Estonian Journal of Engineering, 2008, **14**, 1, 3–16 doi: 10.3176/eng.2008.1.01

A conceptual design method for the general electric vehicle

Raivo Sell[a], Mart Tamre[a], Madis Lehtla[b] and Argo Rosin[b]

[a] Department of Mechatronics, Tallinn University of Technology, Ehitajate tee 5, 19086 Tallinn, Estonia; {raivo,mart}@staff.ttu.ee
[b] Department of Electrical Drives and Power Electronics, Tallinn University of Technology, Ehitajate tee 5, 19086 Tallinn, Estonia; {mlehtla,vagur}@cc.ttu.ee

Received 19 January 2007

Abstract. The paper discusses conceptual design of mechatronic systems considering a mobile electrical vehicle platform as an application example. A set of design templates are developed and organized into libraries for the use in early stages of the system design. The advantages of retaining usability of component libraries, allowing verification of design alternatives on the conceptual level are demonstrated.

Key words: mechatronics, mobile robotics, system design, conceptual modelling, simulation.

1. INTRODUCTION

The design of mechatronic systems differs considerably from the domains like mechanics, electronics, etc. Although mechatronics is often defined as a combination of mechanics, electronics and control theory, design of a mechatronic product can not be divided into three separate parts. A mechatronic system needs to be designed as an integrated product from the very beginning. Domain-specific design plays also a certain role but the designer must always be aware of the interaction of design aspects of various components. Mechatronic system design is closely related to system engineering and therefore many tools and techniques used in system engineering are applicable also in the mechatronic system design.

Decomposition of the general design cycle has been considered by several authors [1–3]. The design cycle starts with a conceptual stage, which consists of the specification of the requirements and situation analysis. According to French [4], the conceptual design stage puts greatest demands on the designer and in this stage the most important decisions are made. The result of this process is a candidate for the design solution and a clearly formulated set of desired measurable properties of the future product, which introduces the quality measure into the design process. This cross-domain design takes into account the overall system requirements and goals. The conceptual design as well as other design stages are implemented in many cycles. Complex mechatronic product design cycles (macrocycles) are described in greater detail in [5].

In real design, many tools and techniques are used to carry out the whole design process. For the domain-specific stage, a large selection of most advanced tools exist. Conventional domains like mechanics, electronics and control engineering are well exploited. Computer-aided engineering (CAE) tools like computer-aided design (CAD), computer-aided manufacturing (CAM), finite element method (FEM), printed circuit board (PCB) routing & layout, etc. are probably known for every engineer. In software design, computer aided software engineering (CASE) and the unified modelling language (UML) are used [6,7].

On the conceptual design stages fewer tools are available, although there is a great needed for CAE. Domain-independent techniques as Bond graphs [8], Petri nets [9] and hybrid automata [10] are not widely used in the design of mechatronic systems. However, recent research is focused on the conceptual design and automation of the generation of the candidate for the design solution. Several investigations [11–13] exploit the combination of artificial intelligence and domain-independent techniques. One of the reasons why the conceptual level lacks computer support is that the used methodology must support high-level conceptual design with the option to apply very specific constraints at the same time. The system design stage needs similar tools, but the system must be modelled on a more detailed level. The system components, subcomponents, behaviour and performance, etc., must be modelled in the frame of the whole system. The most used technique here is block diagrams of different modifications. In the recent years many efforts are put on the system design and mechatronics software development. Some software package examples are AMESim [14], Dymola [15], 20-sim [16] and also the well known MatLab/Simulink environment.

Methodology of the mechatronic design process is presented in the mechatronic design guideline VDI 2206 [5], which proposes several tools and techniques for the design of mechatronic systems.

The objective of the present work is to combine mechatronics design methods with the system modelling language and to develop practical tools for the design of a mobile electrical vehicle platform on conceptual level. We create a specific toolbox for a general electric vehicle, which can be utilized in the conceptual design process. Some practical examples are shown, based on current projects at Tallinn University of Technology.

2. DESIGN APPROACH

The conceptual design method, considered in this paper, expands the approach [5] with the design templates and concept simulation aids. System modelling language (SysML) [17] is used as the basic modelling language. The general conceptual design modelling approach is described in Fig. 1.

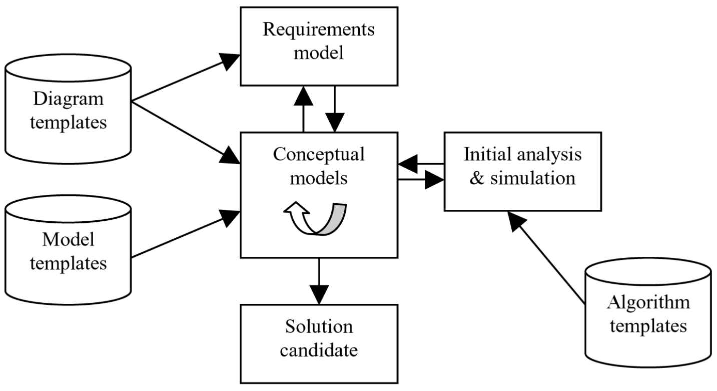

Fig. 1. The SysML toolbox for mobile platform design.

The approach consists of three integrated substages: requirements modelling, conceptual solution development and design candidate simulation. All stages are supported by specific template libraries. A template class from the template library provides a parameterized description of the model element, specifying its attributes and operations. By binding multiple elements to the template it is possible to generate new elements with the same characteristics as the template.

SysML is used for requirement and concept modelling and Simulink for conceptual simulation, although other tools are not excluded. The proposed approach relies on the design methodology [18] and the SysML toolbox for the mobile platform design, which is taken as an example. The mobile platform is a generalization of different types of (mainly electrical) vehicles. Mobile robots, unmanned ground vehicles (UGV) and railway vehicles are used as examples for different diagrams later on.

3. MODELLING OF THE REQUIREMENTS

Formulation of the requirements is the foundation of a project. Every requirement is tightly related to the cost and therefore the requirements modelling and analysis must be carried out with great care. Big changes in requirements in later stages of the design process may increase significally the cost of the whole project. Requirements arise from customer needs, regulations, legislation, organization environment, technology availability, etc.

Definition of the requirements is a complex process and typically includes performance analysis, trade studies, constraint evaluation and cost analysis. Requirements modelling is not just a top-down process, but must be carried out with the interaction of the initial analysis and simulation of the concepts.

Initial requirements model is completed usually on the system level. Once archived, it is necessary to allocate and flow the requirements down to lower levels. According to [19], the requirements modelling process is iterative for each phase, with continuous feedback as the level of design specifications increases.

The design of an electrical vehicle and mobile platform follows the system engineering concept. The general requirements model is shown in Fig. 2. The model is based on the SysML requirements diagram and describes general concept of the requirements model. A single requirement is described as a box with various parameters. The requirement can be decomposed into subrequirements and is linked with each of them as well as with analysis, design, implementation and testing elements. In a general requirement element the following parameters are used: ID (unique identifier across the model), priority, text (textual representation of the requirement or reference to a document), risk, weight, type, etc.

According SysML specification, a requirement can be generated or deduced from another requirement using *Derive* relationship. A requirement can be fulfilled by another model element using *Satisfy* relationship. A requirement can be verified by various behaviours using the *Verify* relationship. Standard or specific test cases (*TestCase*) are developed for this purpose. All of these are specializations of the UML *Trace* relationship; which is used to track requirements and changes across the model [17].

Requirements template. Introducing new design tools to practising engineers is often related with various problems as people are used to work with habitual tools and methods. Therefore it is important to make the implementation of the new system as easy as possible. One way to do this is to use pre-defined model

