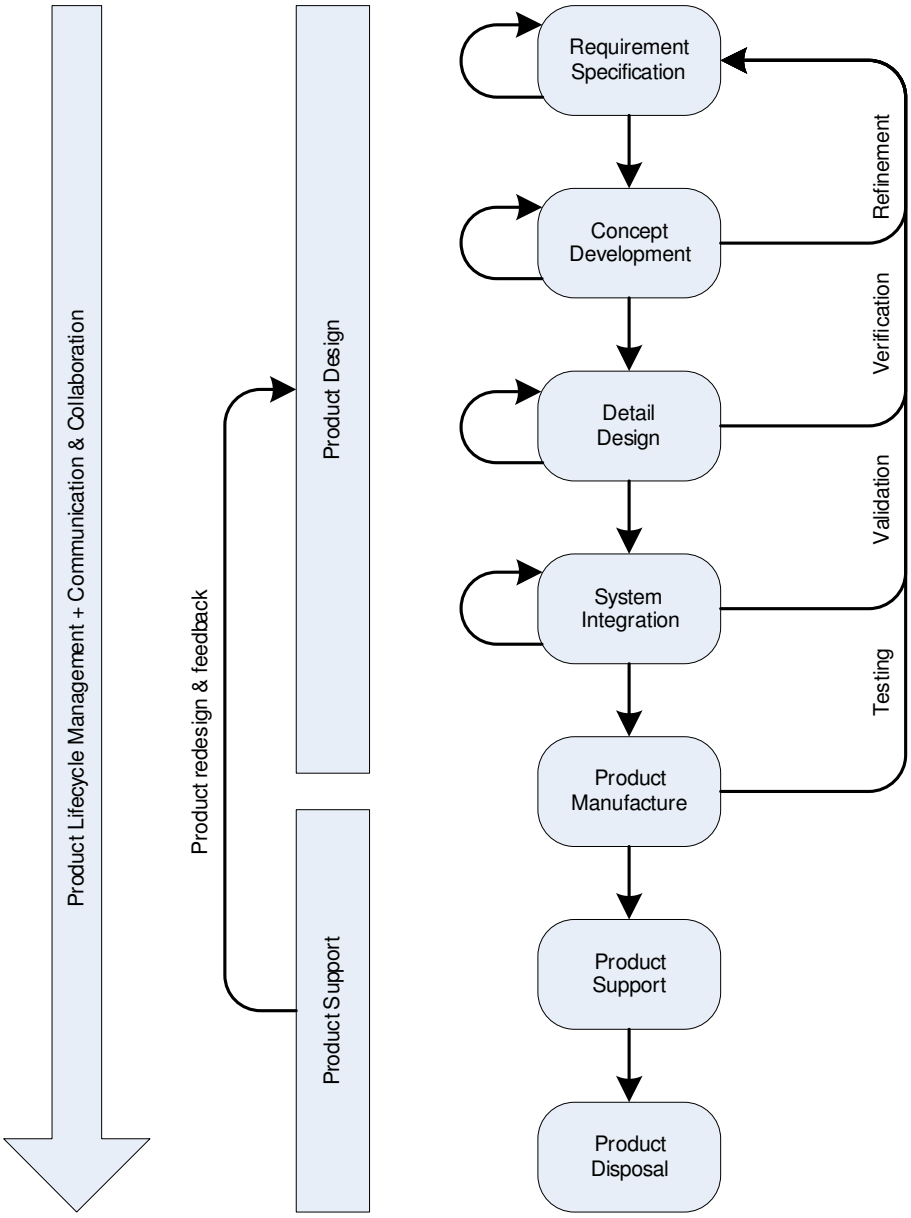

Figure 7 Adapted design process tree

In the requirement specification stage the main goal is to understand the design problem thoroughly respective to the customer requirements. The biggest problem here is that there is very little information about the design problem available in this stage. Initial requirement model is completed usually at the system level. Once this is archived it is necessary to allocate and flow the

requirements down to successively lower level. According to INCOSE System Engineering Handbook [INCOSE04] the requirement modeling process is iterative for each phase, with continuous feedback as the level of the design specification increases. The result of this stage is a design specification and a clearly formulated set of desired measurable behaviors of the future product, which introduces the quality measure into the design process.

The next stage - conceptual design, is particularly important because the decision made here affects most the overall design process. In this stage uncertainty and limited knowledge about the product and big design freedom as well are the major keywords. The danger here is that designers tend to take the first idea and start to refine it towards the desired product. This is a poor methodology because there is great possibility that the idea is not optimal or not the best one at all. Therefore it is important to generate at least more than one or two design candidates. Then it is much more plausible that designer may find a good solution [Ullman02]. One goal of this stage is to choose best alternatives with the least expenditure of time and other resources. Techniques generating concepts and making decisions are used interactively as knowledge is increasing with new ideas [Ullman02]. The iteration is less expensive during this stage than in the following stages. The result is gained with the careful selection process from different competing design candidates. The initial analysis and simulations of competing solution candidates are already necessary even if there is a great lack of rigid mathematical parameters and lots of loosely defined constraints. The result of this stage is usually system architecture, functional and behavioral specification of the future product.

The detail design, which can also be referred to the product design phase, is the most time and labor consuming part of the design process. Here the concept generated and evaluated on the conceptual design stage will be refined and all sub-system and components will be developed. Detail design is quite well defined. This includes a technical drawing generation, detail product sub-system model generation, etc. For example in mechanical engineering part and assembly modeling as well as stress analyses with FEM will be carried out. The stage defined here (figure 7) overlaps Pahl & Beitz's embodiment design and the detail design. Although the input of this stage is evaluated and the concept is selected the detail design may embody a parallel layout design. The purpose of simultaneous layout design is to obtain more information about advantages and disadvantages of different variants. According to Pahl & Beitz detail design stage description, the optimization is a crucial activity compromising optimization of principles, optimization of the layout, forms and materials; and optimization of the production [PBFG07].

The integration is a key element on the successful mechatronics design. The integration stage embraced into adapted design tree, is derived from the nature of the mechatronics and system engineering specifics. The integration is important activity enough to deal with it as a separate stage in whole development process. In the classical domain engineering, the integration before the final product is also presented but it is not as complex as in nowadays mechatronics products. The reason of the great complexity is related to integration issues of sub-systems originated from dissimilar domains where even the design methods and tools can have totally different concepts. The integration stage includes the testing and simulating of the final product model before the prototype or the product will be manufactured.

The production stage is not only the actual manufacturing but also a planning of manufacture, building and assembling, quality check and testing.

The product service and disposal stage covers all activities related with the product design support: support for manufacturing/assembling, support for vendors, support for customer and disposal activities.

In parallel of modeling several tests and verifications need to be accomplished during the design process. The simulation is widely exploited method to test the system or subsystem in specific computer environment. Several software packages are used for mechanical, electrical, software, etc. simulations. There exist special mechatronics-oriented simulation packages like AMESim, Dymola, Adams and in addition some well known general simulation packages e.g. Simulink, suits for mechatronics simulations as well. The early stage simulation on the other hand is not often used and there are no specific packages available. The reason is the lack of system parameters and high abstraction level of the given system. Despite of that the need for early design simulation exists due the fact that early design decisions affect the final product in very wide scale.

Design simulation can be extremely valuable but needs also a good simulation model which is not trivial to develop. The executable model has to reflect the conceptual model as well as reality at the same time [Pelz03]. The verification procedure is used to investigate the executable and conceptual model compatibility and validation procedure investigates whether the executable model reflects the real world adequately. In other words as stated by Peltz [Peltz03]: “Verification ensures the system is modeled right, whereas validation is all about modeling the right system”. The corresponding graph is shown in figure 8.

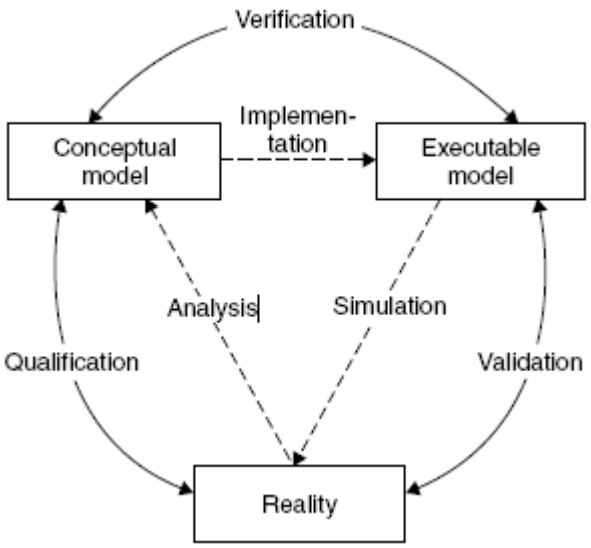

Figure 8 Context of simulation [Pelz03]

1.2 Early Design

Early design is a common term for all activities before detail product design. This includes requirement specification, conceptual design and other concurrent activities. Early design has got the more close attention in recent research due to the fact that the proper requirement engineering and optimized concept selection determine a significant part of the future product cost. Here in early design a big potential for reducing production cost and time to market is available. Some very recent researches have been focused right on early design for the reason mentioned before. Gausemeier's team work [Frank07] (who is one of the author of VDI2206 – Mechatronics Design Methodology) from Heinz Nixdorf Institute, University of Paderborn is one of the example. The research stands for new methods for the conceptual design of intelligent mechatronic systems. Another research actively carried out in Helsinki University of Technology [Coatanea05] has the special focus on the early design. The self-optimization behavior of mechatronics system is one of the growing research interests and this involves directly the early design. Parallel and interactive design in early stage is unavoidable already today. Automation and applying new algorithms for concept creation and concept simulation is one of the key factors of successful early design. This is the main issue in the theoretical part of the current thesis.

The requirement engineering is one of the vital stages of the whole design process. Sometimes the engineers tend to underestimate the requirement specification. It is not enough any more to take just customers' plain

specification and start to develop the concepts from that. It is quite common that a customer has but a cloudy idea or concept what the final product should look like. This is the case even with functionality of the ordered product. Some unimportant functions may cause a decline of the budget or some may compromise the whole system. If a design engineer takes this kind of requirement as fixed without estimating and refining requirements together with the customer it may result in the failure of the whole product market access or the functionality. Therefore the requirement engineering must be carried out with great attention and concurrent refinement during concept development and even detail design must be established.
The well known and accepted technique for bringing together customer requirements and engineering specifications is Quality Function Deployment (QFD) [QFDHb]. Due to the universal approach the concept is deployed besides engineering field in many other areas like marketing-analysis, brand development, software development, etc. It is very often used at early design stage to ensure the customer needs in final product. Although there are proven benefits using QFD it does not come automatically. It is not necessarily easy to start with but when the first effort is performed the later benefit is remarkable. Kenneth Crow – president of DRM Associates proposes a list of recommendations to facilitate initial use of QFD [Crow].
It is also implied in the ISO 9000:2000 standard which focuses on the customer satisfaction.
The requirement engineering and connections with design solutions are discussed in chapter 3.

1.3 Conceptual design

Decisions made during the conceptual design stage have significant influence on the final product. It has been estimated that more than 75% of final product cost is settled in conceptual stage [Lotter86, HW98]. Different methodologies for detail product design have been proposed [VDI04] and applied, but even the highest standard of detailed design cannot compensate for poor design concept formulated at the conceptual design stage [PBFG07]. For this reason the concept evaluation and verification of the conceptual design development becomes more important. The reason is also the complexity of mechatronic systems and rising of new techniques together with increase of computational power as well as time and cost saving demands in today's rapid world. Many efforts are still needed to develop industry proved semi-automated tool for the conceptual design phase. To archive this goal we need to define a problem in unified form at the phase where the big uncertainty is a common feature. This is not a trivial task considering that it must be easily understandable for product designers and other technical staff as well as for customers.

In the conceptual design process, the designer takes the problem statement and generates a broad solution for it in the form of schemes, block diagrams, etc. [PB97]. At this stage engineering science, practical knowledge, commercial aspects, production methods and other relevant aspects need to be brought together, and the most important decisions are taken. Techniques and methods used for conceptual design are not for replacing the designer or generating complete solutions. They are intended to improve the quality and speed of the design process and to help designers by [French99]:

- increasing insight into the problems, and speed of acquiring the insight,
- reducing the size of mental step required in the design process,
- diversifying the approach to the problems,
- generating design philosophies for the particular problem in question,
- documenting the design process progress,
- improving communication between designers and/or other relevant people.

There are several widely used techniques suitable for conceptual design process. These are covered in detail in the Conceptual Design for Engineering [French99]. A selection of techniques is as follows:

- construction of table of options, functional analysis,
- mathematical models,
- the search for alternatives,
- logical chains,
- past practice,
- problem solving and analytical techniques.

As conceptual design stage is very creative in its nature these techniques are important to know and use. Most known techniques are:

- brainstorming - generating options,
- critical path analysis - planning and scheduling complex tasks,
- decision trees - powerful quantitative analysis of decision impact,
- force field analysis.

Detailed information about problem solving and creative techniques can be found from reference [Higgins93] and [Dombroski00].

Nowadays complex systems and rapid time-to-market demand on mechatronics product sets new demands for conceptual design process. Traditional methods described above still work fine but are inefficient as they do not take into account the interdisciplinary aspects from the very beginning. In mechatronics design more abstract methods are needed for modeling and evaluating the concepts at an early stage. The early integration has to be started on the very beginning when starting to define the customer requirements. Requirement definition has to be universal and at the same time enable verifying later design solutions. For example, the software requirements might have demands and definition methods quite different from mechanical ones. Nevertheless for a

mechatronics product which is one integrated unit, the utilized method and evaluation values have to be comparable. New approaches for integrated product conceptual design are constantly searched around the world. Some of the methods are covered in paper [Sell04a]. One ongoing research on this issue is introduced by Coatanéa [Coatanea05] where base theories are a topological model of design theory presented by Braha and Reich [BR03] and the C-K theory developed by Hatchuel [HW02]. The practical implementation and advancement of these theories results in a design synthesizes process at the conceptual stage supported by the dimensional analysis [CYHHSM07] and semantic atlas [CYHSML07].

1.4 Template libraries

An effective and optimum design exploits often pre-defined design patterns and templates. Although most mechatronic products today share many common concepts and components the design is in most cases independent and unique. Component libraries exist and are exploited widely but not before the product detail design stage. The conceptual stage for a mechatronic product still lacks common design methods as well as general design patterns or templates.

Templates offer several significant advantages. Templates are general and reusable. Thus, they can be used easily for testing the system variations in very early stages. Moreover, it is characteristic for robotics applications that similarly to variety of routes for reaching a target location there are several alternative solutions especially in early stages of design. Templates exploit general algorithms and descriptions created and tested by experts or verified by experience. Thus, many errors can be avoided compared with the untested model algorithms. The system design can be started in a shorter time using templates and an engineer could exploit many design options in parallel.

Two basic approaches for component based design are commonly utilized [BBC06]:
The designer compiles a system mechanically from “black boxes”, where the black box is considered: a component that can be used without knowledge of its inner workings; the user supplies the input and the output is more or less guaranteed to be correct without a need to understand the algorithmic details or component physics.
The second approach, where the designer wants to be able to fine tune the system: adoption of a component for a specific application and environment in order to obtain better performance.
Both approaches are widely used particularly in software development. Designing mechanical subsystems engineers have standard part libraries available. In electronics design, an engineer can pick a motor driver, an interface or other component from the library without knowing the inner structure and

working principles of that particular component though the system needs fine tuning. The conceptual stage of a mechatronic system development integrates these approaches utilizing the advantages of both.
A template is a description of an algorithm, abstracting away from implementation details, thus a template is a description of a general algorithm and in our cases also an executable object. Nevertheless, although templates are general descriptions of the key algorithms, they may offer whatever degree of customization the user may desire. They can be configured for the specific structure of a problem (e.g. drive mechanism) or for the specific computing language [BBC06]. For example the simulation template is initially implemented in Simulink but can be also defined in Modelica language [Modelica] to execute the simulation in Dymola environment.
Pre-defined templates are very common in software development field. For example Standard Template Library and Active Template Library for C++. The template library provides a ready-made set of common classes, such as containers and associative arrays, that can be used with any built-in type and with any user-defined type that supports some elementary operations (such as copying and assignment). STL algorithms are independent of containers [STL]. In UML models, templates are model elements with unbound formal parameters that can be used to define families of classifiers, packages, and operations.

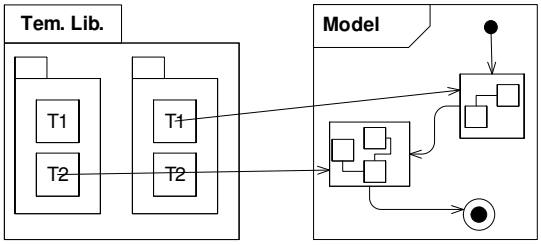

Figure 9 Element bindings from template library

1.5 Domain-independent modeling methods

A number of established methodologies have been adopted for product design in general. Methodologies and techniques are widely used by industry and research institutions, however different methodologies are applied depending on the problem, organization and others factors. Some methods are more popular and are further advanced, other are used rarely or only in a very specific problem domain. Some general engineering design approaches can be pointed out according to several design science sources [Hubka96, Pelz03]:

- Concurrent engineering workflow
- Bottom-up design
- Top-down design

- Trial-error approach
- Intuitive design process
- Automated design process
- Design in context
- Modular design.
- Design for Six Sigma
- Digital simulation engineering.

According to Hubka the intuitive design is the most traditional and widely used in different domains but although the resulting technical solution is usable it is not optimal. In addition it needs a good experience and a broad view on the given problem. The most effective design process type according to Hubka design process typology [Hubka96] is a partly or totally automated design process where the data is an input and the solution is an output which will be further evaluated by the engineers. When supported by correct tools and methodology the result will be optimized and of good quality within a short time.
In mechatronics design domain-independent techniques are required due to its cross-domain nature. It is especially important in conceptual design where the working principles are developed without linking it to the actual solution. The simple block diagram is definitely a most widely used domain-independent technique. Hereafter some more specific cross-domain approaches are described in more detail.

1.5.1 Bond graphs

The bond graph theory is a widely recognized domain-independent technique. Bond graphs were introduced by Henry Paynter, professor at MIT, who described the junctions in 1959 and the whole framework in 1961 [Paynter69]. A bond graph is a graphical description of a physical dynamic system. The concept is based on the energy flow. Bond graphs have a number of advantages over conventional block diagram or code-based modeling technique [Gawthrop99]:

- they provide a visual representation of the system,
- since they work on the principle of energy conservation, it is difficult to accidentally introduce extra energy into a system,
- the bonds are symbols which contain meaning,
- they separate the structure from causality,
- since each bond represents a bi-directional flow, systems which produce a 'back force' (e.g. a motor back emf) on the input are easily modeled without introducing extra feedback loops,
- hierarchy can be used to manage large system models.

Bond graphs are based on the principle of continuity of power. If the dynamics of the physical system to be modeled operate on widely varying time scales, fast continuous-time behaviors can be modeled as instantaneous phenomena by using a hybrid bond graph.

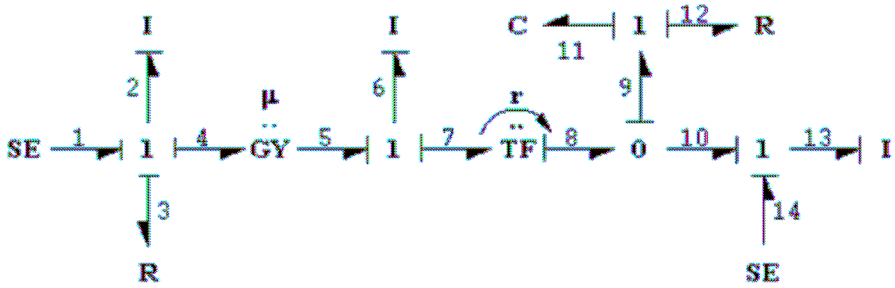

Figure 10 Example of the bond graphs [BG01]

The example above represents the simple model of electrical motor driven winch. There are following different types of elements:

- three basic one-port passive elements,
- two basic active elements,
- two basic two-port elements,
- two basic junctions.

The basic variables are effort (e), flow (f), time integral of effort (P) and the time integral of flow (Q). Basic one-port passive elements are R-element, C-element and I-element. The R-element is resistor type element where flow and effort are related by static function. Usually they dissipate energy. The C-element is similar to electrical capacitor or mechanical spring. It stores and gives up energy without loss. The I-element is respectively similar to electrical induction or mechanical inertia. The element stores the energy if the momentum, P, is related by a static constitutive law to the flow, f.

Basic active elements are sources of effort and flow denoted SE and SF respectively. For example the electrical source is defined as an effort source (SE). The active elements are Transformer and Gyrator. Transformer is representing the transformation device which converts and transmits the power across the net. Ideal examples of transformer are electrical transformer, mass less ideal lever and electrical motor. A transformer relates flow-to-flow and effort-to-effort. Conversely, a gyrator establishes relationship between flow to effort and effort to flow, again keeping the power on the ports same. The simplest gyrator is a mechanical gyroscope [Broenik99]. The above bond graph example is explained in figure 11.

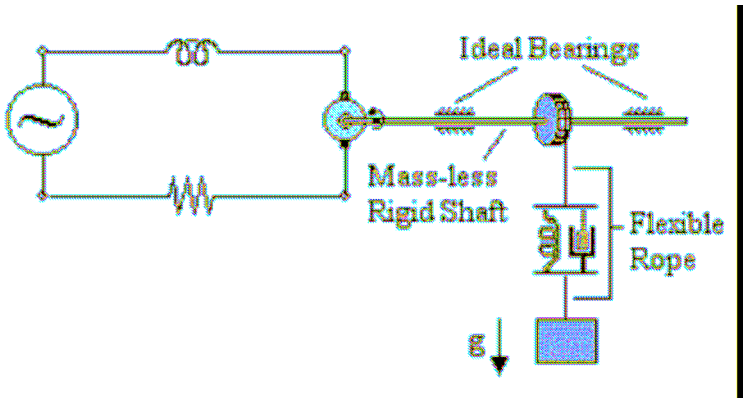

Figure 11 Corresponding system to bond graph example [BG01]

However the bond graph concept is not very much exploited by practicing engineers. The reason is that developing the complex system the bond graph diagram turns to very complicated and voluminous which makes it very difficult to read. Reading the bond graph diagrams needs knowledge of bond graph concept and elements. Although the concept is not very sophisticated and the number of standard elements is rather small the representation is very symbol oriented. Without knowing the meaning of a symbol it is almost impossible to figure out what is presented. Other concepts are usually more user friendly and are easier to understand the context even without knowing the meaning of all symbols. Despite of that, in mechatronics field the bond graph concept is much more used than in other domains. In many cases it is not directly used but the concept itself is utilized as a base approach for modeling mechatronics systems.

1.5.2 Hybrid dynamic system

Another technique dealing with complex systems is a hybrid dynamic system approach. Hybrid dynamic systems (HDS) contain subsystems with continuous dynamics and subsystems with discrete dynamics that interact with each other [AN98]. A hybrid system is a system where both the discrete states and continuous behavior are represented in the same model – continuous model with continuous flow where discrete events can occur. Modeling such kind of system is a complicated mathematical problem since continuous dynamics and discrete event dynamics have entirely different mathematics. The detail hybrid automata logic is explained in the widely recognized Davoren IEEE article [Davoren00] in 2000.
The standard hybrid automata graphical description is shown in figure 12. The hybrid automata are described as an infinite-state transition system. Discrete states are denoted with p and q in figure 12. The transition includes the guard and reset. Transition is an event occurring when the guard constraint is set true. In this example the guard equations are y<y’ and x=x’. When the transition is

completed the reset function is executed. Here the reset function sets the x and y to zero in a and c transitions respectively.

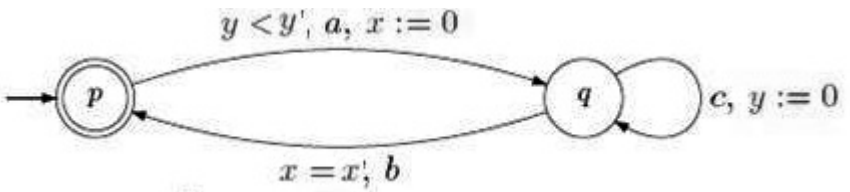

Figure 12 Hybrid automata graphical representation

Hybrid automata models are used to model embedded real-time controller behavior. To fulfill hard real-time requirements is becoming an increasingly important task in different application areas, e.g. in robotics, control technology, medicine, etc.
Practical software tools exist for hybrid system modeling. One of the tools is Uppaal – developed jointly by Swedish and Danish universities, where addition to discrete event modeling the verification, validation and simulations can be executed on real-time model. Another software package is HyVisual - Hybrid System Visual Modeler, developed by UC Berkeley. This modeling environment has the continuous and discrete event modeling capabilities in same model. Both softwares have a non commercial version for educational use free of charge. In addition there are numerous other software tools more or less oriented to hybrid system modeling, simulation, validation and verification. A list of packages can be found on Virtual Action Group on Hybrid Dynamic Systems website [HDS].

1.5.3 Petri nets

One possible way to model a hybrid dynamic system is the Petri net concept. Petri nets were invented in 1962 by Carl Adam Petri in his PhD thesis [Petri62]. The Petri nets or predicate/transition (Pr/T) nets are a graphical representation of discrete system. The concept is rather simple where the net has place nodes, transition nodes, and directed arcs connecting places with transitions. The execution of Petri nets is nondeterministic which means that multiple transitions can be enabled at the same time, any one of which can fire and none are required to fire at all. Since firing is nondeterministic, Petri nets are well suited for modeling the concurrent behavior of distributed systems.

A Petri net is a 5-tuple (S, T, F, M_0, W), where:

- *S*, is a set of *places*.
- *T*, is a set of *transitions*.
- *F*, is a set of arcs known as a *flow relation*. The set *F* is subject to the constraint that no arc may connect two places or two transitions, or more formally: $F \subseteq (S \times T) \cup (T \times S)$.

- $M_0 : S \to \mathbb{N}$ is an *initial marking*, where for each place $s \in S$, there are $n_s \in \mathbb{N}$tokens.
- $W : F \to \mathbb{N}^+$ is a set of *arc weights*, which assigns to each arc $f \in F$ some $n \in \mathbb{N}^+$denoting how many tokens are consumed from a place by a transition, or alternatively, how many tokens are produced by a transition and put into each place [Desel01].

An example of a Pr/T net is shown in figure 13.

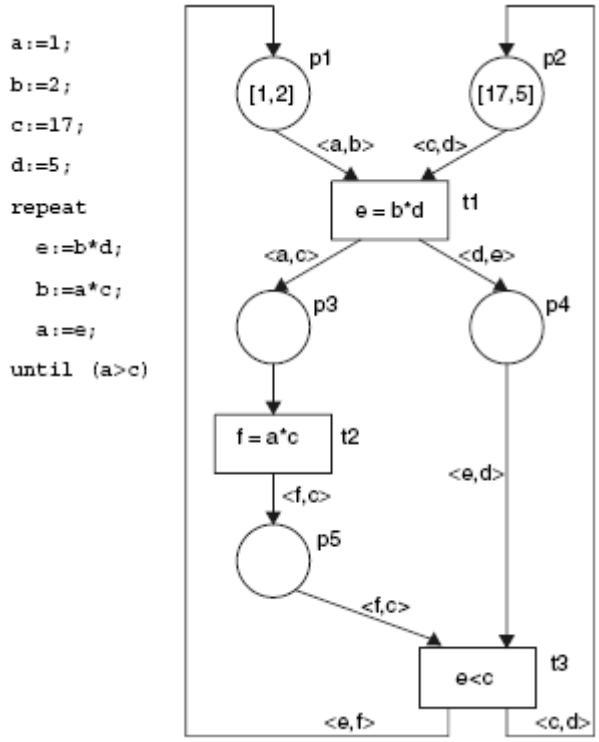

Figure 13 The example of Pr/T net modeling of digital behavior [BK93]

Petri nets are often used for modeling of software and digital electronics and simulating mixed systems [Peltz03]. It is also used in modeling and simulation of manufacturing systems, hardware design and process modeling. Some authors have applied the technique to the mechanical and other physical domains [BSHW97, KTT97]. The main strength of this technique is effectiveness of dealing the parallel processes. The drawback of Petri nets on the other hand is their poor structuring facilities. They are thus difficult to use for large scale systems [EK96].

Some closely related techniques worth to mention are Mixed Logical and Dynamic Systems framework proposed by Bemporad and Morari

[Bemporad99], Hybrid State System Modelling [Dogruel97], Numerical Integration method for Hybrid Dynamic Systems [Schlegl97] and others.

1.6 Normatives on mechatronics design

There are some industry normatives and guidelines related to mechatronic system design process. The latest and probably the widest coverage guideline has been developed by German Association of Engineers (VDI) in 2002 and revised in May 2004. The guideline VDI2206 covers the mechatronic design process proposing numbers of tools for design support. Concerning ISO standards, there are no direct documents related to the mechatronics design process. Closer normatives that affect mechatronic system design and conceptual design are:

- ISO 15288(2002) Systems engineering – System life cycle processes
- ISO 11442-5 (1999) Technical product documentation Handling of computer-based technical information. Part 5: Documentation in the conceptual design stage of the development phase
- ISO 10303/STEP Industrial automation systems and integration – Product data representation and exchange / Standard for the Exchange of Product Model Data

VDI guidelines affecting mechatronics system design are the following:

- VDI 2206 (2002, revised 2004). Design methodology for mechatronical systems (Entwicklungsmethodik für mechatronische Systeme. Richtlinienentwurf)
- VDI 2422 (1994). Systematical development of devices controlled by microelectronics (Entwicklungsmethodik für Geräte mit Steuerung durch Mikroelektronik)
- VDI 2221 (1993). Systematic approach to the development and design of technical systems and products (Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte)

International projects and industry standards affecting the system engineering and mechatronics in general:

- IEC 61499 Function Block Standard [IEC05]
- DoDAF Department of Defense Architecture Framework
- AUTOSTAR Automotive Open System Architecture
- EAST-EEA Electronics Architecture and Software Technologies – Embedded Electronic Architecture
- RIF Requirement Interchange Format
- SDL System Description Language

Besides the documents mentioned there exist numbers of additional standards and guidelines proposed by industry or associations. Applying these documents to the design process depends on the developers need and companies or customer policies.

1.7 Conclusions

1. In this chapter widely recognized product design process trees are described and analyzed. Similarities and differences of most well known process descriptions are pointed out. The mechatronic specific needs and problems are shown and trends mentioned. Based on the widely recognized authors an adapted design tree has been compiled. The adopted solution is primarily originated from the specific needs of mechatronics and system engineering specifics. Main difference compared with the classical engineering process is a strong focus on the integration and early design stage. The major reason is the later integration issue where all developed sub-systems must work together as one synergetic system.
2. According to integration issue in mechatronics design, but also the growing importance of conceptual design support several well known domain-independent techniques like bond graphs, Petri nets and hybrid automata are discussed. Advantages and disadvantages of these techniques as well as main application fields are pointed out.
3. The result of this chapter is an overview of the existing design methods and domain-independent techniques. It is demonstrated that the optimal results in complex systems are very difficult to archive with traditional methods in a mechatronics field. It is not possible to solve the whole range of early design needs and therefore we need to combine different techniques with suitable methodologies. This issue is discussed in more detail in next chapter where state-of-art solutions and research trends are studied.

2 STATE OF THE ART

2.1 Complex system modeling methods

The mechatronic system development process varies from classical engineering process in many ways as mentioned several times before. It is still quite new paradigm and there is no proved methodology in this area. Another reason is that mechatronics itself is developing at a great pace and the design methodologies must follow this advancement. However, a number of acknowledged frameworks exist. Some of them are more practice oriented [VDI04, DODAF] and implemented in different new technology industries, others are more theoretical [SEO03, GFK07] and under the research in universities and research institutions.

A highly industry oriented and huge work has been done by VDI association. This work stands for a new mechatronics system development guideline known as VDI2206. This guideline presents the most recent points of view and has been worked out by 40 well-known specialists in the field of mechatronics.
In the VDI2206 guidelines the design methodology main base elements are [VDI04]:

- the general cycle of problem solving on the micro-level,
- the V-shape model on the macro-level,
- pre-defined process modules for repeating an operation step during the design of mechatronics systems.

VDI2206 complements the previous related guidelines like VDI2221 (Systematic approach to the development and design of technical systems and products) [VDI2221] and VDI2422 (Systematical development of devices controlled by microelectronics) [VDI2422].
The main aim of this guideline is to support a cross-domain design of a mechatronic system in a systematic way. Procedures, tools and methods for the design process are described on the frame of system design approach. The guideline is supposed to structure the variety of findings which have been developed through research and industrial applications in the last years and to introduce them to practice in a concise and understandable way [Gausem03].

The scheme for describing mechatronic design process is the V-shape lifecycle model shown in figure 14. The model is based on the traditional waterfall process model but has a set of useful properties in description in the sense that relations between different design stages can be easily illustrated. Although this thesis focuses mostly on the first and second phase of the V-shape model (specification and conceptual phase) as early design stages, the product will be developed as many circles around the V-shape as needed. Every so-called

product does not have to be a real product, but can be a specification, model, prototype and of course the product itself as well. Needed interactions depend very much on the content and complexity of particular product design. This guideline is mostly adopted in Germany and other European countries.

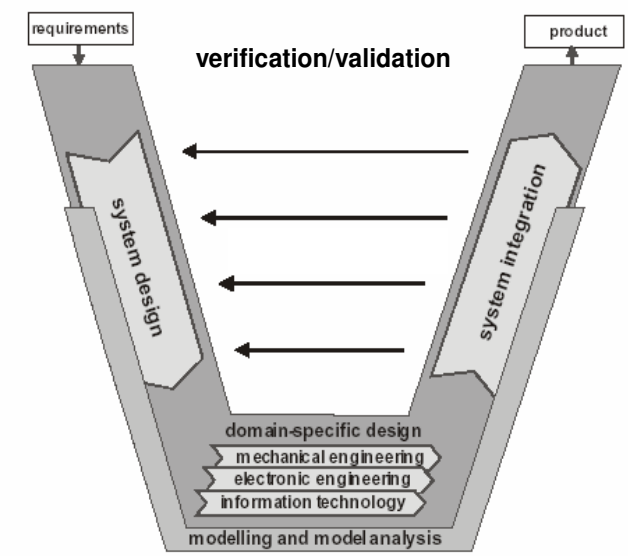

Figure 14 Adopted V-shape model from VDI2206 [VDI04]

The conceptual design part is denoted here as a *System Design*. As explained before, the conceptual design can also be the whole round of V-model, where the product is a selected concept. It is even possible to have several circles of the V-model what will lead into selected and optimized solution candidate. The interaction number depends very much the particular case.

The *System Design* comprises the following activities according to [VDI04, PB97, KBS97]:

- Abstraction to identifying the main problem,
- Setting up the function structure overall function – subfunction,
- Searching for operation principles/solution elements for the subfunction,
- Concretizing to form solution variants in principle
- Assessing and selecting
- Establishing the domain-embracing solution concept.

However, before the design process can begin, the need for the product must be clearly analyzed. This is usually done in the requirement engineering stage which is not covered by this guideline in depth. In most cases there is a direct market demand. Although the most new products are market-driven, the design

process can also start without the market demand and in this case the design process is product or producer driven.

While the VDI2206 is relatively known in Europe, the DoDAF methodology for mechatronic product design, or more general the system engineering, is the recognized methodology in U.S. The Department of Defense Architecture Framework (DoDAF) is a framework for development of a systems architecture or enterprise architecture (EA). All major U.S. Government Department of Defense (DoD) weapons and information technology system procurements are required to develop an EA and document that architecture using the set of views prescribed in the DoDAF [DODAF]. The intent of the DoDAF is to ensure that the system architecture description can be compared and related to another across the organizations. Although the framework is clearly aimed at military systems, it has broad applicability to define the architecture of complex systems so that it can be evaluated and understood alongside the other architecture description developed according to the same guidance. Decision makers can then use the DoDAF compliant reports to compare the alternative solutions and evaluate them amongst the existing system [WIDNEY06].

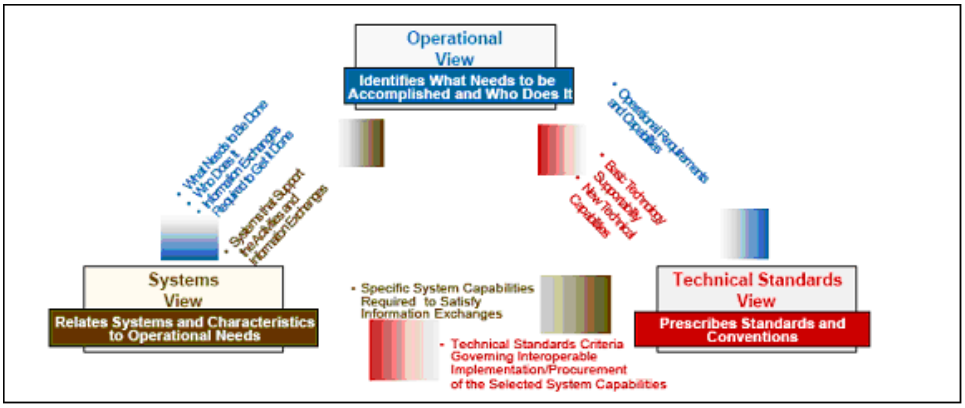

Figure 15 DoDAF views [DODAF]

According to the definition of DoD Architecture Framework Working Group [DODAF] in the framework, there are three major perspectives (i.e., views) that logically combine to describe an architecture description. These are the Operational View (OV), Systems View (SV), and Technical Standards View (TV). Each of the three views depicts certain architecture attributes. Some attributes bridge two views and provide integrity, coherence, and consistency to architecture descriptions [DODAF].

- Operational View
  The OV is a description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions. The OV contains graphical and textual products that comprise an

identification of the operational nodes and elements, assigned tasks and activities, and information flows required between nodes. It defines the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges.

- Systems View
  The SV is a set of graphical and textual products that describes systems and interconnections providing for, or supporting, DoD functions. The SV associates systems resources to the OV. These systems resources support the operational activities and facilitate the exchange of information among operational nodes.
- Technical Standards View
  The TV is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements. Its purpose is to ensure that a system satisfies a specified set of operational requirements. The TV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established, and product lines are developed. The TV includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria organized into profile(s) that govern systems and system elements for a given architecture.

2.2 System engineering modeling concepts

According to the International Council on Systems Engineering [INCOSE], the Systems Engineering "is an interdisciplinary approach and means to enable the realization of successful systems. The whole design process focuses on defining customer needs and required functionality early in the development cycle, documenting requirements then proceeding with design synthesis and system validation while considering the complete problem of operations, performance, test, manufacturing, cost & schedule, training & support and disposal". This definition points out the importance of early design and integrated activity very clearly. It also sets high demands for modeling concepts and tools. Hereafter the most approved and state-of-art system modeling technique based on model driven architecture is described.

In the software design world UML is de facto standard for object-oriented software design. Starting with UML 1.1 and UML 1.5 the most recent official version is now UML 2.0. The primary driving force behind UML 2.0 is model-driven development, an approach to develop software that shifts the focus of development from code to models, and to automatically maintaining the relationship between the two. The essence of software modeling (as in all modeling) is abstraction: the removal of fickle and distracting detail of

implementation technologies as well as the use of concepts that allow more direct expression of phenomena in the problem domain [UML].

In today's world the software engineering is a leader deploying the concept of model driven architecture. Object Management Group (OMG) – the coordinator of software standard development has approved the Model Driven Architecture (MDA) specification in 2001. MDA supports model-driven engineering of software systems, also the concept is applicable to system engineering as well. The MDA model is related to multiple standards, including the Unified Modeling Language (UML), the Meta-Object Facility (MOF), the XML Metadata interchange (XMI), Enterprise Distributed Object Computing (EDOC), the Software Process Engineering Metamodel (SPEM), and the Common Warehouse Metamodel (CWM) [MDA]. However there are some concerns with MDA approach. So far there is no mainstream industry acceptance; the Gartner Group still identifying MDA as an "on the rise" technology in its 2006 "Hype Cycle"[GARTNER], and Forrester Research declaring MDA to be "dead on arrival" in 2006 [Zeite06]. There are of course always opposite opinions [Frankel06].

There have been several attempts to apply UML for non-software design. Serious improvement has been reached in recent years. An important outcome is OMG SysML specification finalized in this year (2007) which is initially derived from UML RFP: UML for System Engineers Request for Proposal [SERFP] in 2003. However there are several state-of-art works carried out by research groups based on the UML profile mechanism:

UML Profile for Schedulability, Performance, and Time Specification [UMLSPT];
UML 2.0 Profile for Embedded System Design [KRH05];
UML Testing Profile [UMLTP];
UML Profile for SoC (Systems on Chip) [RHSZN05];
UML 2 to Solve Systems Engineering Problems [Gurd03];
UML for Hybrid Systems [BBHP06];
SYSMOD – Systems-Engineering-Profil [Weilkiens06].

Recently the major players in government, industry and ICT have collaborated to extend UML to cover the domain of Systems Engineering. This new standard – SysML – is adopted by the Object Management Group in the autumn of 2005 [SysML10]. During 2007 the finalized version of SysML 1.0 is expected. So far the version 1.0 draft from May 2006 is the adopted specification.
SysML reuses a subset of UML 2.0 diagrams and supplements them with new diagrams and modeling constructs appropriate for systems modeling. SysML is designed to complement UML 2.0, so systems engineers who are specifying a

system with SysML can collaborate efficiently with software engineers who are defining a system with UML 2.0 [SysML09].

The following figure describes SysML modifications and relations with UML 2.0.

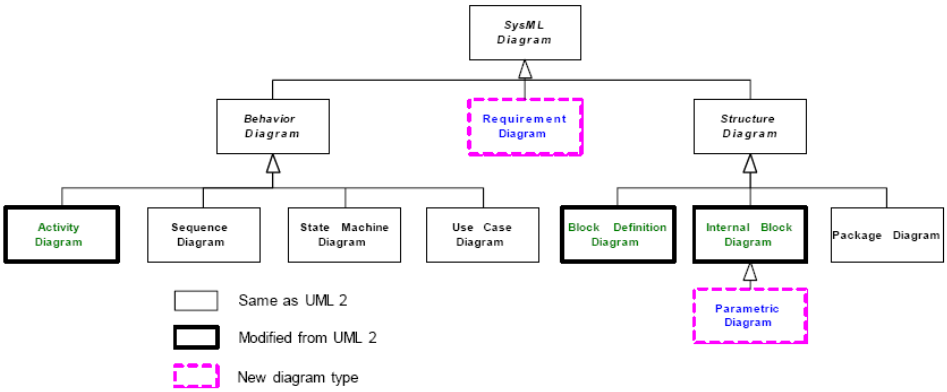

Figure 16 SysML diagram taxonomy [SysML10]

SysML language is a modeling language where the system model and sub-models are described with a set of diagrams. Several diagrams are adopted from UML and some like *Requirement* and *Parametric* diagrams are developed for this purpose. The concrete syntax and notations are described in SysML specification which is OMG Adopted Specification ptc/06-05-04 [SysML10].
SysML diagrams can be generally divided into the four sections:

- Requirement specification,
- Structure definition,
- Behavior definition,
- Parametric definition.

When modeling a system, an important primary task is to decide what belongs to the system and what does not. The *Context* diagram is an informal way to represent the boundaries of the system [Balmelli06]. The context of a system can be defined by combining different elements from different diagrams. The *Use Case* diagram is a standard diagram to represent the usage of a system and interactions between the system and the surrounding environment and actors. It is possible to use mixed diagrams where several elements from Use Case Diagram, like *Actors* and *Cases,* and from Block Diagram like *Blocks,* are compounded into one diagram. The main idea is to draw boundaries over the system and describe as clearly as possible the contextual view of a system.

The requirement section is a new addition compared with the standard UML diagram taxonomy. The requirements are traditionally represented by text.

SysML enables to represent requirement as a model element similarly to behavior or component in a system. This feature makes the model more consistent and enables the modeler to develop different relationships between the requirement and other model element. The advantage is that the modeler can track requirement satisfaction through all the different stages of the modeling process. The requirement diagram can contain both functional and non-functional requirements. Depending on the design case or domain profile several stereotyped requirements with customized attributes can be exploited. For example a specific predefined requirement stereotype for functional, structural, economical, environmental, etc. can be defined.

The structure of a new system is defined by two diagrams based on UML4SysML Class metaclass. A block diagram is usually used for describing system's static structure. A block can be any modular unit including a part, assembly, user, software, etc. It has a specific functionality and interaction interfaces. It also has a certain collection of features which can be both structural and behavioral, such as properties and operations. SysML blocks can be used through all the phases of the system specification and design. Another way to represent a system structure is to use *Internal Block* diagram. This diagram type represents the interactions between blocks and basically defines the usage of the system components. The block interacts with other elements through the ports. Several types of ports can be defined depending on the type and flow through this port. For example, standard ports are used to specify services the block can provide or is expected to provide from others. Flow ports are used to specify input and output items that can flow through this port.

Parametric diagram can be used to define a constraint network of a model. Relationships between the parameters and mathematical expressions can be defined by this diagram. The parametric model can be also an initial source for performance analysis and simulations. Through the objective function the parametric models can be used to compare alternative solutions in different design stages.

Behavioral modeling is carried out by the *Activity*, *Sequence* and *State Machine* diagrams, similarly to software system modeling concept. There are various approaches how to define a system behavior and this depends quite a lot on design domain and modeler experience. It is possible to model both continuous and discrete or hybrid systems. However the language itself has no guarantee to validate the correctness of hybrid system. From diagram point of view the activity modeling emphasizes the inputs, outputs and conditions for coordinating other behaviors. The *Sequence* diagram describes the flow of controls between actors and systems or between parts of a system. The *State Machine* defines a set of concepts that can be used for modeling discrete behavior through the finite state transition systems [SysML10]. However the actual usage of diagrams can

be treated several ways. Several approaches can be exploited depending on the engineering experience [SysMLF] or guidelines. Some approaches using activity and/or state machine:

- The choice depends on the character of the behavior whether it is process-driven or state-driven. *Activity* diagrams are more used for analysis and communication with the customer while a *State Machine* is generally used for design and communication with the developers – Tim Weilkiens.
- Oliver, et al. [OKK] point out that a functional flow block diagram (FFBD) and a state model are two representations of the same thing. The essential difference is function modeling. When the focus is on the functions the *Activity* suits more and the *State Machine* is used when the focus is on transactions between states, respectively. At an abstract level both can represent the behavior and functions from different viewpoint – Richard Sorensen.
- As *State Machine* runs continuously while the condition for that state exists similarly to software source code, e.g. do-while and do-until. An *Activity* on the other hand is a single run through the actions – Daniel C. Lanotte. The concept can be stated that the state machine is a "parent" and every state is defined by separate *Activity* diagram. When state changes another *Activity* is executed inside the current state.
- Another opposite approach is that *Activity* is taken as a "parent" and every single action has the *State Machine* specifying the inner structure of this particular action.

As it can be seen, the behavior of a system can be modeled in different approaches and there is nothing wrong in it. Even more, the software developers and system engineers have discussed without consensus about that issue as it can be seen from SysML developer Michael Latta sentiment in SysML forum [SysMLF] – "There was some debate on this point in the UML and SysML specification teams with nothing close to a consensus reached. In the end both *Activities* and *StateMachines* are ways to define a behavior of an object"

Some other diagrams like *Timing*, *Interaction Overview*, *Communication* diagram, defined by UML can also be used when needed as they are left unchanged by SysML specification.

The four view picture of modeling diagrams are shown in figure 17.

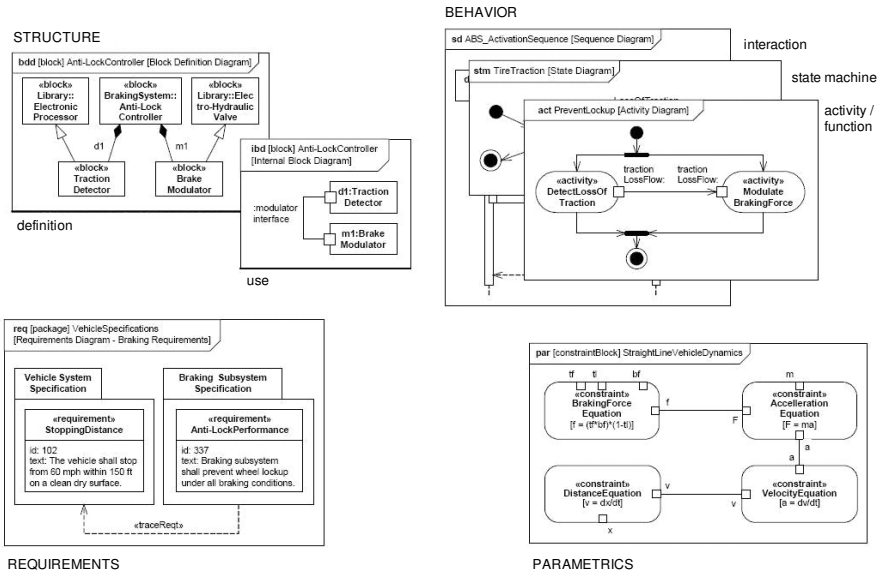

Figure 17 Four pillars of SysML [OMGSysML]

The analysis and simulation of system model can be successfully supported by a specific software package. The best known is Matlab/Simulink software. There are several companies working actively to provide a software platform enabling a link to SysML model and Simulink component [VD06]. At the moment Telelogic Rhapsody is the one that has demonstrated the working solution in this issue [TeleL07].

2.3 Artificial Intelligence methods for complex problems

Many non-traditional techniques and methods in engineering problem solving domain have come to the fore recently. One of the reasons is definitely an increase of the computing power. These opportunities allow solving the engineering tasks, which can not be described with linear differential equations and are non-deterministic. The techniques applicable for more advanced mechatronics system modeling, which are taken into account, are the following:

- Genetic & Evolutionary algorithms
  In evolutionary and genetic algorithms mainly four approaches are distinguished: genetic programming, evolutionary computation, evolutionary algorithms, and genetic algorithms. The differences are described in the Hitch-Hiker's Guide to Evolutionary Computation [HB01]. Here the GA and GP are only explained. Genetic Algorithms (GA) are basically algorithms based on natural biological evolution. The architecture of systems that implement genetic algorithms is more able

to adapt to a wide range of problems. A GA functions by generating a large set of possible solutions to a given problem. It then evaluates each of those solutions, and decides on a "fitness level" for each solution set. These solutions then breed new solutions. The parent solutions that were more "fit" are more likely to reproduce, while those that were less "fit" are more unlikely to do so. In essence, solutions are evolved over time. This way you evolve your search space scope to a point where you can find the solution [HM00]. Genetic Programming (GP) on the other hand is an extension of genetic model of learning into the space of programs. It means that population objects are not fixed-length strings but when executed they are candidate solutions for a given problem [HB01].

- Artificial neural networks
  Artificial neural networks (ANN) are collections of mathematical models that emulate some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning.
- Multi-agent systems
  A multi-agent system (MAS) is a loosely coupled network of problem-solver entities that work together to find answers to problems that are beyond the individual capabilities or knowledge of each entity [DLC89].
- Fuzzy logic
  Since the seminal work of Zadeh [Zadeh65] fuzzy logics is highly accepted in industrial applications in order to model non-linear input–output relations.

A more detailed coverage can be found in [SL03].

These methods are successfully applied in several cases for solving specific problem on optimization, machine learning, adaptive control, path planning, etc. For example, fuzzy logic is widely used in controller systems or neural networks on parameter prediction. However in many cases the theory is applied only in computer environment while calculating or simulating certain problem. Genetic algorithms are often used for finding global optimum in case of great state space. The advantage of AI methods over the traditional is the ability to search over entire solution space and they are applicable to a wide range of problems including non-continuous functions and functions involving different types of variables. Although there have been lots of research results and success when exploiting AI techniques for a certain problem, applying a single technique for a complex interdisciplinary problem is not a trivial task. Nevertheless there have been a limited number of attempts of exploiting the above techniques for design solution generation.

2.4 Artificial Intelligence based research for early design

As mentioned above, the conceptual stage of product development process is extremely important when searching an optimal solution for the final product. Although there are methodologies for modeling the conceptual design stage, most work is done according to designer experience and personal predilections. In the last decade many non-traditional techniques have been widely available, because of a rapid increase of computational power. Therefore increased interests exploiting new techniques for modeling purpose has arisen.

Some attempts to combine non-traditional techniques for early stage modeling are described hereafter in more detail.

2.4.1 Conceptual design supported by multi-agent system

Rzevski [Rzevski03] introduces the concept of mechatronic systems conceptual design with the support of multi-agents technology. Intelligent agents are used for designing mechatronics system in the conceptual phase. The introduced technology can be used to design small intelligent units capable of competing and/or co-operating with each other for a specified task and making decisions under the condition of uncertainty through a process of negotiation. The major elements of these systems are intelligent agents, which are software objects capable of communication with each other, as well as reasoning about received messages. The multi-agent technology is exploited in the software world quite often to solve complex problems without powerful central unit. Instead, small independent units are developed and in cooperation they can reach the solution for complex problem. Applying the technology for a conceptual design is a new and promising approach. Although this research presents several case studies there are no clear examples or guidelines how to develop the mentioned agents.

2.4.2 Genetic algorithms with bond graphs

The Genetic Algorithms Research and Applications Group (GARAGe) has developed a method using bond graphs and genetic programming. The solution is stated as a unified and automated design methodology for synthesizing designs for multi-domain systems [SEO03]. The approach evolves designs (represented as bond graphs) with improving performance. The design is improved in an iterative loop of synthesis, analysis, and feedback to the synthesis process.
The approach [SEO03] combines bond graphs (BG) for representing the mechatronic system model with genetic programming (GP) as a means of exploring the design space. The flow of the entire algorithm is shown in figure

18. At the very beginning the user has to specify an embryonic physical model for the target system (i.e. its interface to the external world). That determines an embryonic BG model and corresponding embryo (starting) element for a GP tree. From that, an initial population of GP trees is randomly generated. BG analysis is then performed consisting of two steps – causal analysis and state equation analysis. Based on those two steps, the fitness function is evaluated. For each evaluated and sorted population, genetic operations: selection, crossover and mutation are performed. This loop of BG analysis and GP operation is iterated until a termination condition is satisfied. The final step in instantiating a physical design would be to realize the highest-fitness BG in physical components [SEO03]. The exploited BG technique is quite popular for representing the multi-domain system. It suits also well for GP and GA trees due to its clear topology. In this research several results are generated regarding to mechatronic system design combined with genetic programming. These research results are a good basis for the further development and integration for automated concept generation of a mechatronic system.

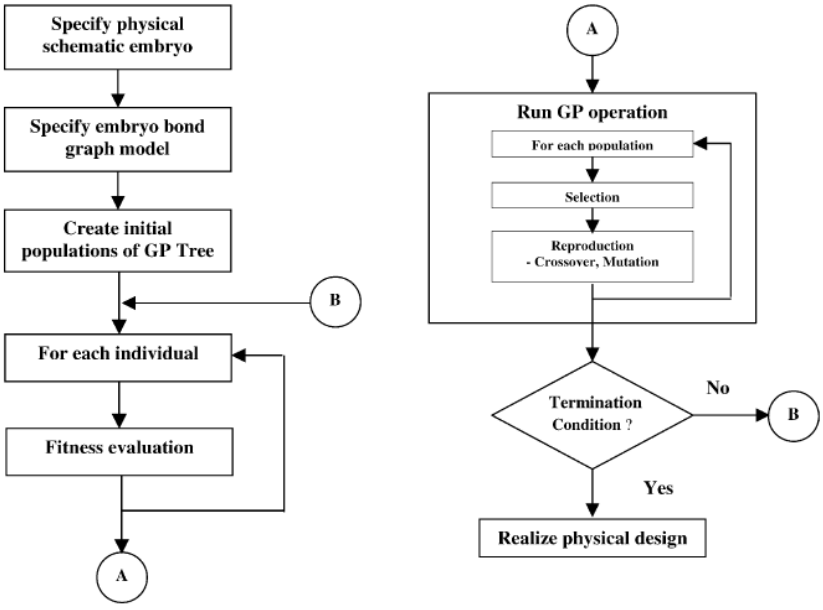

Figure 18 The GARAGe BG/GP algorithm [SEO03]

2.4.3 Bond graphs with Simulink support

The approach developed by Granda [Granda02] integrates theoretical principles of bond graph modeling and a graphical software tool called Computer Aided Modeling Program with Graphical input (CAMP-G). The tool implements the theoretical principles of bond graphs and then the model is transformed into MatLab/Simulink. The system generates first order state space differential

equations, symbolic transfer functions and Simulink S functions in source code from such as MatLab .m or .s files, which Simulink can process. The idea is to generate automatically the differential equations from the bond graph model. This means the generation of the system A, B, C, D matrices to produce the state space representation and also the computer generation of transfer functions and output equations in symbolic form. For non-linear systems this implies a set of equations where the individual non-linear constitutive relations can be accomplished as well as any conditional switches, non-linear time dependent effects and of course a form suitable for logical execution of all these by MatLab and the non-linear S functions of Simulink [Granda02].The research focuses on linking the bond graph theory and software systems. A special design environment has been developed to automatically generate BG models and then the models are linked with Matlab/Simulink. Some trivial examples are introduced and corresponding automatically generated Simulink block described. However the focus is more oriented to the mechatronic component or subsystem development rather than the whole system. A considerable result of this work is a simulation linkage of an initial bond graph model, although it is achieved by using special software system.

2.4.4 Artificial intelligence method application in machine design

A mechanical design supportive environment based on AI methods has been developed by an Estonian research group and covered in the master thesis [Tiidem01, KT04]. Visual programming for mechanical design in early design concept is used. The developed system has a graphical schema editor based on (NUT) system. Experiments have been carried out for the gear drive, drive, d-chain and power screws. The ExpertPIRZ and NUT [Tyugu91] is exploited as artificial intelligent tools for automated solution generation. The software platform is further developed and implemented in Java & C [Tyugu06]. The work is mainly aimed for specific solution generation automation, e. g. for a gear drive. For applying the proposed method to another application domain a new design package needs to be developed for this specific application.

2.5 Conclusions

1. It has been shown that new mechatronics and system engineering oriented methods and techniques have been developed within last years and in some respect accepted by the industry. A very recent achievement is SysML language specification which is a profile extension derived from UML 2.0. The very final specification is adapted by the OMG and INCOSE in 2007. At present three main frameworks are considered as state-of-art in mechatronics design and system engineering – VDI2206, DoDAF & SysML. In addition to these several research projects are running to find

optimal way to integrate different techniques. Bond graph technique is often used but not as a pure modeling tool but rather the BG concept itself integrated with other methods. The interest for AI techniques is rising due to the availability and increase of computation power. Applicability of these methods for mechatronics design process is discussed.

2. In the VDI2206 guideline the design process is described and some tools are suggested to carry on the process. The VDI2206 itself is not a tool but a set of suggestions how to carry out the mechatronics design process. However, the support for early design stage is rather poor. Connections between requirements and concept model are not covered in deep. The SysML on the other hand defines a modeling language enabling to model the requirements, system structure and system behavior in an integrated way. Although the SysML is not implemented by the industry yet, the great potential exists. The third new aspect in mechatronics early design is the AI methods. Research focus is turned from theory to practical applications in AI field and several attempts to automate the mechatronics system design are described in this chapter. Many research results where AI methods are applied use the BG concept as a model description technique. In a small system the BG is a good and universal technique but when the system complexity increases the graph grows difficult to read and understand. There is also a lack of behavior and event modeling in the BG concept. However this concept is a good input for the AI method due to the explicit representation of model topology.
3. Applying the SysML, supported by a domain specific extension to the design process according to known mechatronics design methodology is a key factor for effective and successful design in mechatronics field. The time-to-market demand as well as system complexity have increased very rapidly and automated design support is therefore needed. However the developed models have to be assured against requirements. Existing solutions, described in previous chapters, do not cover all these aspects but have good features in certain area. The BG concept exploited in several AI solutions is good enough while used by the machine to create initial abstract solution or verify the integrity of a sub-system. Nevertheless the conversation to a more human friendly modeling technique is required, otherwise it is not going to be used by practicing engineers.
4. In this chapter it is shown that exploiting AI methods the automated model development is possible. To get more advanced results these methods must be further developed and bound with methodological design process. A domain specific extension is definitely necessary to carry out the effective and optimized design in the early stage. The extension can be design patterns, widely exploited in object-oriented software design and design element libraries for reuse.

3 DESIGN FRAMEWORK

In the previous chapter several state-of-art concepts and techniques were analyzed. There has been discussed that conventional methods and tools do not cover the integration nature of mechatronics. New concepts and techniques developed over the last years deal with integrated system engineering but have weak focus on early design stage.
The proposed approach combines the advantages of new mechatronics and system engineering works with a focus on early design. The design framework is a generic model of mechatronics product design in early stage. The framework compromises requirement engineering, concept generation, concept evaluation techniques and tools. In previous chapter AI methods were studied. A semi-automated concept generation method is proposed based on these studies. The method combines mechatronics design and AI algorithms. The framework is driven by a model based approach and MDA concept. Based on that fact main MDA advantages like interoperability, portability and component reuse [POK01] are derived. Component reuse is one of the key factors in the proposed framework model. Reuse is well exploited in the software world during several years already. For example, in Japan where the software development and robotic innovation is undoubtedly the world leader component reuse is very widely used [Cusu91]. Several companies have reached the reuse level 25-50% and in some certain projects even around 90% [JGJ98]. Component reuse leads to substantial gains like reduction of time to market, defect density and maintenance cost. It is stated that the overall reduction of development cost in software design is 15% to as much as 75% in a long term project [JGJ98]. It is clear that the reductions of these factors are not automatically guaranteed, but show however the possibilities and hidden resources when applying the reuse concept. In mechatronic systems many design concepts are developed from scratch but nearly always elementary parts and standard assemblies are used in domain-specific development stage. This is even more characteristic to robotics applications where a certain sub-set of common solutions is integrated from different fields, adding certain new design elements to achieve a quite well defined goal within specified constraints. The similar reuse concept is also exploited by mechatronics design tools and is known as a standard component library. Well known mechatronics system development environments like Dymola, AMESim, Adams, etc. use component libraries for different mechatronics subdomains, e.g. mechanics, electronics, hydraulics, control, etc. and methods, e.g. multibody, bond graph, etc. However in early design the concept generation is usually developed from zero and no suitable pre-defined models are available. The quality of the result is based mostly on the experience of engineering team and available technical aids.

Applying the model based design and component reuse to the early design technique is one of the challenges of this work. Process automation is taking

place almost in every field. Some recent research works in mechatronics design are presented in the previous chapter, where design solutions were developed automatically. Based on these experiences and research results, an early design semi-automated concept is proposed and described in papers [Sell04b, Sell04c]. The automated design is not however always suitable and therefore the solution is proposed as a reference or an alternative and complementary option for engineering teams. It can be successfully combined with traditional methods and ways the engineering teams are used to use.

To achieve the reuse benefit in conceptual design a template system is introduced. The idea can be compared with the software object-oriented design where design patterns are exploited. The conceptual model template system itself is a generic approach and is not restricted only to robotic applications. The initial concept is proposed by the author in paper [ST03].

3.1 Semi-automated conceptual design

The proposed generic early stage development approach consists of techniques and tools for three integrated sub-stages: requirement modeling, conceptual solution modeling and conceptual solution simulations [Sell05]. All stages are supported by the specific toolkit and algorithm libraries. The toolkit consists of rules and template libraries for the specific application domain. A template class from the template library provides a parameterized description of the model element, subsystem and working principle, specifying its attributes and operations. The conceptual solution modeling is a design candidate development process where alternative solutions are developed and evaluated. The generic process concept developed tools are illustrated in figure 19.

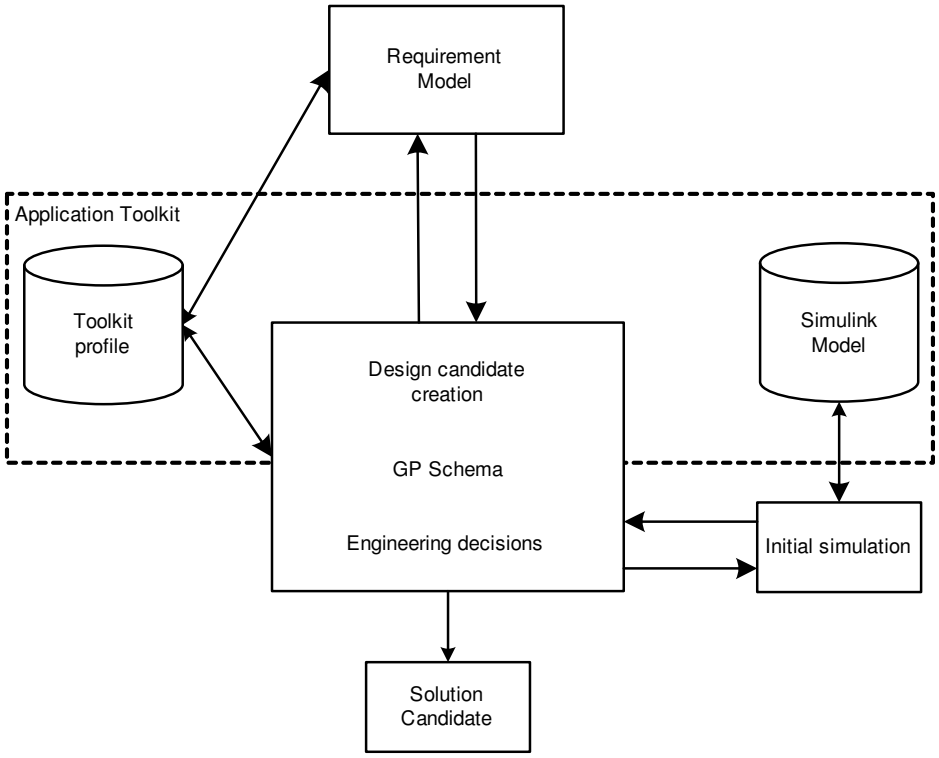

Figure 19 Generic concept of early stage design

The automated solution candidate creation process is proposed to carry out with the interaction between GP schema and engineering team. The specific GP schema develops a base design candidate based on the so-called embryo model. The prototype of this kind of GP schema has been developed by the GARAGe Group [SEO03] from Michigan State University, discussed in detail in chapter 3.4.2. An analogous schema is proposed to use in early design semi-automated solution development process in this framework.

The GA and GP concepts for real system design generation have been in research focus in many areas. Probably the most known work is [KBAKD97] where GP concept is used to automatically synthesize the topology and size of analog circuits, including low-pass filters, controllers, and amplifiers. There are several other works in automated analog circuit design based on GA and GP concepts [Grim00, LC99, FSGRZ01]. The dynamic system automated design is studied by Tay a. o. [TFB98], where bond graphs are used as a system representation.

Exploiting the BG concept tends to be popular in automatic design process as well as conceptual design [Coatanea05, Granda02, SEO03, TS05]. Although the concept has disadvantages and is not widely used by practicing engineers, it suits well for different algorithms due its clear semantics and universal representation

of integrated system's topology. It suits well also for GP algorithm proposed by the GARAGe group where the input and working model is expected to be bond graph model. Therefore the prototype algorithm proposed to be exploited within this framework use a BG concept as well. However the disadvantage of BG is the reading difficulty in case of complex systems. In addition the BG is representing the system structure but not the behavior and context which is important when developing an integrated system. According these concerns the proposed modeling approach exploits the BG representation for automated design candidate creation while the rest of the process is carried out by the support of SysML & application profile extension. It needs some conversation between BG model and SysML model. Initial concept is generated by the engineering team, inversely the usual GP initial input which is generated randomly. Some common solution development techniques like brainstorming, 635 method, Delphi method, etc. can be successfully exploited as preferred by the particular engineering team.

The process is roughly divided into the following steps:

- The initial solution developed by the engineering team as denoted before, is an input for GP schema and has to be represented by the bond graph technique as the algorithm accepts only this format.
- The concept generation and evaluation is an interaction between engineering team and GP schema. The algorithm generates design candidate according to initial input and requirements. The output of the GP schema is BG representation of design candidate and has to be converted to the SysML. The solution itself is modeled by the SysML domain specific toolkit.
- After the end of GP run engineers start to evaluate the result developed by the algorithm. When necessary they can set additional constraints and limitations. Depending on the maturity of the output, additional runs can be executed. Several outputs can be collected to start a more deep evaluation through the initial simulation procedure.
- Different developed solutions are specified with simulation parameters and bound with a simulation algorithm in the template library. The initial simulation is executed and results are compared. The interaction number depends on the problem complexity and application domain specifics.
- The overall result is an optimized and simulated solution candidate which will be further developed in detail design stages.

The developed profile supports the early design, but can be easily expanded to support the whole design process. Therefore it is advisable to exploit the tools used in conceptual design also through the whole design process with the support of domain specific tools. This guarantees the model consistency and compatibility with requirement model. An additional feature is automated documentation generation through the whole process, including design changes.

3.2 Framework application

3.2.1 Modeling approach

The modeling approach adopts the V-macromodel proposed by VDI2206 guideline [VDI04] which is initially adopted from software development process [Brö95]. The V-model is transformed to reflect the conceptual stage where the product of cycles is the solution concept. In VDI V-model, the System Design part reflects partly the early design dealt in this thesis.

The modeling approach of this research refines the conceptual modeling and emphasizes the requirement engineering and linking requirements to the model element. The adopted V-model focusing on the conceptual design is presented in figure 20.

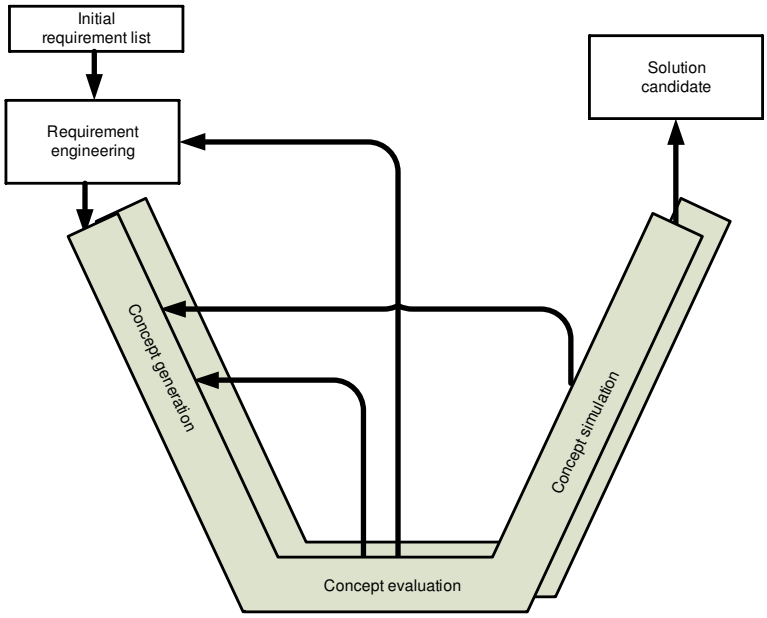

Figure 20 V-model for conceptual stage

According to the proposed adoption the process starts with requirement engineering where the initial requirement list is expected to be provided by the customer. In most cases it is represented as a text document and graphics. Based on this documentation the requirement engineering stage refines and formalizes the requirements followed by the toolkit formalization rules. The requirement engineering has several stages where the models can be refined according to progress of development. The concept generation and evaluation is interactive process carried out either by the proposed AI algorithms, discussed in chapter

3.1, or by the engineering team as usual. Whatever method is used the concept generation sub-steps have to be dealt anyway. When the design concept is mature enough it is bound with simulation model and executed in the simulation environment. Several pre-defined simulation algorithms are available in toolkit libraries and additional algorithms can be developed according to the current need. For example in mobile platform toolkit, the performance simulation model is linked with a design candidate and a key performance parameter evaluation algorithm is executed. The solution generation should not end with the first cycle but has to develop several solution candidates which can be evaluated against each other and compared with requirements. The best matching solution candidate will be selected and passed to the next stage.

The mechatronics design process involves several related models in concept generation process. Each model describes a specific aspect of the system like behavior, static structure, dynamics, etc. and has available diagrams and documents in a predefined format. In this approach SysML with toolkit extension used as the modeling language is. The toolkit is derived from SysML by the standard extension mechanism. The SysML itself is derived from UML 2.0 by the same mechanism and is consistent with the OMG Meta Object Facility (MOF). According to this mechanism the language can be extended by the stereotypes or restricted by the subset of metamodels. In addition to profile the model libraries are included to embrace the design templates with reusable model elements sub-systems parts, working principles and so on.

3.2.2 Template libraries

In our case a template is defined as a parameterized model element that describes or identifies the pattern for a group of model elements of a particular type. The templates are not directly used in models. Instead, they will be first initiated by binding the parameters to actual values, in the similar way as software patterns used in Rational Software Architecture [RSA]. The binding between a template and a model element generates a new model element based on the template like a specific drive or transmission mechanism, robot leg or manipulator mechanism etc. Then it can be used as bound element to model part of a system [RSA].

In this approach different types of design templates are developed and organized into libraries. These templates are used for robotics applications whereby the main goal is to retain advantages for early design stages of usage of component libraries offering design alternatives verification and fine tuning capabilities at the same time. Derived from the effect of the real-world nature the mechatronic system design templates, as a part of the toolkit, are grouped as *Diagram* template library, *Model & Principle* template library and *Algorithm* template library. Template package is illustrated in figure 21.

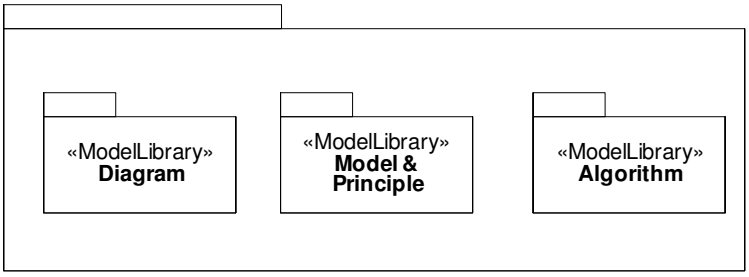

Figure 21 Template package

Each library consists of several sub-categories. The detail content and structure of the library depends on the application toolkit. Nevertheless the standard subcategories are proposed as follows:

- Diagram template library (system view)
  - General Requirement
  - Subsystem Requirement
  - System Context
  - System Behavior
  - System Static Structure
  - Subsystem Interaction
  - System Dynamics
- Model & Principle template library (component view)
  - Mechanics
  - Hydraulics
  - Electronics
  - Control
  - Real-Time
  - Robotic
- Algorithm template library
  - Performance
  - Power Consumption
  - Robustness
  - Reliability
  - Control

Additional subcategories are application toolkit specific and specified in the toolkit specification where application domain characteristics are taken into account. However the subcategories overlap between similar domain toolkits and are therefore stored into general library with domain tags.

3.3 Mobile Platform Toolkit

The robotic platform and its design methods have gained a constant by growing attention in several sectors. Recently the European Robotic Platform network EUROP [EUROP] was established to support new companies and networks in maintaining and strengthening the Europe position in the robotics technology field. Many other similar actions taken and networks [CLAWAR, EURON] indicate a strong focus on the robotic sector. These actions produce a clear need for effective and compatible development methods suitable for SME type companies as well as research institutes and other stakeholders. Based on these trends Conceptual Framework of Mobile Robotic Platform has been selected. The Mobile Platform Toolkit (MPT) is an application example, and based on this example additional profiles for different mechatronics sub-domains can be easily developed. It is clear that some parts of toolkit will overlap on the future and these parts, e.g. *Principle Library,* can be used crosswise.
The mobile platform has been dealt with as a generalization of different types of (mainly electrical) vehicles. A mobile platform can have several totally different locomotion types. An overview of existing locomotion applications is given in technical report [Brooke03]. However it is clear that a conventional wheel is the most common locomotion type and therefore the main focus is on this type. In addition, the special focus is on hybrid locomotion and dynamically configurable locomotion. During the doctoral research a unique dynamically configurable wheel-leg (wheg) has been invented and patent application compiled.

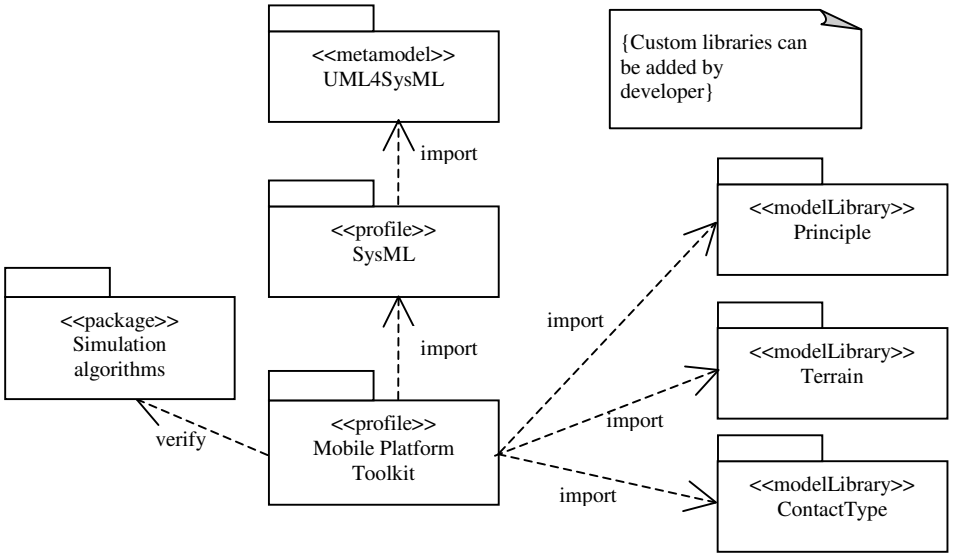

Figure 22 Mobile Platform toolkit structure

The toolkit is defined as a SysML profile and external simulation package algorithms. The profile itself consists of template libraries, diagram extensions and model libraries. Standard model libraries are *Principle*, *Terrain* and *ContactType*.
The model library *Principle* is a collection of standard mechatronic sub-systems, elements and working principles. This library is the most similar to the existing design software packages part library concept where standard parts are defined and collected into categories.
The MPT *Principle* library contains also the working principles and subsystems formulated in SysML and an extended profile. This means that similar subsystems can be found in different libraries but the abstraction level is higher and the subsystem is defined in formal language rather than physical component. The boundaries between the physical domains are not very sharply defined and can be determined later on detail design stage. The model can be developed by linking the subsystems and working principles from library with loosely coupled relations whereas the certain key parameters are defined. These parameters are in most cases derived from the requirement model and are related to many other parameters of a system. For example, a simple mathematical model linking different parameters is defined by *Parametric* diagram and key parameters are defined with extended stereotypes.

Terrain and *ContactType* library are holding the parameters of different terrain and vehicle-soil contact. The reason for establishing the *Terrain* and *ContactType* library was the mobile platform performance analysis and simulation need. Depending on the required terrain capabilities, the mobile robot must deal with obstacles, surface characteristics, slopes, etc. Terrain properties affect greatly the robot design where a smart and optimal design can save the energy, improve the performance, optimize the budget and so on. These parameterized models can be linked to the design element or design candidate and used in initial simulations.
The terrain vehicle relation in off-road conditions has been studied for a long time already. However the terrain properties are not very easy to define and mostly the statistical and heuristic methods must be exploited [Bekker69]. The Terrain library holds the terrain model described with the following characteristics:

- Terrain structure
- Holes and humps
- Obstacle density and distribution
- Geometry of terrain obstacle
- Irregularities
- Mechanical properties of soil:
    - viscosity

- moisture
- density
- nature of the surface (ice slice, low blanket, vegetation)

Although the terrain is described by several parameters the real terrain condition is never fully met. However, simulating a locomotion model linked with different terrain models gives the designer a valuable feedback when comparing concurrent design solutions. Some parameters have a bigger effect than others in different situations. For example, the lack of vehicle traction may be caused by extraneous situations like rain or freezing in outside driving and initial surface roughness has only incidental relation to it. Terrain classification based on geomorphology or botanical concept is not suitable for mobile locomotion study [Bekker69]. Therefore the *Terrain* library is introduced and only selected parameters are included to the terrain description. The structure of a terrain library objects is described in figure 23.

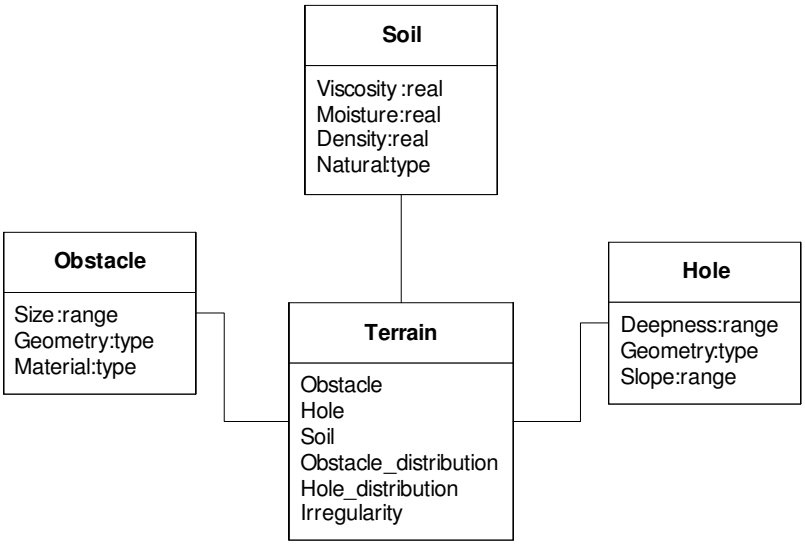

Figure 23 Terrain library structure

The *ContactType* specifies the wheel-surface or actuator-surface (in case of non wheel locomotion) contact characteristics for advanced locomotion analyses and simulation. The *ContactType* is very tightly related to terrain and is used more for mechanical analysis of wheel-terrain interaction. The *ContactType* library element can analogously be connected with the model element and used for example wear analysis. One important end-application is railway vehicles where disc-disc contact can be picked from the library and the wear analysis in various design solutions and environmental conditions can be executed. In a similar way

the tire-road friction simulation can be carried out with the connection of *ContactType* library element.

The profile itself consists of pre-defined diagram templates and extended diagram elements – stereotypes. Requirement and design template included into Mobile Platform Toolkit is described in tables 2 and 4. These templates are specific to the mobile robotic platform. However some of them can be successfully exploited when developing a similar mechatronic system. For example navigation templates can be used when developing an automatic car navigation system for automotive industry. Depending on a particular application domain the corresponding templates may still need smaller or bigger adjustments.

3.3.1 Requirements model

Requirements are the foundation of the project. Every requirement is tightly related to the cost and end solution, therefore the requirement modeling and analysis must be carried out with great attention. Big changes in requirements on later design process may increase significally the cost of the whole project.

Requirements arise from various sources like customer needs, regulations and legislation, organization environment restrictions and technology availability, etc. The requirement definition is a complex process and typically includes performance analysis, trade studies, constraint evaluation and cost analysis. Depending on the problem domain different requirement templates can be defined.

Toolkit requirement extension specifies additional stereotypes derived from the base stereotype <<*Requirement*>>. The profile enhancement exploits some non-normative requirement stereotypes proposed in SysML Appendix C [SYSML10] and additional profile specific elements. Following general mechatronics design and robotic platform specific aspects are considered: structure, functions, performance, environment, cost, payload, navigation, safety and performance. In MPT a requirement is defined by the textual representation of the constraints and the values of the parameters. Textual representation can refer to more detail document e.g. requirements defined by the client. Default parameters for extended requirement element are *Weight*, *Risk*, *OptimizationDirection*, and *Source*. Optional parameters are *ConsistentStandard* and *VerifyMethod*. Based on the extended requirement several sub-level requirement elements are specified. Every sub-element can have additional parameters and constraints. Base structure of requirement model hierarchy is described in figure 24.

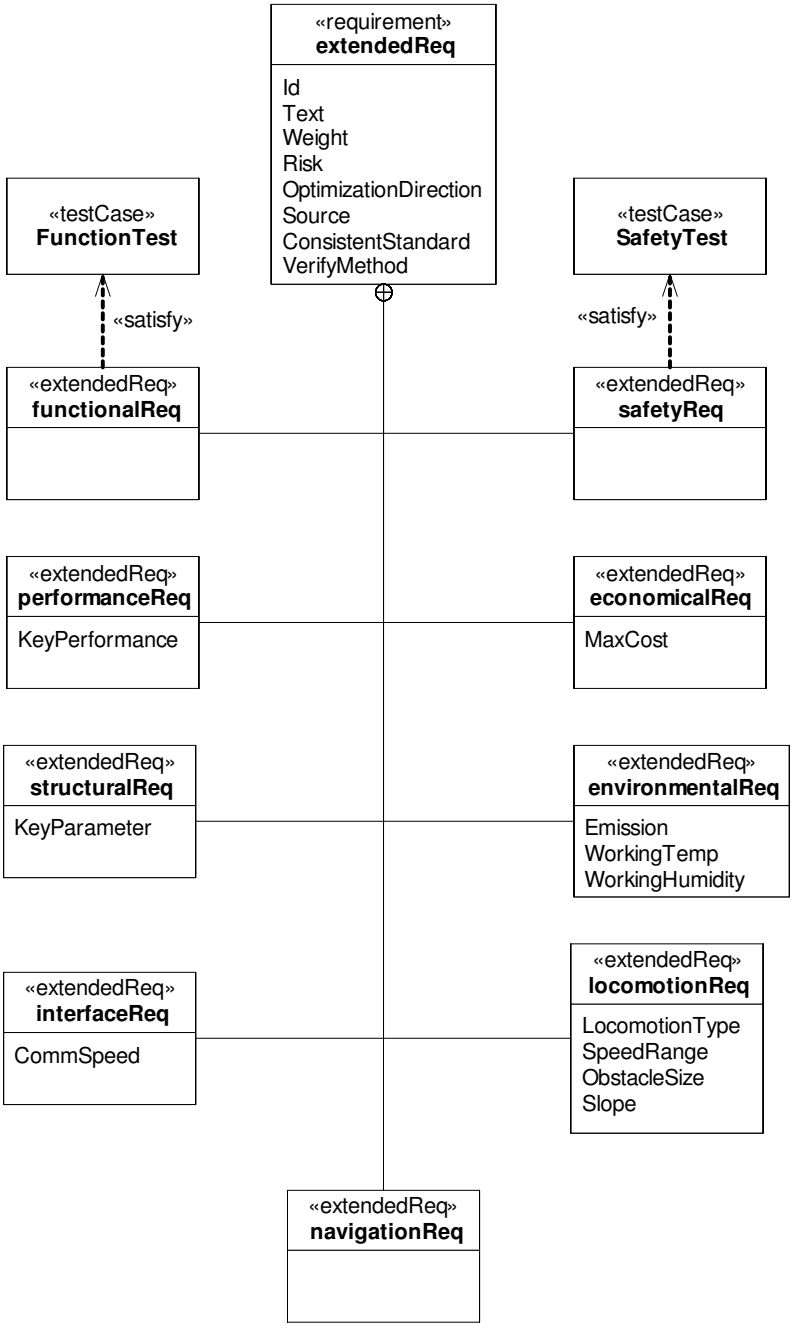

Figure 24 Requirement template elements

During the requirement modeling the suitable template will be instantiated by binding the parameters to the actual values. The single requirement element or the entire template for the domain problem can be initiated depending on the

actual need. Every template element has general parameters and optional parameters. Only necessary parameters can be bound.

Table 1 Add-on stereotypes in MPT

Requirement /Parent	Parameters	Description	Constraint
<<extendedReq>> <<requirement>>	Weight Risk OptimizationDirection Source ConsistentStandard VerifyMethod	Extended stereotype – The base for extended stereotypes.	n/a
<<functionalReq>> <<extendedReq>>	n/a	Platform basic functions – operations and behavior the system must perform.	Must be bound with a *testCase.*
<<performanceReq>> <<extendedReq>>	KeyPerformance	Measures quantitatively the extent to which a system satisfies a required capability.	Satisfied by a value property and block constraint.
<<structuralReq>> <<extendedReq>>	KeyParameter	Specifies the physical dimensions and characteristics of a system.	Satisfied by the block element.
<<safetyReq>> <<extendedReq>>	n/a	Specifies the safety conditions and standards the system must meet.	Must be verified by the *testCase* specified with given standard.
<<interfaceReq>> <<extendedReq>>	ComSpeed	Specifies interfaces between subsystems and/or environment.	n/a

<<economicalReq>> <<extendedReq>>	MaxCost	Specifies the economical constraints for a system or sub-system.	n/a
<<environmentalReq>> <<extendedReq>>	Emission WorkingTemp WorkingHumidity	Environmental requirements.	n/a
<<navigationReq>> <<extendedReq>>	n/a	Specifies the platform navigation requirements.	n/a
<<locomotionReq>> <<extendedReq>>	LocomotionType SpeedRange ObstacleSize Slope	Locomotion requirements of mobile robot.	n/a

The requirement models are developed in cycles of interaction levels. Depending on the problem complexity and quality of the customer requirement list, the number of interaction levels can vary. The proposed model sets three standard interaction levels as a general guideline for mobile platform developer.

Level I Base diagrams, directly derived from the requirement list. Every requirement element is specified by *id* and *text* or link to full text in external documentation. The requirements are decomposed into sublevels as deep as specified in initial requirement list.

Level II Requirement elements are refined with additional parameters defined by the toolkit. Some parameters can be left blank or defined with a range of values. Main requirements are connected to the behavior and structural elements in *Activity*, *State Machine* or *Block* diagram.

Level III Every base requirement has a refined verification link with *testCase* specification. Initial simulation may be passed in this stage and corresponding requirement property assigned.

Table 2 Requirement templates in MPT

Template type	Template name	Level	Description
Requirement	req_general	I	General requirement hierarchy. System and sub-system generalization structure.
	req_platform	II	Platform Structure. The main requirement of platform sub-systems.
	req_energy	III	Energy requirement specification.
	req_locomotion	III	Locomotion requirement specification.
	req_navigation	III	Navigation requirement specification.
	req_payload	III	Payload requirement specification.
Use Case	uc_context	I	Context specification of given problem. System boundaries and external interaction.

Described requirement templates are pre-defined in MPT enabling to start the conceptual modeling process quickly with initial models on the front. Although the profile defines the mobile robotic platform specific templates, some of them are more general (e.g. *req_general*, *uc_context*) and can be used across the systems, or some are universal for different mechatronics product (e.g. *req_energy*, *req_navigation*). A template system has an open architecture and every company or engineering team has possibility to build his own additional templates or even template libraries.

3.3.2 Design model

The conceptual design level is actually the first stage to start with the development of the desired system which should correspond in maximum and optimal way to the set of the requirements. The stage is tightly related to the previous one – the requirement modeling.

Tools for modeling the system structure and component interactions are *Block Definition* diagram, *Internal Block* diagram and *Parametric* diagram. The static structure of the system is represented by the *Block Definition* diagram. The robot structure is represented as a component hierarchy or the system classification tree. In this case different relationships could be used. In the *Principle* library two different types of design element units can be found.

- A single block, representing a logical unit of a system. This is usually a physical component like a motor, an electronic device or a control algorithm.
- Group of blocks, representing a logical model of a subsystem. This is usually a smaller unit of the system like a drive, an operator interface or an obstacle avoidance algorithm.

The block element encapsulates parameters and operations that can be hidden in certain design level. The system behavior is described with *Activity* and/or *State Machine* and *Sequence* diagram. The particular behavior model corresponds to specific requirement element defined in requirement model. The model specifies the actual way how certain requirement is satisfied and how the system or subsystem behaves in different defined conditions. These conditions are either use scenarios defined by *Use Case* or environmental conditions. Environmental aspects are temperature, humidity, pollution, etc. which can be simulated when appropriate simulation algorithm is available in *Algorithm Library* and terrain conditions where behavior model can be linked with selected terrain profile from *Terrain Library.* The *Activity* and *State Machine* will specify also the verification procedures and test cases to ensure the requirement satisfaction.

The toolkit concept design extension specifies additional stereotypes derived from base stereotypes: *<<Block>>, <<Actor>>, <<Action>>, <<State>> and <<Bindingconnector>>*. The following figure shows the abstract syntax of extensions.

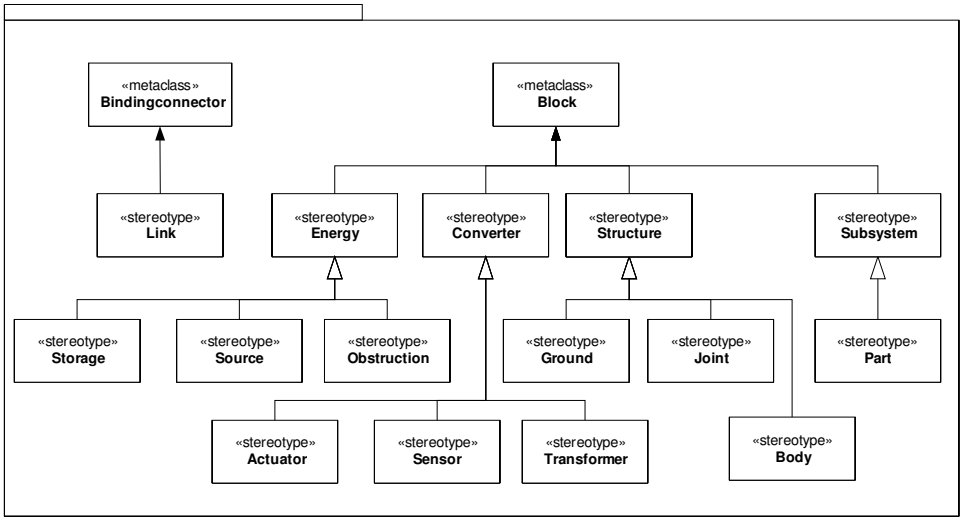

Figure 25 Abstract syntax of structure design stereotype extensions

It can be argued about the stereotype selection and definition but the stereotype derivation mechanism allows the designer to define new objects based on the metaclasses or toolkit stereotypes. Therefore the general approach is proposed

here which can be expanded according to the need of a particular case. When derivation mechanism is followed the new introduced block will be fully compatible with the existing software supporting the UML/SysML design. At the present moment (2007) MDG Technology for SysML (Sparx Systems), Rhapsody (Telelogic), SysML Toolkit (EmbeddedPlus), Artisan Studio (Artisan Software), TAU G2 (Telelogic) and MagicDraw (No Magic) have stated the support for SysML/UML modeling.

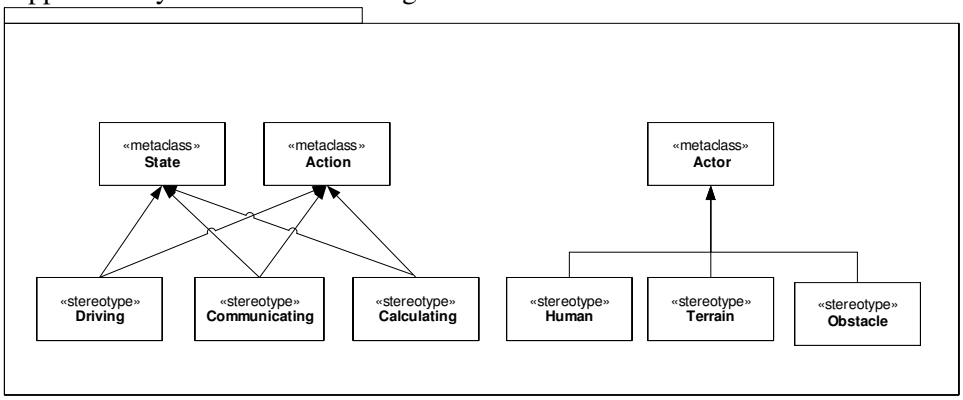

Figure 26 Abstract syntax of behavior design stereotype extensions

Table 3 Stereotype description

Stereotype / Parent	Parameters	Description
<<subsystem>> <<block>>	n/a	Designates the well bounded subsystem usually with well defined inputs and outputs. Mechanical assemblies can be defined as subsystems.
<<part>> <<subsystem>>	n/a	The part stereotype is a general entity for physical parts. It can be used as a black box for undefined parts used in platform design.
<<converter>> <<block>>	Input Output	General stereotype for different kinds of energy or movement converting elements.
<<sensor>> <<converter>>	Type	Common sensor stereotype. Designates all components measuring some physical phenomena and outputting a signal.
<<actuator>> <<converter>>	Type ForceTorque	All type of actuators: electrical motors, hydraulic, electromagnetic, etc. actuators. Converts energy from one

		domain to other.
<<transformer>> <<converter>>	TransferCoefficient	Designates converters which convert energy or movement in a same domain. For example transmission, electrical transformer, etc.
<<structure>> <<block>>	Dimension Material Mass GravityCenter	General element for construction material, frames, joints, etc.
<<body>> <<structure>>	n/a	Body construction elements. Rigid structure elements.
<<joint>> <<structure>>	Type	Joints, e.g. rotational pairs, linear joints, etc.
<<ground>> <<structure>>	n/a	General ground point.
<<energy>> <<body>>	n/a	Energy elements of platform.
<<storage>> <<energy>>	n/a	Energy storage elements, e.g. springs, capacitors.
<<source>> <<energy>>	n/a	Energy sources, e.g. batteries, fuel cells, compulsion engine, etc.
<<obstruction>> <<energy>>	n/a	Energy consumers, e.g. resistor, damper, etc.
<<bindingconnector>> <<link>>	flow effort	Special connector for elements, used when flow and effort have to be transferred.
<<terrain>> <<actor>>	n/a	Specific external user which may be linked to the Terrain Library.
<<human>> <<actor>>	n/a	External user with specific interface, i.e. HMI.
<<obstacle>> <<actor>>	n/a	External object which affects the platform capability.
<<driving>> <<state>>,<<action>>	n/a	General activity or state reflecting the platform movement or driving commands.
<<communicating>> <<state>>,<<action>>	n/a	General activity or state reflecting the platform communication procedure.
<<calculating>> <<state>>,<<action>>	n/a	General activity or state reflecting the platform calculating needed parameters.

The MPT *Diagram* library incorporates template categories for concept design described in table 4. Diagram templates are divided logically according to SysML diagram types.

Table 4 Design templates in Mobile Platform Toolkit

Template type	Template name	Level	Description
Block definition	bdd_general	I	General sub-system hierarchy. System and sub-system generalization structure.
	bdd_body	II	Chassis subsystem.
	bdd_energy	II	Energy subsystem.
	bdd _locomotion	II	Locomotion subsystem.
	bdd_sensor	II	Sensor subsystem.
	bdd_communication	II	Communication subsystem.
Internal block	ibd_system	II	Subsystem interaction (port types and flows) specification.
	ibd_energy	III	Energy component interaction.
	ibd_locomotion	III	Locomotion component interaction.
	ibd_control	III	Control, sensor & communication component interaction.
Activity	act_system	II	Main functions.
	act_autolocomotion	III	Automatic locomotion function.
	act_pathplanning	III	Path planning function.
	act_navigation	III	Navigation function.
	act_obstacleavoidance	III	Obstacle avoidance function.

The interaction levels are defined similarly to requirement level definition.

Level I System and subsystem hierarchy, subsystem general interactions are defined. The main functionality and system states are indicated.

Level II Subsystems are opened and defined in general 'black box' components. The parameters of subsystems are initiated. Subsystem inner activities are distinguished.

Level III Component parameters are defined and cross diagram relationships established. The allocation diagram can be used to define these

relations. System and subsystem behavior is opened as detail as needed. The parameters of model and component are defined according to selected simulation model need.

3.3.3 Linking design and requirement templates

In the product development process it is extremely important to follow the requirements during the whole development process. It is even more important in mechatronics domain and robotics, because the integration of classical domains sets much higher demands for the design solution in terms of system validation and assurance of desired properties. The verification and validation for a mechatronic system described in VDI2206 are relevant also for the conceptual stage, dealing with first analysis and simulation schemes to determine the best solution candidate.

The SysML specification gives us good tools for linking requirements with design templates. Several relationships are specified for templates, enabling designer to relate requirements with model elements or with another requirement. Figure 27 describes by the example how a model element could satisfy a defined requirement.

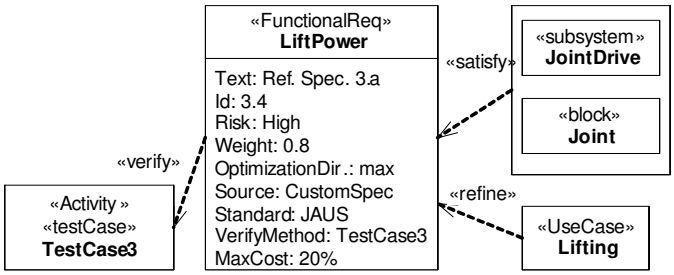

Figure 27 Linking requirements and design

In this example *JointDrive* subsystem and *Joint* block are implemented to fulfill the functional requirement *LiftPower.* Model elements are represented here as black boxes and are linked to the block definition and the internal block diagram. The *refine* relationship is used to describe how the model element can be used to refine the text based requirement. In the figure the *UseCase Lifting* is a context diagram, where the system functionality and interactions with the environment are graphically presented. This gives additional knowledge about the requirement background and rationales. In terms of validation and verification the *verify* relationship is the most important connection between the requirement and the design model. The *verify* relationship describes how the test case verifies the requirement. Verification methods are available in the template library and can be bound with the particular requirement. The verification mechanism can be an abstract numerical algorithm or an executable simulation

model. In MPT the test case is defined as simulation models. For example performance simulation algorithm using external pre-defined Simulink model.

3.3.4 Simulation model

The simulation model is defined by the *Constraint Blocks* model or specific model element. The simulation model can be linked to the design candidate or the simulation model can be included to the design element through the *simu* stereotype. In case of the standard simulation procedure the appropriate model can be picked from the simulation *Algorithm* library. The specific relations between the system parameters exist and these are defined by the *Parametric* model. The idea is that parametric diagram defines the mathematics and constraints of a subsystem and based on this model it is possible to generate the Simulink block model, namely S-function. Later on the same Simulink model can be linked to component in *bdd* model.
The MPT defines two additional parameter types: *KeyConstraint* & *PerformanceConstraint.* These stereotypes add to the *Constraint Block* parameters additional typed parameters – *KeyParameters* and *KeyPerformanceParameters* respectively.

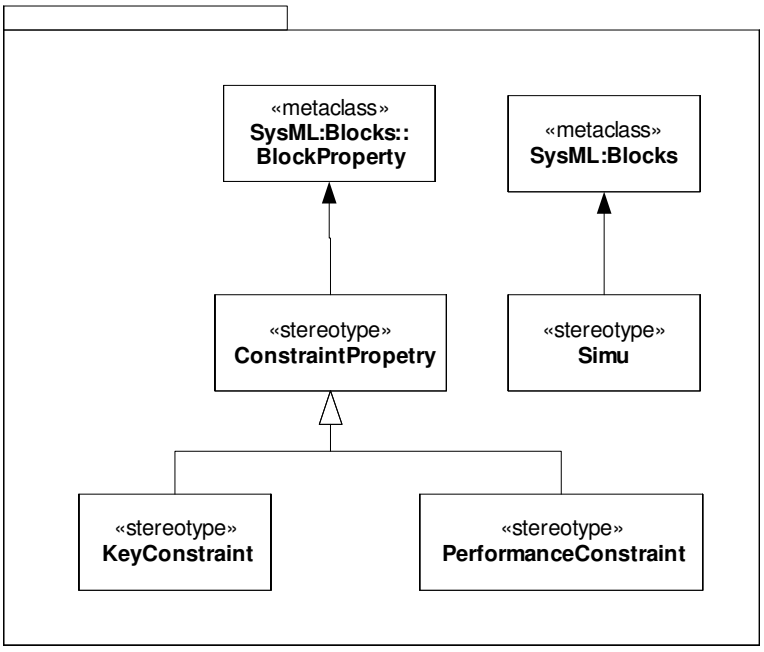

Figure 28 Parametric and simulation stereotypes

Key parameters are the typed parameters which affect very strongly the system behavior. Key performance parameters are the most important performance

requirements which are requested by the customer and cannot be yielded. These parameters are also very important from the point of view of simulation.
Simu stereotype has the following additional defined parameters:

- InitParams
- SimulinkModelRef
- SimulinkCodeRef
- SInputs
- SOutputs

The following figure illustrates the usage of the stereotypes and the parameter network.

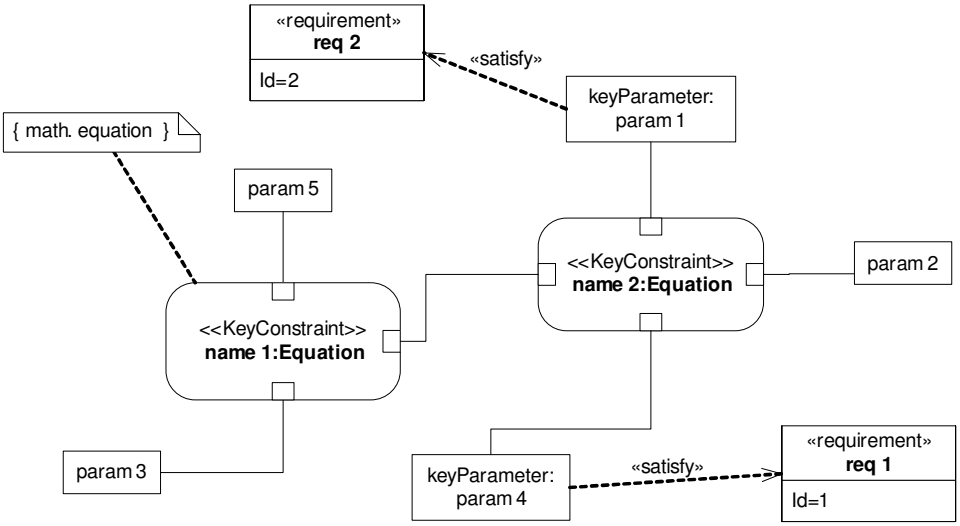

Figure 29 Parameter interrelationships

The constraint blocks are denoted here as *Equation* which describes the mathematical relations and connections between the parameters. This diagram is in most cases the source for generating (or selecting from library, if exists) the simulation model in target environment. The Simulink software is the most known and therefore the Toolkit implementation examples are oriented to this software package.

For the model verification and the solution candidates comparison against each other, the system is divided into subsystems using subsystem metamodels. These subsystem metamodels can also be divided into component models. MatLab/Simulink simulation model has a hierarchical structure and each block can be flexibly composed from configurable sub-blocks. In order to execute the

simulation model, it has to be linked with the source model developed with Toolkit. The linkage is based on defining the relationships between the design model block and the simulation model component. The MPT defines a simulation stereotype <<*simu*>>. The *simu* is a stereotype of the block metaclass. It is possible to define a certain block as <<*simu*>> and link it with a pre-defined Simulink model. The stereotype and its usage are shown in figure 30.

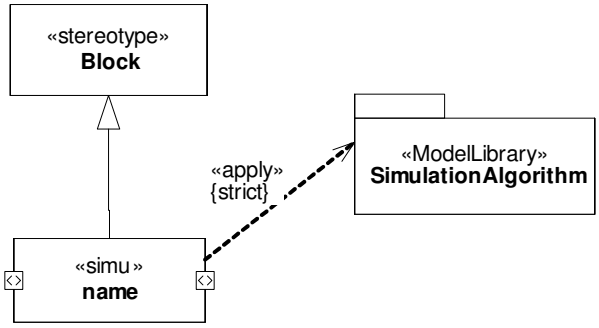

Figure 30 Simu stereotype

3.4 Conclusions

1. In this chapter the generic concept for early design process is proposed. The base methodology has been developed and denoted as Conceptual Framework Model (CFM). The concept features a semi-autonomous design concept generation, template libraries and initial simulation. The concept model itself is a general model with suggested methods and techniques. The real design process depends on the problem domain, complexity and several other aspects. The semi-automated design concept generation is described. The generated design model is mapped with Simulink pre-defined model and a scheme for utilizing other software tools (in case of available mapping interface) is proposed.
2. Early stage design templates have been introduced as practical pre-defined design elements. Template libraries have been composed for requirement engineering and model design purpose. The template concept makes it possible to reuse the conceptual models and its elements bringing the advantage of reuse into mechatronics system conceptual design. The benefits include faster time to market, optimized design solutions and effective resource allocation.
3. Based on the template concept and CFM, the domain specific Mobile Platform Toolkit (MPT) is developed. To fulfill the requirements described initially by the customer and modeled in the requirement engineering stage, the model and requirement mapping guidelines are described. According to the profile rules certain stereotypes demand certain design elements to satisfy or refine the particular requirement. These are described as extended stereotype constraints. Requirement element verification is ensured by the test case which is an activity or a state machine.
4. The result of this chapter is a definition of the design framework, including the generic model and the template library. The toolkit for the mobile robot platform design is developed. As it is well known that a new technique is best acquired with the learning-by-example approach the implementation example should be discussed. The proposed early design concept and Mobile Platform Toolkit particularly is implemented in a real mobile platform design case and covered in detail in the next chapter.

4 IMPLEMENTATION PROCESS

The developed methodology and tool are intended to help speed up the conceptual design process, improving the efficiency and productivity at the same time. The guideline is intended to summarize the methodological approach described in the previous chapters and the implementation example is a practical case study for designing a mobile robotic platform.

When starting to design a new mechatronic product according to the proposed methodology the following main steps must be passed:

- Requirement analysis and model development
    - Requirement list creation (jointly with customer)
    - Requirement feasibility analysis and decomposition
    - Requirement model development – supported by requirement templates found in MPT *Template* library
- Initial concept solution development according to requirement model. When automated design candidate generation is intended to use (according to GP concept described in chapter 3.1) the initial model has to be developed in bond graph technique. In case of manual design SysML and extensions should be used. In both cases more than one solution candidate should be developed for a given problem.
- Developed solution candidates will be initially evaluated against requirements and appropriate constraints and limits are set, if necessary. Design candidates are improved by the automated GP algorithm or manually by the engineering team.
- When mature enough rational candidates will be selected appropriate *Parametric* diagrams should be developed describing the system dynamics and mathematical relations of the system parameters.
- A design candidate model will be connected with a simulation model picked from *Algorithm* library or developed for this special case. Simulation is either carried out for a specific model element (e.g. process controller) or for the whole model (e.g. performance simulation) finding the optimal component parameters and combination.
- After the concurrent design candidate simulation the optimum solution is selected and refined if necessary.
- A refined design candidate is the conceptual solution which is passed to the product detail stage. Based on this concept all functions and components will be implemented.

Although the steps are described here as a sequence, the real process is integrated in relation between sub-stages and developers. Solutions and functions have to be directly derived from the requirements and assured.

The implementation process is a case study of the developed methodology. The task is to demonstrate the use of the Mobile Platform Toolkit in real design process. The practical realization of the profile is targeted for two different systems. It is demonstrated, based on the analysis of the existing platform and on the synthesis of a new design. The practical examples are dealt with concurrently and appropriate explanations are provided.

The analytical case study is based on the Hybritor platform and Workpartner application developed in Automation Technology Laboratory of Helsinki University of Technology during 1998-2006. The Hybritor platform has four legs equipped with wheels and active body joints. Each leg has three actively controlled joints and a wheel actuator. The platform has a modular structure and unified communication interfaces [HLSYK03]. Hybritor is a base platform for mobile service robot called Workpartner. Workpartner is a high adaptive service robot for outdoor tasks. The platform is equipped with two-hand human like manipulator, which can be used for manipulations and handling tools. The subsystems of Workpartner are divided according to generic functions [Ylönen06]:

- Locomotion subsystem
- Manipulator
- Energy subsystem
- Navigation and perception subsystem
- Control subsystem
- Human-machine interface

The Hybritor platform is shown in figure 31 and Workpartner project information can be found on public website:
http://www.automation.hut.fi/IMSRI/workpartner/

Figure 31 Hybritor platform [HLSYK03]

The similar mobile robot platform is under development in Tallinn University of Technology, Department of Mechatronics. The platform is called UGV (Unmanned Ground Vehicle) for demining purpose. The development process started in 2005 concurrently with the thesis design framework and Mobile Platform Toolkit development process. The platform is equipped with independently actuated legs and wheels. The platform has a modular structure where the modules can be easily removed or replaced when needed. UGV can provide different functionality depending on the equipped module. Standard service module is the manipulator module and additional modules can be NBC detection, surveillance, cutting tool, etc. The vehicle has the following modules and subsystems:

- Energy subsystem
- Locomotion subsystem
- Track subsystem
- Control subsystem
- Vision subsystem
- Manipulator subsystem
- Communication subsystem

The early prototype of UGV design is shown in figure 32.

Figure 32 UGV platform

There are several other mobile robot platforms in the world and they are actively developed for many areas. The most advanced results are in the military section. The well know solutions are iRobot PackBot [iRobot], Foster-Miller Talon [Talon], RHex [SBK01] and others. However there is no unified platform for

new mobile robotics. Most of the solutions are developed from zero and the engineering design data are unavailable for public. The Mobile Platform Toolkit developed in this thesis is trying to offer an open unified platform development solution. The example diagrams based on the real systems are presented in the following as a guideline for using this toolkit.

4.1 Requirements modeling

Requirement engineering starts in most cases when receiving the initial system specification from customer. On the other hand some (robotic) projects can start without external customer and are driven from research interest or new technology interest. This is usually the case in academic institutions or research institutes where the robotic platform is needed to study different robotic modules like control algorithms, navigation algorithms, manipulator subsystem, etc. In this case the requirement list has to be defined before the development process can start. The practical difference between these two cases are usually the requirement flexibility where the research project requirement can be changed more easily during the development than in the case of the industrial application. Although the requirements can change (and usually some change takes always place) the requirement modeling should support to track these changes and point out the affected side systems. Therefore the requirement model must be created with great care.

The initial requirement model is decomposed in Level I according to MPT. On the first cycle the decomposed requirements are opened in separate diagrams and linked with the requirement list by the *ID*.

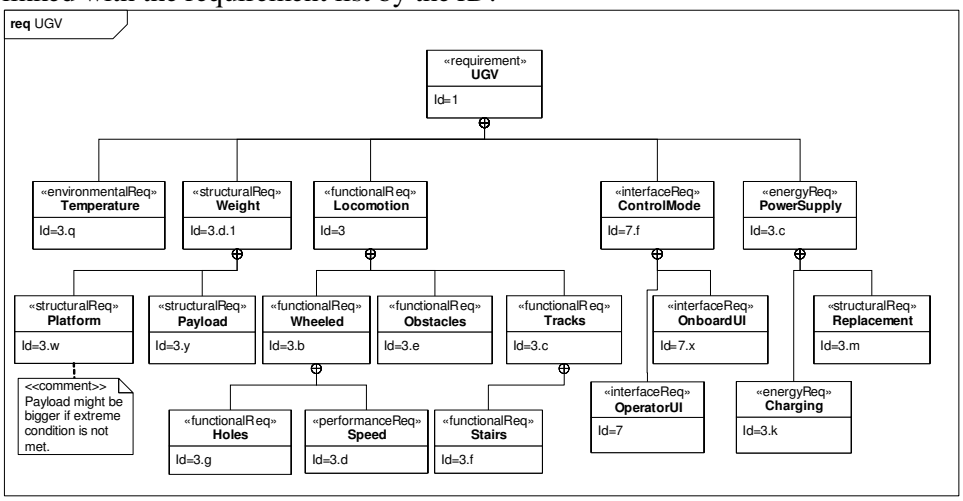

Figure 33 General requirement decomposition of UGV (a fragment)

Defining the general requirement decomposition the *req_general* template diagram from library is used. Suitable requirement elements are selected and imported to the model by binding element parameters i.e. *ID*, with actual values. Appropriate relationships and types are imported automatically. The requirement model is connected with the requirement list by the *ID* parameter. It is possible to include the particular requirement text into the requirement element by the parameter *Text*, but usually the text document is longer than a single sentence and including all text in a model element would make the model less comprehensive. The requirement diagram text explanation defined on figure 33 is shown in figure 34. When needed, the text can include figures, schematics or other graphical elements.

table Requrement list

ID	Name	Text
1	UGV	The UGV is the base platform for demining robot
3	Locomotion	The platform is equipped with two different locomotion types: wheels and track. Both types should be operated separately. The passableness is determined by Bekker's method [Bekker60].
3.b	Wheeled	The platform is equipped with electrical brushless DC motors, enabling turning around on the spot.
3.c	Tracks	The platform should be equipped with removable tracks to improve the off-road capabilities and climbing up stairs.
3.d.1	Weight	Maximum total weight of robot is 300 kg.
3.e	Obstacles	The platform is able to overpass the obstacles with maximum height 40 cm.
3.f	Stairs	The platform is able to climb up and down the stairs with an ascent angle 40 deg with maximum robot weight. The mass center must not change significantly.
3.g	Holes	The platform is able to pass deep holes with maximum width 30 cm.
3.d	Speed	Constant speed, at least 5 km/h, is guaranteed on smooth and parallel slopes (max 20 deg) with maximum robot weight.
3.c	Battery	The platform is equipped with battery enabling to work at least 1 hour in normal environmental condition (outside temp. 24 deg).
3.k	Charging	The charging is assured by 220V/50Hz, 110V/60 Hz and 12/24 V DC main supplies.
3.m	Replacement	The main batteries have to be replaced without interrupting the control and communication module.
3.q	Temperature	The working temperature is -15 deg …. + 50 deg. In working temperature below -10 deg the working time can be reduced 40%.
……	…….	…….

Figure 34 Requirement list

In the requirement engineering stage the context and relationships with the environment have to be specified. Here it is possible to use either the *Block Definition* diagram or the *Use Case* diagram. According to the profile specification the *Use Case* is shown as an example, see figure 35. *Use Case* in SysML language is directly derived from UML 2.0 without any modification. There are quite few elements and therefore it is easy to understand. The system is outlined with the system boundary line and outside the system different so-called actors are presented. The user of the system can be a person, other system or an environmental item. The system itself provides certain services and interacts with external users. The example describes the overall context of the mobile robotic platform - UGV.

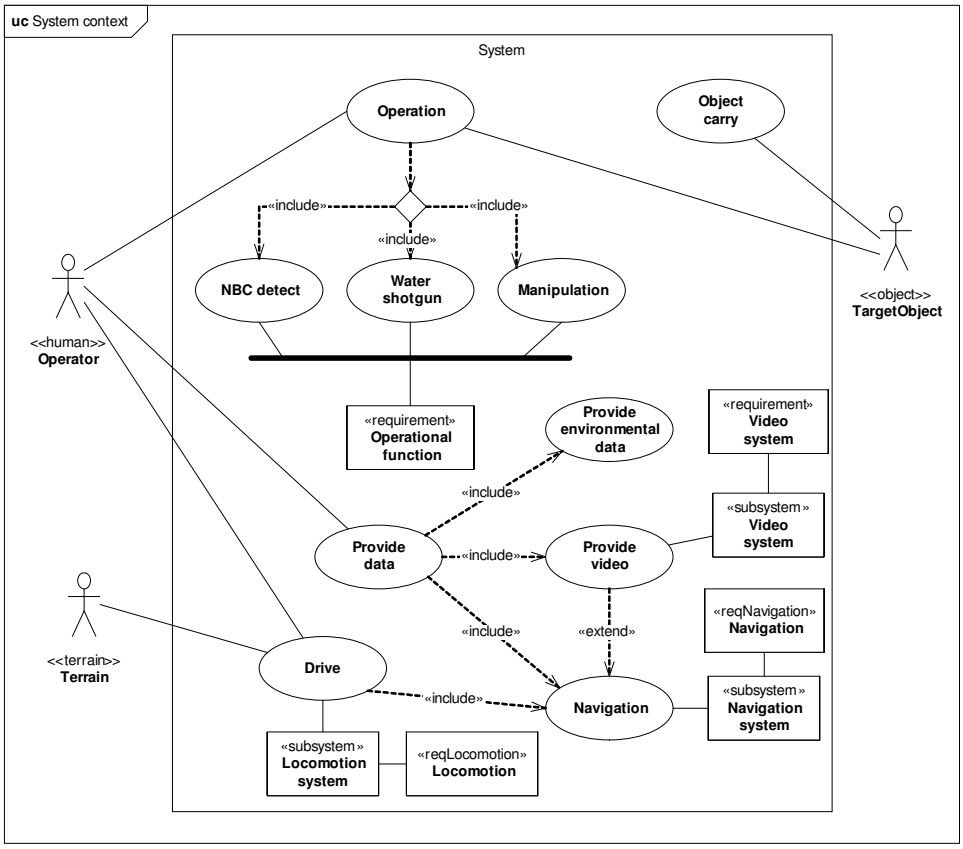

Figure 35 System context

After the first level system context definition and requirement specification, every subsystem requirements have to be opened in more detail. Level II embraces the subsystem details as described in level description in chapter 3.3.1. The following diagram shows a subsystem requirement model at level II. The

subsystem *Energy* is decomposed and parameterized with toolkit parameters. Generic design elements are connected with the requirement element to show the satisfaction and verification relationship of a particular element. The example is based on Workpartner case where two different energy sources are used. The energy system includes a lightweight compulsion engine and series connected batteries. In addition the computer control is included to the energy system to obtain maximum efficiency. Control software starts and stops the compulsion engine to recharge the batteries according to efficiency [Ylönen06]. The energy requirement model is shown in figure 36.

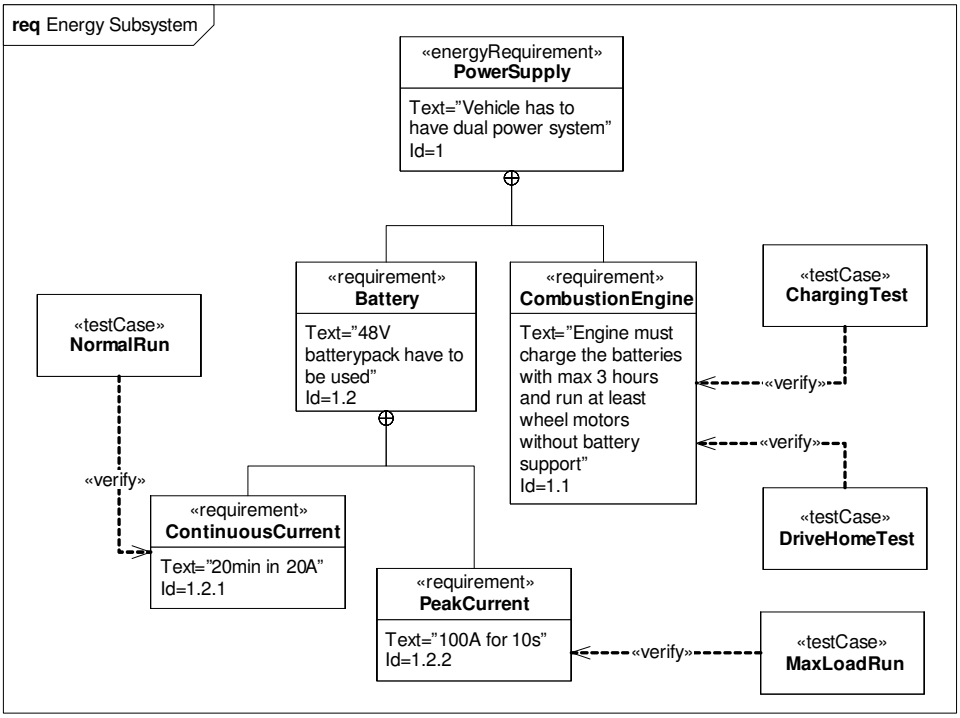

Figure 36 Requirement model for Power subsystem

The outcome of the requirement engineering cycle is a set of models, describing the requirement structure and needed design elements to satisfy or verify the particular requirement. The model may be refined during the concept generation stage in case new aspects arise. However the changes must be documented and carefully dealt with. When changing the requirement all connected elements (design blocks and activities) have to be rechecked as well to assure the consistency of the whole system. In case of software support for toolkit the conflicting relationships can be detected automatically. The rule check algorithm validates the design model and requirement model according to profile specification.

4.2 Static structure and component interaction

The component decomposition and relationship definition is the first actual system modeling step. Some of the components can be directly derived from the requirement model, whereas some components or subsystems will join to one and some diverge. The general subsystem decomposing is similar to general requirement model. The next step is to decompose the subsystem into functional blocks. In figure 37 the energy supply subsystem of Workpartner is shown.

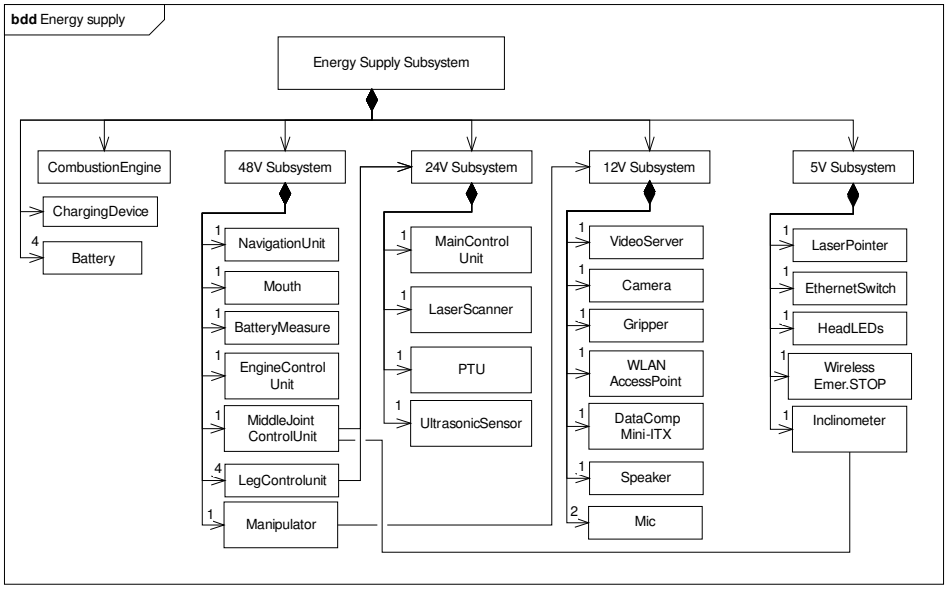

Figure 37 Energy supply subsystem of a Workpartner

Here only the name of the block is represented and dealt as a black box approach. Some blocks are generally one-to-one understood like *Battery*, some can be realized in very different way, e.g. *Gripper*, *ControlUnit*, *Manipulator*, etc. The actual design and parametric model of a particular block will be developed in the next design stage. While the *bdd* diagram describes the component hierarchy, the *ibd* diagram defines the component interaction. Ports and flows are defined enabling the interaction between the components across the system.

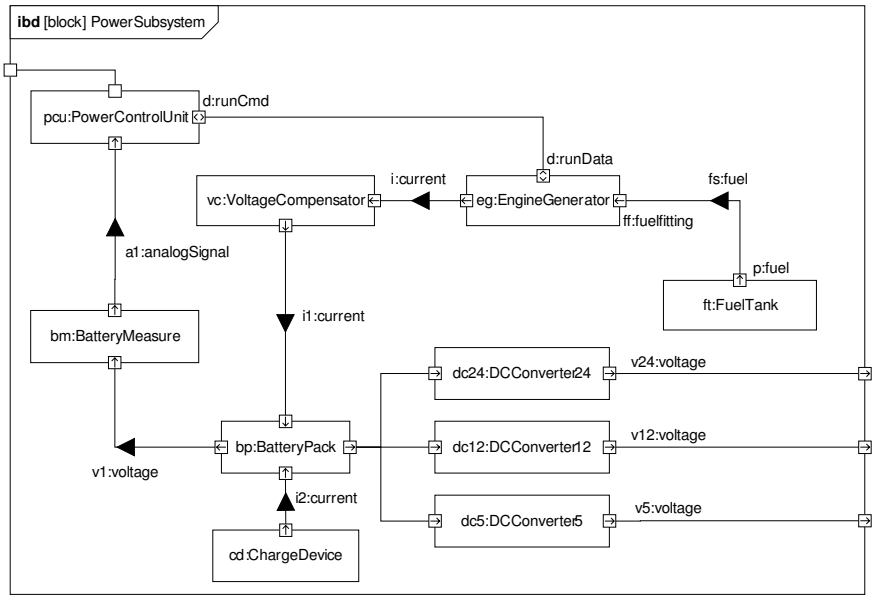

Figure 38 Power subsystem interaction

Here the dual power system is described (figure 38). Components have different port types, i.e. *AtomicPort*, *non-AtomicPort* and flow directions. *AtomicPorts* are single usage of a block while the *non-AtomicPort* is a multiple usage of a port. In case of *non-AtomicPort* the flow specification must be specified. The previous example corresponds to the Workpartner analytical diagram. The power subsystem involves the main components *EngineGenerator*, *BatteryPack* and *PowerControlUnit*. Other outlined components are converters, sensors and accessories. Components are connected through the ports that can be in this case of two different types. For example, a fuel flowing from *FuelTank* to *EngineGenerator*. Both blocks have an atomic port *p* and *ff* respectively. Another possible port type is non-atomic port. A non-atomic flow port relays items of several types specified by a *FlowSpecification*. In this example a port *d* of *PowerControlUnit* and *EngineGenerator* is a non-atomic port. The specification of this is shown in figure 39.

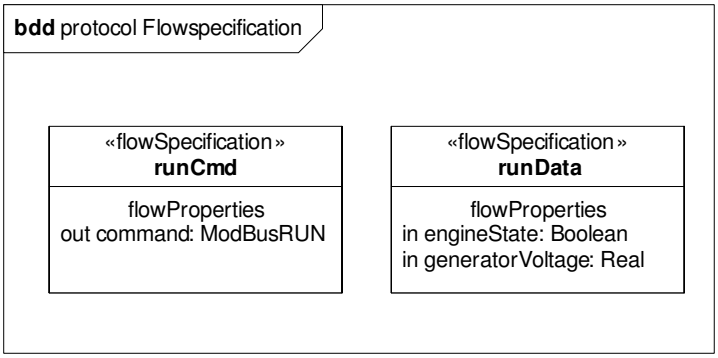

Figure 39 Flow specification for a non-atomic port

All subsystems are modeled in a similar way to the presented power subsystem model. Later on the integrity of a component model is examined and different models are cross-mapped using the *Allocation* diagram. Allocations are often used in early design as a precursor to a more detailed rigorous specification and implementation.

4.3 Behavior modeling

Behavior modeling describes the behavior of the system through the *Activity*, *State Machine*, *Sequence* and *Use Case* diagrams. With activity we can specify the actions and their input/output sequences. Continuous and discrete flows, such as material, energy or information flows can be specified by *Rate* stereotype.
The general behavior and system services are described with the *Use Case* diagram. The *Use Case* is a simple and trivial representation of system usages and users. Nevertheless it is necessary to represent visually the main services of a system and connections between them. The simple and human friendly representation is useful when explaining the further system to the non-technical persons. They can be different stakeholders like customer, market-analyzers, financers, etc. Presenting the system overview and context in this way enables us to ensure that developers and non-technical stakeholders understand the design problem and proposed concept in the same way. The templates *uc_context* in the early stage and *uc_system* in behavior design can be used for this purpose. Figure 40 shows the main services of a system and its boundary as well as external “actors”.

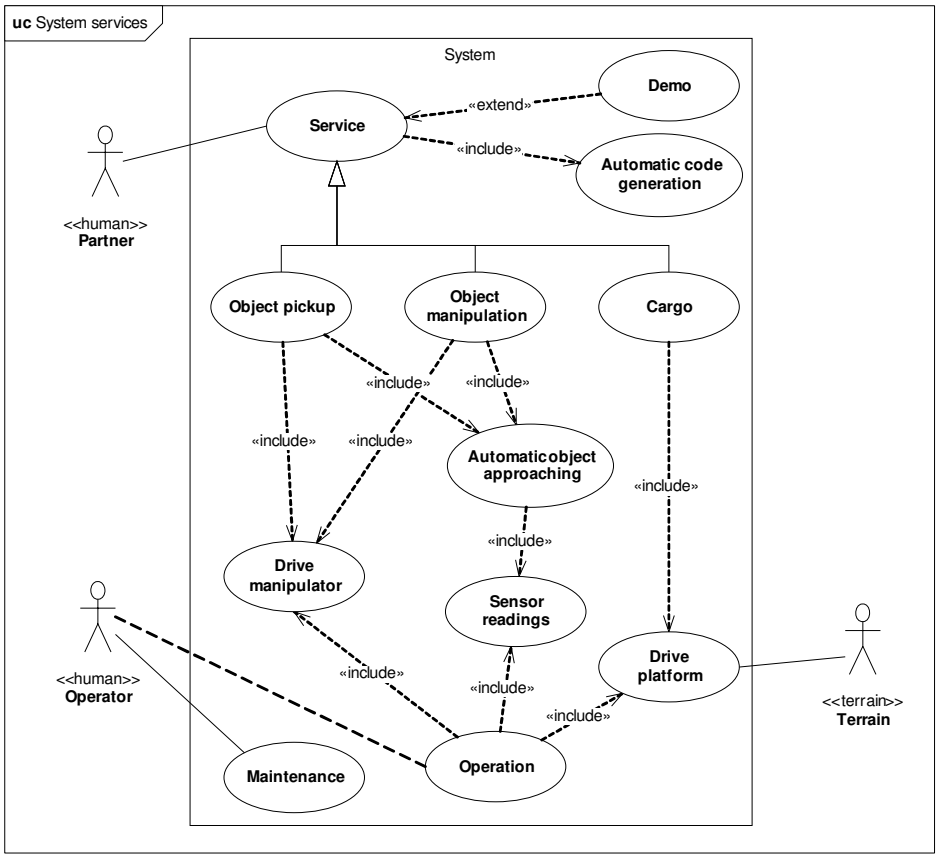

Figure 40 General services of the system

System activity is more specific and models the system or subsystem actions. The common activity of the system is specified with template *act_general s*howing the high level action sequence. According to SysML specification, there are several types of elements, like *Action, Events, Signal, TimeEvent* and so on. The Toolkit specifies additional stereotypes described in the previous chapter. Based on this model the main process will be implemented either in the hardware or in the software. Operational inputs as well as main sensor signal are connected to the certain action. The following figure is a general activity diagram based on the Workpartner mobile robot.

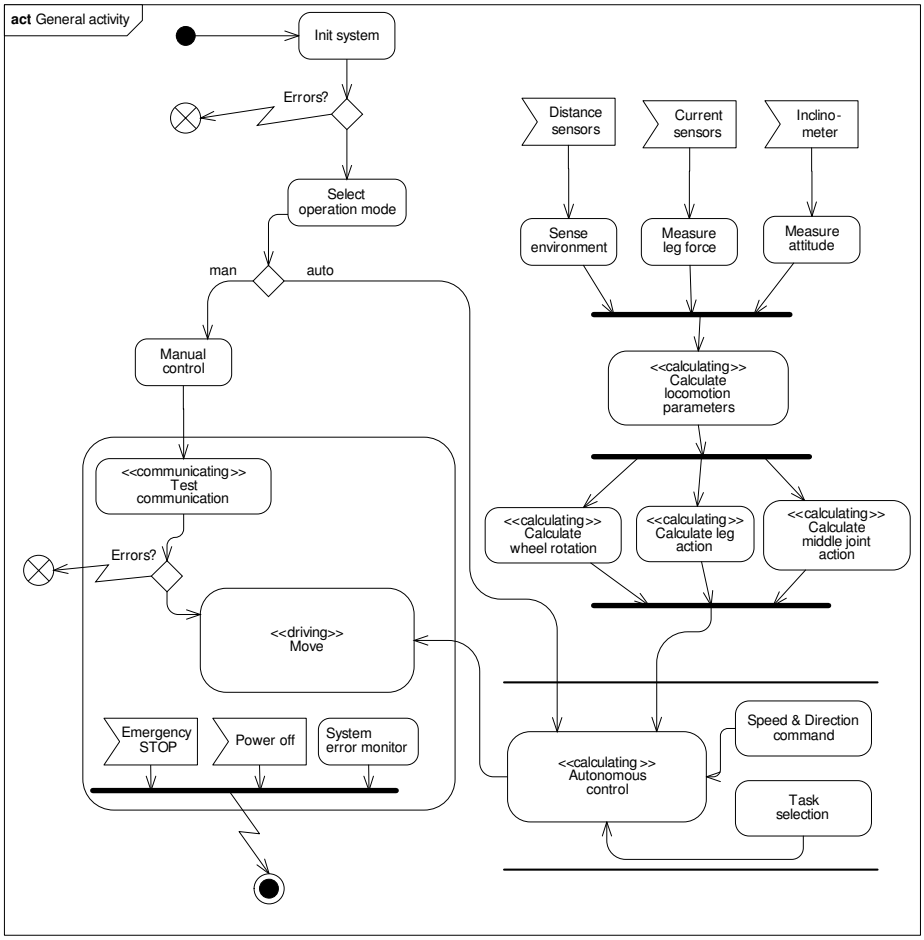

Figure 41 General activity diagram of Workpartner

Implementation of this activity is described as follows:
When the system is turned on, the initial selftest action is carried out. All subsystems will run appropriate tests shown here as one black box activity. With the tests done, the next action follows. The system has two different operation modes and selection will be made by the operator. If the operator selects remote teleoperation guidance the additional communication test will be executed. If no error results the general black box activity *Move* is reached. The *Move* activity holds in this stage all operational and functional activities of the platform. In addition to commands from the operator, the actual measurements are performed. Sensors measurements are inputs for control logic to calculate the locomotion parameters in order to select a right locomotion type and save the energy. The whole system operation will be interrupted when either the emergency stop is pressed or a fatal system error occurs during operation. The

normal shutdown is realized by power off signal. As it is seen, the actual activity of the subsystem is not represented in this diagram. The *act_general* template is used for describing the system behavioral concept. The next logical step is to open basic functions and describe the certain functionality of subsystems. For example the Workpartner has three locomotion types: ordinary rolling, walking and rolking. Rolking is a special hybrid locomotion type combining the rolling and walking on the same time where leg joints and wheels are generating the propulsive force simultaneously [HLSY00]. The next diagram opens one subsystem and shows the activity model of rolking locomotion mode.

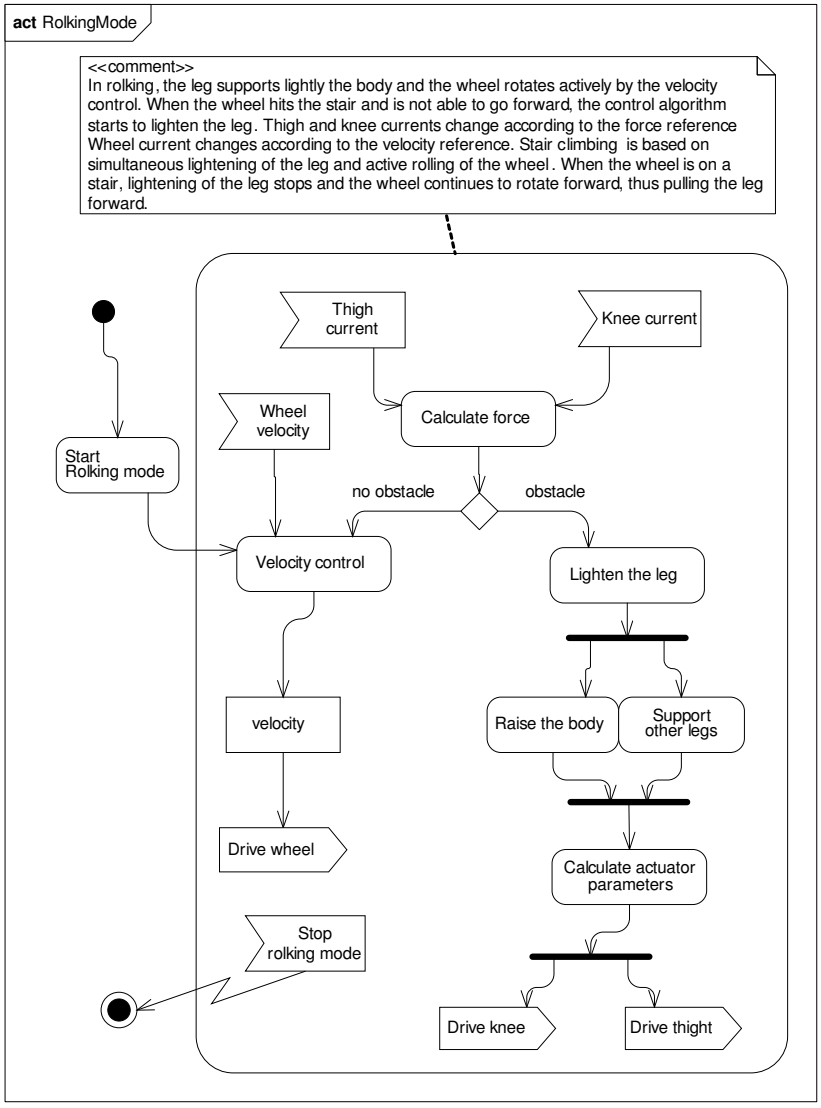

Figure 42 Activity diagram of rolking mode

Activity represented in this diagram is implemented on the Workpartner robot. Workpartner has a special feature that enables us to select the locomotion type automatically. The selection bases on the energy efficiency and terrain estimation, which can be pre-simulated using terrain library and the corresponding simulation model. The goal is to minimize the energy consumption without loosing considerably the speed and performance. Here the energy consumption is a key parameter which will be optimized. The automatic locomotion selection is modeled in state machine diagram. The state machine represents the system discrete behavior and through finite state transitions. The activities are invoked during the transitions which are entry and exit of state. The execution of activities is specified by guard conditions. The transitions between the states can be continuous or discrete. In addition the composite state can be used to nest the states in sequence or parallel action. Using the *State Machine* and *Activity* diagram for behavior modeling is not very strictly defined. The discussion about this is found in chapter 2. Having considered these different approaches discussed in chapter 2.2 the Toolkit specifies his own approach based on previous ones. In general the *State Machine* is treated as a higher level system states. Every state is specified with the specific *Activity* diagram describing the inner structure of a given state. The concept is similar to hybrid system modeling where the discrete states consist of continuous process inside. Nevertheless depending on the characteristics of a particular system and problem, opposite approach can be successfully applied (as discussed in chapter 2.2). The example given in figure 42 is a decomposed action with guard condition represented by *State Machine*. The further decomposition can be implemented again with *Activity* diagrams where every state has its own diagram. The reason of dictating the diagram usage is to give a concrete guideline for engineers who want to have practical guides but not many open options. However the previously described approaches are not precluded and can be successfully used by experienced engineers. The corresponding *State Machine* diagram is shown in figure 43. This is the detail state model describing the behavior when the autonomous regime is selected.

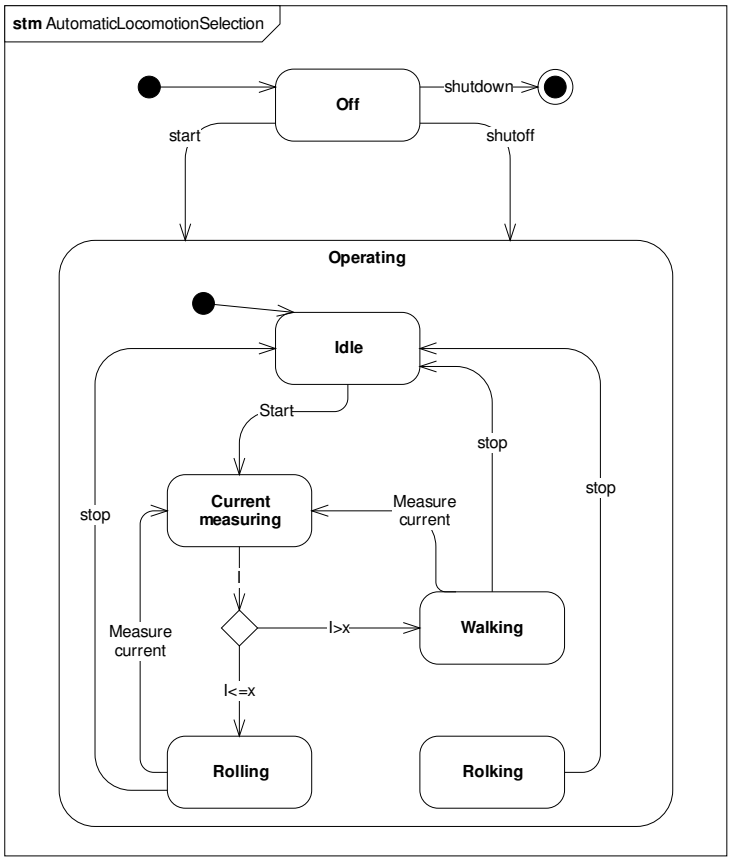

Figure 43 State machine diagram

In addition to *State Machine* and *Activity*, the behavior can be modeled with the *Sequence* diagram. This type of diagram is often used in software modeling and suits well for embedded control algorithm modeling in a robotic platform. A sequence diagram describes the flow and control between actors and system blocks. Sending and receiving messages between lifelines are represented according to timeline shown on the vertical axis. The *Sequence* diagram, shown in figure 44 enables modeling of timing and component allocation. In figure 44 the automatic charging system is modeled. Based on this model power control algorithm can be later analyzed and implemented.

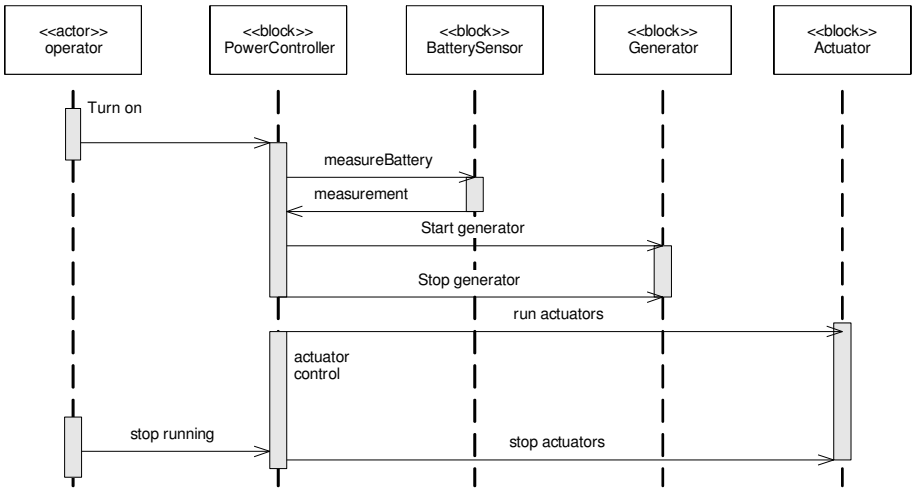

Figure 44 Sequence diagram

The described model is used to represent the behavior of general system and its subsystems in a different level and situations. In conceptual design several parallel models might be developed for a single situation. Allocating the activities with a system static structure (i.e. blocks) pros and cons of particular model can be easily determined and analyzed. On the next design stage behavior models are used to generate process control algorithms either in hardware or software. When appropriate tools are available some software creation process in controller design can be automated. The high level behavior diagrams are more general and cannot be directly used for automated design. However they are important to understand the system behavior and assure the fulfillment of the requirements.

4.4 Parametric modeling

The real-world systems are in most cases non-linear and therefore it is important to have tools for early stage simulations where even the system dynamic model is not yet fully defined. The early stage system parameter relations are modeled by the *Parametric* diagram. It shows how one value of the structural property affects the other value. Parametric constraints are tightly connected with the system structure and are used in combination with *Block* diagrams. Two new stereotypes are introduced by the MBT which determine the key parameters of a system. The key parameters are the main input for system analysis and simulation. Different conceptual solutions are described by *Parametric* diagram and will be executed in simulation environment. The simulation results are used for design improvement. For example, this is used for performance and

reliability analysis as well as for meeting all requirements specified by the *Requirement* diagram.
The *Parametric* diagram of UGV performance is taken as an example.

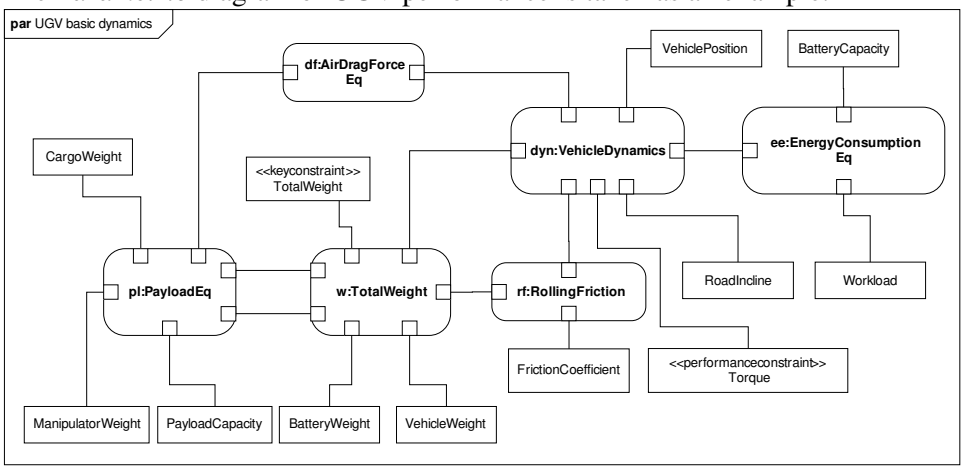

Figure 45 Parametric diagram of UGV

In figure 45 it is seen that *TotalWeight* is denoted as a *keyconstraint*. This indicates that the subsystem should be optimized according to this parameter. The total weight of UGV is derived from requirements and affects the overall vehicle performance as well as many other parameters. From the performance point of view the output torque is another key parameter which needs to meet the appropriate requirement. The optimum relation between the needed torque and maximum allowed weight is searched without compromising the required functionality. The mathematical relations between parameters are denoted by the rounded rectangle with defined ports. Parameters and links with other functions are connected through these ports. In this example the mathematical equations are not shown but can be described by a comment object when needed. An example of this mechanism is shown in figure 47.

4.5 Simulation modeling

Simulation is usually exploited at a later design stage where the system model is relatively precisely defined. To get the maximum benefit, the proposed design framework includes simulation in the conceptual design stage. Simulation principles are generally defined in chapter 3. According to these principles and toolkit extension described in chapter 3.3.4 the following two options are presented:
The model (structure and behavior) consists of a special block element stereotyped as *simu*. An example is shown in figure 46 where *simu* block is a control algorithm of robot, controlling the leg and wheel motors according to

terrain changes. The *ControlFPGA* block is a link to the simulation model. Simulating the control algorithm, the engineering team gets feedback on critical component parameters required to fulfill the initial requirements, or simulating different algorithm candidates determining the system feedback.

The following example shows the *ibd* containing *simu* element.

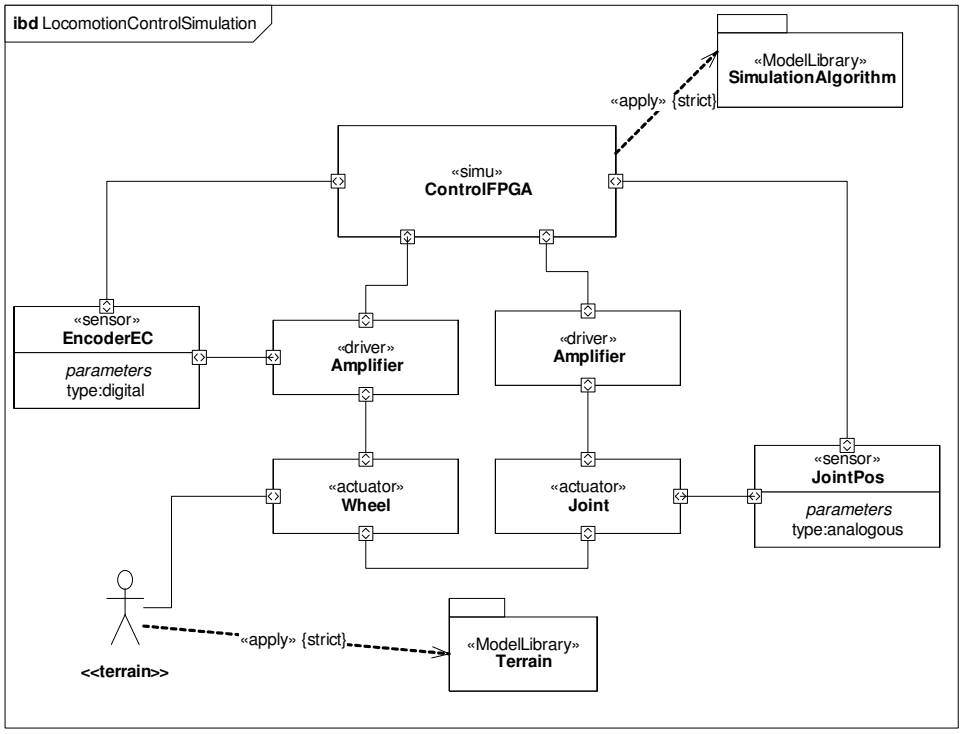

Figure 46 Locomotion Control including simulation blocks

Another simulation link defined by MPT is a parametric diagram connection to Simulink model. The parametric diagram defines the mathematics and constraints of a subsystem. Based on this model Simulink block diagram is generated. Later on the same a Simulink model can be linked to a component in *bdd* model as described above. An example of corresponding parametric model is shown in figure 47.

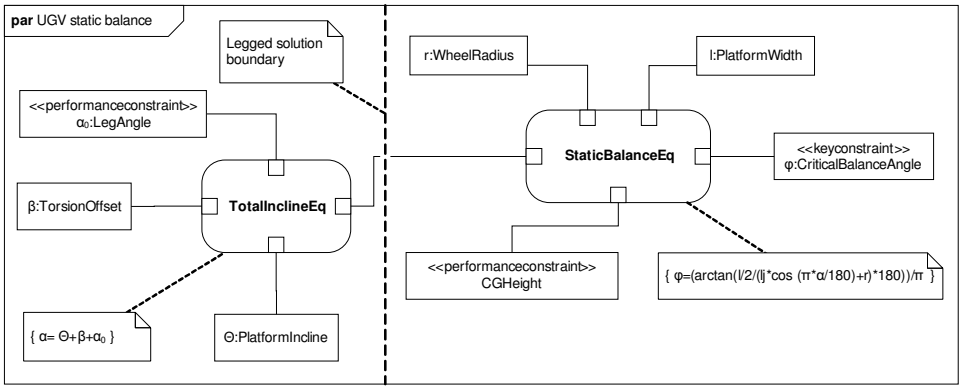

Figure 47 Balance condition in incline situation of platform

The diagram shown in figure 47 is a source for Simulink model shown in figure 49.
The parametric model defines the balance conditions of a mobile platform. Here two concurrent design candidates are compared. The principal concepts are shown in figure 48. A design candidate is linked with the simulation model and simulation results are compared with appropriate requirement. Simulation is executed in various terrain conditions.

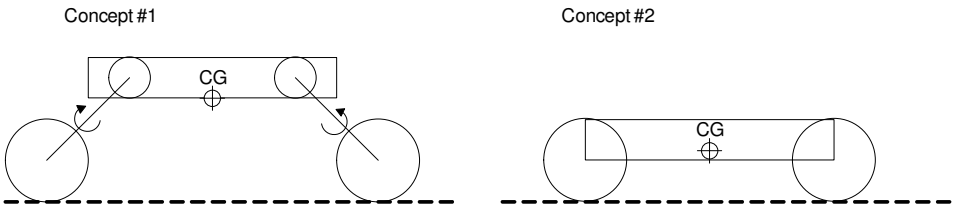

Figure 48 Two conceptual design candidates

The following static balance condition formula is used for a legged platform (design candidate #1):

$$\varphi = \frac{\arctan\left(\frac{l \cdot 0{,}5}{l_j \cdot \cos\left(\frac{\alpha \cdot \pi}{180}\right) + r}\right) \cdot 180}{\pi} \qquad (4.1)$$

where:

α – the controlled leg angle

l – platform width
l_l – leg length
r – wheel radius

Balance condition for a conventional platform (design candidate #2) is described with following formula:

$$\varphi = \frac{\arctan\left(\frac{l \cdot 0{,}5}{l_i + r}\right) \cdot 180}{\pi} \qquad (4.2)$$

where:

l_i – wheel axe shift

The Simulink model has a S-function which is generated from *Parametric* diagram shown in figure 47. Different design candidates can share the same S-function or have the specific S-function depending on the solution configuration. Terrain and simulation parameter inputs are the same in single simulation enabling comparable results generation.

These simple examples are for description purpose. Contact properties, dynamical forces and movement parameters are excluded due to the space limitation.

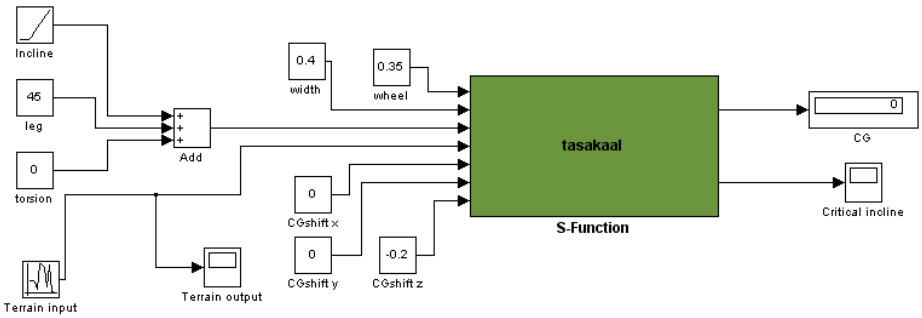

Figure 49 Balance simulation model

The example shown in figure 49 has two structural concepts of the mobile platform simulated against incline requirement. The requirement specifies critical balance angle which is the key constraint of this simulation.

The solution candidate selection is carried out by comparing the different simulation results and linking the results with the requirement model. Traditional

analytical and evaluation methods can be utilized to select the best machining solution.

The above examples show how to implement the initial simulations in the conceptual design procedure. Compared with the detail design stage, the conceptual design has much bigger design freedom but also lacks precisely defined system parameters. This characteristic is the same in simulations which give big freedom to implement a simulation model but lack the simulation parameters at the same time. It might be necessary to estimate some unknown parameters in initial simulations. Nevertheless the main target is to compare the design candidates and see the difference in certain conditions rather than to get the precise behavior of a further system. This aim enables us to unify some common simulation parameters which were not known the beginning.

Solution candidate selection is usually affected by many aspects. The selected solution should best match the requirement criteria as well as meet other relevant aspects. The simulation result is definitely one of the essential factors and these results can be interpreted by the engineering team to support the selection decision or improve the concept.

4.6 Conclusion

1. In this chapter the implementation process based on two real examples is described. The design methodology framework has been implemented in real design and two different design cases are involved to illustrate the analysis and synthesis of a mechatronic system. The implementation is a practical example of using Mobile Platform Toolkit developed in the frame of this thesis. The toolkit definition has been explained in the previous chapter in detail.
2. The chapter has a logical subchapter sequence corresponding to design process described in chapter 3. To illustrate several aspects of conceptual modeling with a developed toolkit real system examples are presented. The examples are analytical model fragments of the mobile platform Workpartner, developed in Automation and System Technology Laboratory of Helsinki University of Technology, and design model fragments of Unmanned Ground Vehicle – UGV, currently under the development in Department of Mechatronics Tallinn University of Technology. The covered early design stages are requirement engineering, model development and initial simulation. Diagrams are often compiled for power or locomotion subsystems as these are the most important in mobile platform design. Vehicle dynamics and control algorithms are partly covered in the simulation behavior sections.

3. In addition the chapter is intended for use as a design guideline of using developed Toolkit. The practical examples are covering the toolkit specific stereotypes as well as templates from the template library. The templates are intended to simplify technical issues of developing models. Pre-defined templates are developed based on the existing systems and include most common subsystems and parts. In case of using templates it helps to keep track of all the essential parts of the platform design without forgetting any important aspects. At the same time the creativity and novelty of developing a new solution is not prevented as templates are open and only relevant parts of the particular template have to be imported to the design. Nevertheless the connections and other relations are imported automatically without losing the consistency of the system.
4. Although two real cases are described the full development process is hardly covered. In a real design process all subsystems must be fully modeled and more than one design candidate developed and simulated. Including all diagrams and design documents into thesis is not rational due to the huge amount of space and detailed technical documentation of the particular system.

5 CONCLUSIONS AND FUTURE WORK

For conclusions the following aspects are considered: novel findings and developments of the present work, practical implementation and application example, further development of the research. Two main aims of this research are to develop and propose a base framework for conceptual design of a mechatronic system; utilizing recent developments and techniques to develop a practical tool for advanced conceptual design.

The primary theoretical result is a model based mechatronics system development methodology framework for conceptual design stage.

The following conclusions are reached and novel aspects introduced in the thesis

1. Through the analysis of different product design approaches the adopted general design sequence is composed. In different sources the design stages of product development process are described conformably, varying mostly in names and boundaries. The composed design stage tree is a unified approach from analyzed solutions with the mechatronics specific additions. The integration stage has been separated from general system design and a stronger emphasis has been placed on requirement and concept connection. This modified design process reflects better the mechatronics design nature and emphasizes the early stage.

2. A main result – mechatronics conceptual design framework denoted as Conceptual Framework Model (CFM) is the base ideology of this thesis. The framework relies on the VDI2206 V-macromodel methodology whereof the concept is adopted for conceptual design. New sub-stages are defined and linked with each other for verification and refinement of a model. A newly developed System Modeling Language has been selected as a modeling tool which is expanded by the application specific profile. The concept features a semi-autonomous design solution generation, template libraries and initial simulation model. In the state-of-art situation analysis relevant research works of design process supported by artificial intelligence methods are investigated. The advantages and disadvantages of the techniques are shown and based on that genetic programming concept for the semi-automated design solution generation is selected. The selected technology is utilized in this framework as a practical tool for the concept generation automation.

3. Based on the proposed framework a practical tool derived from the System Modeling Language is developed as a language profile. This is done by using profile inheritance mechanism built in to SysML and UML. Application toolkit development principles are introduced and

design pattern libraries proposed. The design patterns are denoted in this thesis as design template and are divided into several categories. It is shown that using pre-developed design templates can improve the conceptual design significantly. As an application example the mobile robotic platform domain is selected for implementation of the developed methodology. A partial reason for this selection is recent developments in robotic industry and corresponding international network establishments in the last years. The practical need for a systematic approach to the mobile robot design in TUT where several medium-size mobile robotic projects have been introduced has been an important factor.

The main practical result of the present work is the Mobile Platform Toolkit (MPT) for robotic application design. The toolkit has been implemented and corresponding libraries are developed which are not fully included in the thesis due to the space limitation. Some practical examples are presented in the implementation chapter.

The following outcomes based on the practical results can be formulated:

1. The main practical result of the thesis is a Mobile Platform Toolkit (MPT) based on the theoretical research. The MPT is an application specific toolkit for the conceptual design stage. The toolkit includes the requirement modeling as well as structure and behavior modeling instruments. The toolkit is derived from SysML and is fully applicable in SysML compatible software environments, hence it can be implemented without creating a specific software environment.

2. A practical implementation of the developed methodology is summarized in short and an explicit guideline is presented. The guideline describes the conceptual design sequence according to the developed methodology and explains the available option in a current stage.

3. The implementation process has two parallel applications at different design stages. Some examples are modeled based on the existing system showing the connections between the real system and model diagrams. The selected system is Hybritor platform of Workpartner project developed in Helsinki University of Technology, Department of Automation and System Technology. The examples show the subsystem structure and behavior models. The second application is a similar robotic platform which is under development in the Department of Mechatronics, Tallinn University of Technology. During the work several concepts are tested in the real design process. The requirement analysis is one of the examples described in the implementation chapter.

The whole thesis is organized as a design background analysis, theoretical implementation of a novel conceptual design and a practical application for specific domain compounding the whole concept into one consistent unit. The thesis and research is supported by continuous presentations at international mechatronics conferences.

The author has published more than 15 pre-reviewed international publications on subjects related to this thesis. Some of them are indexed in the international databases, including ISI Web of Proceedings.

The work results are reported during the thesis compilation in several series of conferences: REM (Research and Education on Mechatronics), DAAAM (Danube Adrian Association for Automation & Manufacture), EPE-PEMC (European Power Electronics - Power Electronics and Motion Control), ICOM (International Conference of Mechatronics) and OST (Oulu-Stockholm-Tallinn). Pending conference presentation is IEEE/ASME - International Conference on Advanced Intelligent Mechatronics in ETH Zürich, Switzerland.

The developed conceptual design methodology is partly exploited in the Unmanned Ground Vehicle development process in the Department of Mechatronics, Tallinn University of Technology.

During the conceptual framework development for a mechatronic system, the dynamic wheg (hybrid wheel-leg) was invented with the support of the developed methodology. The invention is a unique application for mobile robot platform. Patent application has been composed and registered in Estonian Patent Office with registration number P200700027 on 01.06.2007.

Finally the research goals of the thesis have been completed successfully. The novelty and actuality have been continuously presented and discussed in oral presentations at international conferences as well as in pre-reviewed publications.

Future research is an important aspect and has to consider the following aspects:

Semi-automated algorithm improvement and integration with SysML concept
The algorithm proposed in chapter 3, developed by the GARAGe group has been tested in a lab environment and with non-complex problems. In real design the improvement of this algorithm is needed. However, the concurrent method may be introduced for semi-automated conceptual solution generation based on the artificial intelligence method. A joint continuous research is planned in conceptual design automation already.

Integration of conceptual design approaches
Several researches are currently active in universities around Europe. The integration of the approaches is important to have bigger coverage of application domains as well as techniques and interfaces between them.

Software development for integrated design environment
Today's product design can not be imagined without the support of software. Many platforms and packages are available for several years already but some are very recent. Developed conceptual design framework must be integrated with design environments by creating application toolboxes as described in chapter 3.3. On the other, hand simulation software integration is also needed to simulate created concepts. The main target simulation software would be Matlab/Simulink, LabView and Dymola. The important issue is also to specify in more detail the model exchange interfaces. The suggested standards here are XML/XMI and AP233 specifications.

Library enhancement and new application specific toolkit development
In this thesis an application example described in more detail is a mobile robotic platform. Based on this example many other toolkits should be developed according to the need. It is suggested that wherever possible the toolkits would be freely available for download. A central database may be created for design templates and toolkits.

ABSTRACT

The aim of the doctoral thesis is to develop a mechatronics design methodology focused on the early design stage. The motivation is a drastic growth of the mechatronic and robotic sector and practical needs in Tallinn University of Technology as well as other technical institutions.

The practical need is directly related with robotic application. Robotics is one of the key technologies of the near future. Robots have been available for a long period already but only in specific environments and factories. In very recent years several authorities in robotic sector have discussed robotic revolution especially in service and mobile robotics. First examples can be seen already where home robotic applications have grown very popular. The biggest mobile robotics evolution is taking place in the military and rescue sector. Many new projects are running and several unmanned ground vehicle type mobile robots have been brought to the market.

The drastic growth of robotics sets new demands for the development process. Design reuse, modularity and effective development process are the main issues to be successful in this business. Therefore it is necessary to find new solutions for effective product design. The robots are true mechatronics products and this leads us to search mechatronics development methodologies. In conventional engineering there are lots of different methods and tools but not in mechatronics field where only some recent works can be found. Also, a successful product development means that the design solution has to be selected very carefully, based on a strong research in the very beginning of the design. Strong methodical focus on the conceptual design and requirement engineering can save the design cost and time significally. Based on these facts new mechatronics design methodology is clearly needed and robotics is a well suited application. However the methodology and tools have to be clear and easy to learn.

This thesis is focused on the model based design methodology for mechatronic systems in the conceptual stage. It proposes a generic framework model for effective and semi-automated design candidate generation in the conceptual design stage. The specific application is focused on the mobile robots. An application specific toolkit was developed as a tool based on the developed methodology. The methodology specifies toolkits and model libraries which are realized on the example of mobile robotic platform design. Additional toolkits can be developed according to this methodology.

The practical implementation of the Mobile Platform Toolkit is presented based on real examples. Two analogous mobile robots in a different design stage were selected to illustrate the use of the toolkit. The Workpartner robot from Helsinki University of Technology is used as an analytical example and the UGV

platform from Tallinn University of Technology is utilized as a synthesis example where the developed approach was applied in the conceptual design stage.

This thesis proposes a methodological approach for a mechatronic system design in the conceptual stage. Practical results are the Mobile Platform Toolkit and application examples with brief guidelines for technology implementation.

Keywords: Conceptual design, mechatronics, mobile robotics, modeling, simulation

KOKKUVÕTE

Doktoritöö raames on töötatud välja mehhatroonikasüsteemi kontseptuaalse faasi modelleerimise metoodika, mis baseerub VDI2206 modifitseeritud projekteerimise V-mudelil ja uuel arendamisjärgus oleval süsteemi modelleerimise keelel SysML. Doktoritöös on pakutud välja kontseptuaalne raamtöö mudel, mis käsitleb disainlahenduste poolautomaatset genereerimist. Mudel sisaldab endas ka rakendusspetsiifilist töövahendite komplekti (*toolkit*), mis on praktiline tööriist projekteerijale. Praktiliseks näiteks on valitud mobiilne robotplatvorm. Valiku määras mobiilsete robotite kiire areng ja osakaalu järsk suurenemine tehnoloogiarakendustes viimastel aastatel ja lisaks ka praktiline vajadus Tallinna Tehnikaülikoolis.

Doktoritöö eesmärgiks oli luua mehhatroonikasüsteemi projekteerimise kontseptuaalse faasi metoodika ja praktiline töövahend. Otsene vajadus selle järele oli tingitud mehhatroonika ja robootika sektori kiirest arengust ning praktilisest vajadusest nii Tallinna Tehnikaülikoolis kui ka mujal.

Käesoleva töö konkreetsed eesmärgid olid:

1. Analüüsida olemasolevaid toote projekteerimise metoodikaid ning nende sobivust mehhatroonika valdkonnaga.

2. Töötada välja üldine mehhatroonikasüsteemidele orienteeritud kontseptuaalse projekteerimise raamistik ja mudel. Mudeli fookus on varajane projekteerimine, mis hõlmab nõuete analüüsi ja modelleerimist ning tooteloome kontseptuaalset projekteerimist ja modelleerimist.

3. Arendada välja konkreetsed töövahendid raammudeli rakendamiseks mehhatroonikasüsteemi projekteerimisel. Töövahendite komplekt peab katma nii nõuete kui ka kontseptuaalse faasi modelleerimise probleemistiku. Töövahendite komplekt on rakendusspetsiifiline ja näiterakenduseks valida robootika.

4. Rakendada väljatöötatud mudel ja töövahendite komplekt konkreetsel robootika rakendusvaldkonnal. Koostada läbi praktiliste näidete lühike metoodika kasutusjuhend.

Doktoritöö koosneb neljast põhiosast. Kaks esimest on projekteerimis-metoodikate ülevaade ja võrdlus ning olulisemate uurimisprojektide analüüs, kus on kasutatud tehisintellekti meetodeid kontseptuaalse disainlahenduse saamiseks. Kolmandas peatükis on kirjeldatud projekteerimise metoodikat koos töövahendite ja mudelite baasi kirjeldusega. Neljas peatükk on konkreetne rakendusnäide väljatöötatud töövahendite (*toolkit*) rakendamisest kahel erineval

reaalsel projektil. Esimene projekt on Helsingi Tehnikaülikooli poolt välja töötatud Workpartneri teenindusrobot, mida on kasutatud kui analüütilist lahendust. Teine projekt on Tallinna Tehnikaülikoolis arendatav demineerimisrobot, mille kontseptuaalses projekteerimise faasis on väljatöötatud metoodika ja töövahendid osaliselt rakendatud.

Töö tulemuseks on väljatöötatud metoodika koos selle kirjelduse ja rakendusnäitega ning metoodikal baseeruv mobiilse platvormi arendusvahendite tööriistakomplekt konkreetse rakendusnäitena. Lisaks on väljatöötatud metoodika ja töövahendid rakendatud ka konkreetsetele projektidel, mille mõned näited on töös esitatud.

Väitekirja kokkuvõttes on antud suunitlused ja soovitused metoodika edasiseks arendamiseks ja integratsiooniks samalaadsete uurimisprojektidega.

Tulemused:

1. Lähtuvalt erinevate olemasolevate tooteloome protsesside metoodikatest ja lähenemistest on loodud mehhatroonikasüsteemidele kohandatud tooteloome protsessi kirjeldus.

2. Peamiseks töö teoreetiliseks tulemuseks on mehhatrooniksüsteemidele orienteeritud kontseptuaalse projekteerimise raammudel, mis on töö baasideoloogiaks.

3. Töötati välja töövahendite komplekti metamudel raammudeli rakendamiseks ja kontseptuaalse faasi praktiliseks projekteerimiseks. Töövahendite komplekt koosneb projekteerimise eeldefineeritud mudelitest, simuleerimise eeldefineeritud algoritmidest ja modelleerimise baasdiagrammidest.

4. Peamiseks praktiliseks töö tulemuseks on rakendusepõhine töövahendite komplekti väljatöötlus. Rakenduseks valiti mobiilne robotplatvorm, kuna antud valdkond on viimaste trendide järgi muutumas väga oluliseks nii tsiviil- kui ka militaarrakendustes.

5. Vastavalt töövahendite komplektile teostati praktiline realisatsioon, mis baseerub kahel reaalsel mobiilse roboti projektil.

Uurimistöö ajal on väljatöötatud metoodikat reaalselt rakendatud kahel projektil ja lisaks on arendustöö käigus loodud leiutis – ratasjalg, mille taotlus on registreeritud Eesti Patendiametis numbriga P200700027.

REFERENCES

[AN98] P.J. Antsaklis, A. Nerode, Hybrid Control Systems: An Introductory Discussion to the Special Issue, *IEEE Trans. On Automatic Control*, 43, 1998, pp. 457-460.

[Balmelli06] L.Balmelli, An Owerview of the System Modeling Language for product and systems development – Part 1: Requirements, use-case, and test-case modeling, Watson Research Center and Tokyo Research Laboratory, IBM, 2006

[BBC06] R. Barrett, M. Berry, T. F. Chan, et al, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods (2nd Edition),* PA: SIAM, Philadelphia, 1994.

[BBHP06] K. Berkenkötter, S. Bisanz, U. Hannemann, J. Peleska , HybridUML Profile for UML 2.0, *International Journal on Software Tools for Technology Transfer (STTT),* Springer, Berlin / Heidelberg, 8, 2006.

[BC04] C. Y. Baldwin, K. B. Clark, *Modularity in the Design of Complex Engineering Systems, in Complex Engineered Systems: Science Meets Technology*, Springer, Berlin, 2006.

[Bekker60] M. G. Bekker, *Off-the-Road Locomotion,* University of Michigan Press, Michigan, 1960.

[Bekker69] M. G. Bekker, *Introduction to Terrain-Vehicle Systems*, University of Michigan Press, Michigan, 1969.

[Bemporad99] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3), 1999, pp. 407–427.

[BG01] A. K. Samantaray, About Bond Graph, http://www.bondgraphs.com/about.html, 2003-08-10.

[BK93] M. Brielmann, B. A. Kleinjohann, Formal Model for Coupling Computer Based Systems and Physical Systems, *EURO-DAC*, 1993, pp.158–163.

[BR03] D. Braha, Y. Reich, Topological structures for modeling engineering design processes, *Res. Eng Design,* 14, 2003, pp.185–199.

[Brö95] A. P. Bröhl, *Das V-modell – Der Standard für die Softwareentwicklung*, Verlag, München/Oldenbourg, 1995.

[Broenik99] J. F. Broenik, Introduction to Physical Systems Modelling with Bond Graphs, *SiE Whitebook on Simulation methodologies*, http://www.ce.utwente.nl/bnk/papers/BondGraphsV2.pdf, 1999.

[Brooke03] H. Brooke, Mobility Analysis of Small Lightweight Robotic Vehicles, ME 567, http://www-personal.umich.edu/~bhaueise/robotics_web/, 2003.

[BSHW97] M. Brielmann, J. Stroop, U. Honekamp, P. Waltermann, Simulation of hybrid mechatronic systems: a case study, *Proc. Int. Conference*, 1997.

[CLAWAR] Climbing and Walking Robots, http://www.uwe.ac.uk/clawar/, 2007-03-01.

[Coatanea05] E. Coatanéa, *Conceptual Modelling Of Life Cycle Design - A Modelling and Evaluation Method Based on Analogies and Dimensionless Numbers*, Doctoral Dissertation, Espoo, 2005.

[Cross89] N. Cross, *Engineering Design Methods*, John Wiley & Sons, New York, 1989.

[Crow] K. Crow, Customer-Focused Development with QFD, DRM Associates, http://members.aol.com/drmassoc/QFD.html, 2007-04-01.

[Cusu91] M. A. Cusumano*, Japan's Software Factories: A Challenge for US Management*, Oxford University Press, New York, 1991.

[CYHHSM07] E. Coatanéa, B. Yannou, S. Honkala, M. Hämäläinen, T. Saarelainen, P. Makkonen, A. Lajunen, Measurement theory and dimensional analysis: methodological impact on the comparison and evaluation process, *Proc. of ASME Design Theory and Methodology Conference: DETC-2007*, Las Vegas, 2007.

[CYHSML07] E. Coatanéa, B. Yannou, S. Honkala, T. Saarelainen, P. Makkonen, A. Lajunen, Towards an automatic synthesis of design solutions via the use of classifications and semantic atlas, *Proc. of ASME Design Theory and Methodology Conference: DETC-2007*, Las Vegas, 2007.

[Davoren00] J.M. Davoren, A. Nerode, Logics for Hybrid Systems, *Proc. of the IEEE*, 88 (7), 2000.

[Desel01] J. Desel, G. Juhas, What Is a Petri Net? - *Informal Answers for the Informed Reader, Hartmut Ehrig et al. (Eds.): Unifying Petri Nets*, LNCS 2128, 2001, pp. 1-25.

[DLC89] E. H. Durfee, V. R. Lesser, D. D. Corkill, Trends in Cooperative Distributed Problem Solving. In: *IEEE Transactions on Knowledge and Data Engineering*, KDE-1(1), 1989, pp. 63-83.

[DODAF] DoD Architectural Framework, Version 1.0, Volume I, "Definitions and Guidelines", 2004.

[Dogruel97] M. Dogruel and Ü. Özgüner, Discrete and Hybrid State System Modeling and Analysis, *Turkish Journal of Electrical Engineering and Computer Science*, 5 (2), 1997.

[Dombroski00] T. Dombroski, *Creative Problem Solving: The Door to Individual Success and Change*, toExcel Press and iUniverse, Lincoln, 2000.

[EK96] P. Estraillier, F. Kordon, Structuration of large scale Petri nets: an association with higher level formalisms for the design of

multi-agent systems In: *IEEE International Conference On Systems, Man And Cybernetics*, Beijing, 1996.

[EURON] European Robotics Research Network, http://www.euron.org/, 2007-05-01.

[EUROP] European Robotics Platform, http://www.robotics-platform.eu.com/, 2007-03-01.

[Frank07] J. Gausemeier, U. Frank, New Methods for the conceptual design of intelligent mechatronic systems, *Int. Workshop on Research & Education in Mechatronics*, Tallinn, 2007.

[Frankel06] D. S. Frankel, A Response to Forrester, *MDA Journal*, 2006.

[French99] M. French, *Conceptual Design for Engineers* (3rd Edition), Springer, London, 1999.

[FSGRZ01] Z. Fan, J. Hu, K. Seo, E. Goodman, R. Rosenberg, B. Zhang, Bond Graph Representation and GP for Automated Analog Filter Design, *Proc. of the Genetic and Evolutionary Computation Conference*, San Francisco, 2001, pp. 81-86.

[GARTNER] Hype Cycle for Emerging Technologies, Gartner Group, http://www.gartner.com/DisplayDocument?ref=g_search&id=494180, 2006-12-01.

[Gates07] B. Gates, A Robot in Every Home The leader of the PC revolution predicts that the next hot field will be robotics, Scientific American Inc., New York, 2007, pp. 58-65.

[Gausem03] J. Gausemeier, S. Moehringer, VDI 2206 - A New Guideline For The Design Of Mechatronic Systems, *Int. Conference On Engineering Design ICED 03*, Stockholm, 2003.

[Gawthrop99] P. J. Gawthrop, D. J., Ballance, Symbolic computation for manipulation of hierarchical bond graphs, *Symbolic Methods in Control System Analysis and Design*, N. Munro (ed), IEE, London, 1999.

[GFK07] J. Gausemeier, U. Frank, S. Kahl, Methodology for the conceptual design of intelligent mechatronic systems, *Int. Workshop on Research & Education in Mechatronics*, Tallinn, 2007.

[Granda02] J. J. Granda, The role of bond graph modeling and simulation in mechatronics systems An integrated software tool: CAMP-G, MATLAB–SIMULINK. In: *Mechatronics,* 12 2002, pp. 1271–1295.

[Grim00] J. Grimbleby, Automatic analogue circuit synthesis using genetic algoriths. *IEE Proc. – Circuits Devices Systems*, 2000, pp. 319-323.

[Gurd03] A. Gurd, Using UMLTM 2.0 to Solve Systems Engineering Problems, Telelogic White Paper, http://whitepapers.zdnet.co.uk, 2003.

[HB01] J. Heitkötter, D. Beasley, Hitch-Hiker's Guide to Evolutionary Computation, http://www.aip.de/~ast/EvolCompFAQ/, 2001.

[HDS] Hybrid Dynamic systems, http://moncs.cs.mcgill.ca/people/mosterman/cacsd/hds/software.shtml, 2006-05-01.

[Higgins93] J. M. Higgins, *101 Creative Problem Solving Techniques*, New Management Pub.,1993

[HLSY00] A. Halme, I. Leppänen, S. Salmi, S. Ylönen, Hybrid locomotion of a wheel-legged machine. *3rd Int. Conference on Climbing and Walking Robots*, Prof. Engineering Publishing Ltd., 2000, pp. 167-173.

[HLSYK03] A. Halme, I, Leppänen, S. Ylönen, I. Kettunen, Workpartner: Interactive Human-like Service Robot for Outdoor Applications, *Int.l Journal of Robotics Research*, 22, 2003, pp. 627-640.

[HM00] S. Hsiung, J. Matthews, An Introduction to Genetic Algorithms. In: *Generation5* http://www.generation5.org/content/2000/ga.asp, 2003-04-10.

[Honda06] Honda ASIMO report, Honda, CD-ROM, 2006.

[Hubka96] V. Hubka, W. Eder, *Design science: introduction to the needs, scope and organization of engineering design knowledge*, Springer-Verlag, London, 1996.

[HW02] A. Hatchuel, B. Weil, C-K theory: Notions and applications of a unified design theory, *Proc. of the Herbert Simon Int. Conference on Design Sciences*, Lyon, 2002.

[HW98] W. Hsu, I.M.Y. Woon, Current Research in the Conceptual Design of Mechanical Products, *Computer-Aided Design*, 30 (5), 1998, pp. 377-389.

[HZ07] P. Hehenberger, K. Zeman, Modularization and Integration in Design from a Mechatronics-oriented Point of View, *Int. Workshop on Research & Education in Mechatronics*, Tallinn, 2007.

[IEC05] international standard IEC 61499 First edition, International Electrotechnical Commission, 2005.

[INCOSE04] *Systems Engineering Handbook*, INCOSE-TP-2003-016-02, Version 2a, Technical Board of International Council on Systems Engineering, 2004.

[iRobot] iRobot PackBot, http://www.irobot.com/, 2007-01-20.

[Jetro06] Trends in the Japanese Robotics Industry, *Industrial Reports JETRO Japan Economic Monthly,* 2006.

[JGJ98] I. Jacobson, M. Griss, P.Jonsson, *Software Reuse: Architecture, Process and Organization for Business Succes,* ACM Press, Addison-Wesley, New York, 1998.

[KBAKD97] J.R Koza, F.H. Bennett, D. Andre, M. Keane, A. Dunlap. Automated synthesis of analog electrical circuits by means of

genetic programming, *IEEE Trans Evolution Comput,* 1(2), 1997, pp. 109–28.

[KBS97] E. Kallenbach, O. Saffert, C. Schäffel, *Zur Gestaltung integrierter mechatronischer Produkte, in Tagung Mechatronic im Maschinen- und Fahrzeugbau, Moers, VDI Berichte 1315*, Düsseldorf, VDI Verlag, 1997.

[KRH05] P. Kukkala, J. Riihimäki, M. Hännikäinen, T. D. Hämäläinen, K. Kronlöf, UML 2.0 Profile for Embedded System Design, *Proc. of the Design, Automation and Test in Europe Conference*. Munich, 2005, pp. 710-715.

[KT04] A. Kalja, T. Tiidemann. TRIZ and Artificial Intelligent Software in the Conceptual Design, *The Sixth Annual Conference of the Altshuller Institute for TRIZ Studies. TRIZCON2004,* Washington, 2004. pp. 29-1-29-7.

[KTT97] B. Kleinjohann, J. Tacken, C. Tahedl, Towards a complete design method for embedded systems using preddicate/transition-nets, *Int. Conference on Computer Hardware Description Languages and Their Applications*, Toledo, 1997, pp. 256–262.

[LC99] J. Lohn, S. Colombano, A circuit representation techniques for automated circuit design, *IEEE Transactions on Evolutionary Computation,* 1999, pp. 205-219.

[Lotter86] B. Lotter, *Manufacturing Assembly Handbook*. Butterworths, Boston, 1986.

[MDA] *The official MDA Guide Version 1.0.1*, OMG Document Number: omg/2003-06-01, 2003.

[Mech96] IEEE/ASME Transactions on *Mechatronics*, 1 (1), 1996.

[Modelica] Modelica – A Unified Object-oriented Language for Physical Systems Modeling, *Language spec. ver. 2.1*, Modelica Assocation, http://www.modelica.org, 2004.

[OKK] D. W. Oliver, T. P. Kelliher, J. G. Keegan, *Engineering Complex Systems with Models and Objects*, McGraw-Hill,1997.

[OMGSysML] What is OMG SysML?, http://www.omgsysml.org/, 2007-02-10.

[Paynter69] H. Paynter, Bond Graphs and Diakoptics, *The Matrix Tensor Quarterly*, 19(3), 1969, pp. 104-107.

[PB97] G. Pahl, W.Beitz, *Konstruktsionslehre – Methoden und Anwendung,* Berlin, Springer Verlag, 1997.

[PBFG07] G. Pahl, W. Beitz, J. Feldhusen, K.-H. Grote, *Engineering Design: A Systematic Approach*, Springer, UK, 2007.

[Pelz03] G. Pelz, *Mechatronic Systems. Modelling and Simulation with HDLs*, John Wiley & Sons, West Sussex, 2003.

[Petri62] C. A. Petri, *Kommunikation mit Automaten*, Ph. D. Thesis. University of Bonn, 1962.

[Pettai05] E. Pettai, *Tootmise Automatiseerimine*, Tallinn, 2005.

[POK01] T. Pokorny, *The Model Driven Architecture: No Easy Answers*, Simulation Industry Association of australis, 2001.

[QFDHb] J. ReVelle, J. Moran, C.Cox, *The QFD Handbook*, Wiley, New York, 1998.

[RHSZN05] S. P. Rajan, T. Hasegawa, M. Shoji, Q. Zhu, and T. Nakata, UML Profile for SoC RFC, *DAC 2005 Workshop UML-SoC 2005 UML for SoC Design Conference*, Anaheim, 2005.

[RIA105] Robotics Industry Sets New Records in 2005, Robotic Industries Association, *Industry report*, 2005.

[RIA205] Robotics Industry Posts 30% Growth in North America, Robotic Industries *Association, Industry report*, 2005.

[RSA] *IBM Rational Software Architecture*, http://www.ibm.com, 2007-02-25.

[Rzevski03] G. Rzevski, On Conceptual design of intelligent mechatronic system, *Mechatronics,* 13, 2003, pp. 1029–1044.

[SA04] D. Sharman, Y. Ali, Characterizing Complex Product Architectures, *Systems Engineering Journal*, 7 (1), 2004.

[SBK01] U. Saranli, M. Buehler, D.E. Koditschek, RHex: A Simple and Highly Mobile Hexapod Robot, *Int. Journal of Robotics Research*, 20, 2001, pp. 616-631.

[Schlegl97] T. Schlegl, M. Buss, Günther Schmidt, Development of numerical Integration Method for Hybrid (Discrete-continuos) Dynamic Systems, *Proc of Advanced Mechatronics*, *AIM97*, Tokyo, 1997.

[Sell04a] R. Sell, Mechatronics Design Process and Methologies, *11th Int. Power Electronics And Motion Control Conference EPE-PEMC,* Riga, 2004.

[Sell04b] R. Sell, Mechatronics System Design In Conceptual Stage, *4th Int. DAAAM Conference Industrial Engineering - Innovation As Competitive Edge For SME*, Tallinn, 2004.

[Sell04c] R. Sell, New Object-Oriented Approach of Modelling Mechatronics System in Conceptual Stage, *5th Int. Workshop on Research and Education in Mechatronics*, Gliwice, 2004.

[Sell05] R. Sell, Integration of V-model and SysML for advanced mechatronics system design, *Proc. of Int. Workshop on Research & Education in Mechatronics*, d'Annecy, 2005.

[SEO03] K. Seo, Z. Fan, J. Hu, E. D. Goodman, R. C. Rosenberg, Toward a unified and automated design methodology for multi-domain dynamic systems using bond graphs and genetic programming, *Mechatronics,* 13, 2003, pp. 851–885.

[SERFP] UML for Systems Engineering RFP, *OMG Document: ad/03-03-41*, 2003.

[SL03] R. Sell, P.Leomar, Methodologies on the Mechatronics Domain, *The Eighth Symphosium on Machine Design*, Oulu, 2003.

[ST03] R. Sell, M. Tamre, Component Based Mechatronics Modeling, *ICOM 2003 International Conference on Mechatronics*, Prof. Engineering Publishing Limited, London and Bury St Edmunds, 2003, pp.111-116.

[STL] Standard Template Library, Wikipedia, 2007-01-10.

[STL05] R. Sell, M. Tamre, P. Leomar, Design Of Modular UGV Using SysML And Extensions, *OST Conference*, Stockholm, 2005.

[STLR07] R. Sell, M. Tamre, M. Lehtla, A. Rosin, Conceptual Design Method for General Electric Vehicle, *Proc. of the Estonian Academy of Sciences Engineering*, 2007.

[SysML09] System Modeling Language (SysML) Specification. Version 0.90 Draft. *OMG document ad/2005-01-10*, http://www.sysml.org, 2005-09-01.

[SysML10] System Modeling Language (SysML) Specification. Version 1.0 Draft. *OMG document ad/2006-03-01*, http://www.sysml.org, 2006-04-01.

[SysMLF] SysML forum discussion thread - Connection between Activities and States, 2006 / 2007.

[Talon] Foster-Miller Talon robot, http://www.fostermiller.com/lemming.htm, 2007-01-30.

[TeleL07] Leveraging the New Simulink and UML/OMG SysML Integration from Telelogic to Solve Hybrid Engineering Design Challenges, Telelogic webinar, Detroit-Troy, 2007.

[TFB98] E. Tay, W. Flowers, J. Barrus, Automated generation and analysis of dynamic system designs. *Res Eng Des*, 10, 1998, pp. 15–29.

[Tiidem01] M. Tiidemann, *Mehaaniliste ülekannete arvutamine konseptuaalsel projekteerimiel*, master thesis, Tallinn University of Technology, 2001.

[TS05] S. Turki, T. Soriano, A SysML extension for Bond Graphs support, 2005 http://icta05.teithe.gr/papers/50.pdf, 2005-01-12.

[Tyugu06] E. Tyugu, Extensible Multipurpose Simulation Platform, *Proc. Of the 6th WSEAS Int. Conf. on Simulation, modeling and Optimization*, Lisbon, 2006, pp. 738-743.

[Tyugu91] E. Tyugu, Three new-generation software environments, *Comm. Of the ACM,* 34 (6), 1991, pp. 46-59.

[UE03] K. T. Ulrich, S. D. Eppinger*, Product Design and Development*, McGraw Hill Higher Education, 2003.

[Ullman02] D. G. Ullman, *Mechanical Design Process*, McGraw – Hill, 2002.

[UML] Unified Modeling Language (UML) Specification: Infrastructure version 2.0, *OMG Adopted Specification, ptc/03-09-15,* http://www.omg.org/, 2003.

[UMLSPT] UML Profile for Schedulability, Performance, and Time Specification, *OMG Draft Available Specification ptc/2003-03-2*, http://www.omg.org, 2003.

[UMLTP] UML 2.0 Testing Profile Specification, version 1.0, *OMG document formal/05-07-07*, 2005.

[VD06] Y. Vanderperren, W. Dehaene, From UML/SysML to MatLab/Simulink: Current State and Future Perspectives, *Proc. Design, Automation and Test in Europe (DATE) Conf.*, Munich, 2006.

[VDI04] *VDI 2206 Design methodology for mechatronic systems*, Verein Deutscher Ingenieure, Berlin, 2004.

[VDI2221] *VDI 2221 Systematic approach to the development and design of technical systems and products*, Verein Deutscher Ingenieure, Berlin, 1993.

[VDI2422] *VDI 2422 Systematical development of devices controlled by microelectronics*, Verein Deutscher Ingenieure, Berlin, 1994.

[Viehl06] A. Viehl, T. Schönwald, O. Bringmann, W. Rosenstiel, Formal Performance Analyse and Simulation of UML/SysML Models for ESL Design, *Design, Automation and Test in Europe (DATE),* Munich, 2006.

[Weilkiens06] T. Weilkiens, *Systems Engineering mit SysML/UML – Modellierung, analyse, Design, dpunkt.verlag*, Heidelberg, 2006.

[WIDNEY06] C. Widney, An IBM Rational approach to the Department of Defense Architecture Framework (DoDAF) Part 1: Operational view, IBM, 2006.

[WIRA] Worldwide Market, International Robotic Association, http://www.robotinvestments.com/RI_worldwide_market.htm, 2007-02-20.

[Ylönen06] S. Ylönen, *Modularity in Service Robotics – Techno-Economic justification thorough a Case Study,* doctoral thesis, Helsinki, 2006.

[Zadeh65] L.A. Zadeh, Fuzzy sets, *Information Control,* 81965, pp. 338–353.

[Zeite06] C. Zeite, MDA Is DOA, Partly Thanks To SOA, Forrester, http://www.forrester.com/Research/Document/Excerpt/0,7211,39156,00.html, 2006-10-01.

Elulookirjeldus

1.Isikuandmed

Ees- ja perekonnanimi	Raivo Sell
Sünniaeg ja koht	09.03.1975 Tallinn
Kodakondsus	Eesti
Perekonnaseis	vabaabielus
Lapsed	Ingmar (s. 2001), Sirli (s. 2007)

2. Kontaktandmed

Aadress	Akadeemia tee 5a-60, Tallinn
Telefon	+372 620 32 01
E-post	raivo@staff.ttu.ee

3. Hariduskäik

Õppeasutus	Lõpetamise aeg	Haridus
Tartu Ülikool	2002	informaatikaõpetaja
Tallinna Tehnikaülikool	2001	teadusmagister
Tallinna Tehnikaülikool	1999	diplomiinsener
Võhma Keskkool	1993	keskharidus

4. Keelteoskus

Inglise	kesktase
Vene	algtase
Soome	algtase
Saksa	algtase

5. Täiendusõpe

Õppimise aeg	Õppeasutuse või muu organisatsiooni nimetus
2007	Helsinki University of Technology
2003	ETH Zürich
2002	KTH Stockholm

6. Teenistuskäik

Töötamise aeg	Organisatsiooni nimetus	Ametikoht
1994	AS "Tallinna Paber"	varustaja
1995 ...1998	Tallinna Tehnikaülikool	arvutiklassi haldur
1997 ...1999	Software Engineering Center OÜ	tarkvara konsultant
1998 ... 1999	Armila Eesti OÜ	arvutispetsialist
1999 ... 2002	Viljandi ÜKK	IT osakonna juht
2002 …	Tallinna Tehnikaülikool	teadur

7. Teadustöö põhisuunad

Mehhatroonikasüsteemi modelleerimise metoodika, mobiilsed robotid.

Curriculum Vitae

1.Personal data

Name	Raivo Sell
Date and place of birth	09.03.1975 Tallinn
Citizenship	Estonian
Family status	open marrage
Children	Ingmar (b. 2001), Sirli (b. 2007)

2. Contact

Address	Akadeemia tee 5a-60, Tallinn
Phone	+372 620 32 01
E-mail	raivo@staff.ttu.ee

3. Education

Institution	Graduated	Speciality
University of Tartu	2002	informatics
Tallinn University of Technology	2001	M.Sc.
Tallinn University of Technology	1999	diploma eng.
Võhma High School	1993	sec. education

4. Language skills

English	Intermediate
Russian	Elementary
Finnish	Elementary
German	Elementary

5. Further training

Apperenticeship	Educational or other organization
2007	Helsinki University of Technology
2003	ETH Zürich
2002	KTH Stockholm

6. Professional employment

Period	Organization	Position
1994	AS "Tallinna Paber"	supplier
1995 ...1998	Tallinn University of Technology	computer manager
1997 ...1999	Software Engineering Center OÜ	software consultant
1998 ... 1999	Armila Eesti OÜ	ICT support
1999 ... 2002	Viljandi ÜKK	head of ICT dep.
2002 …	Tallinn University of Technology	researcher

7. Main research interest

Mechatronics system modeling methodology, mobile robotics.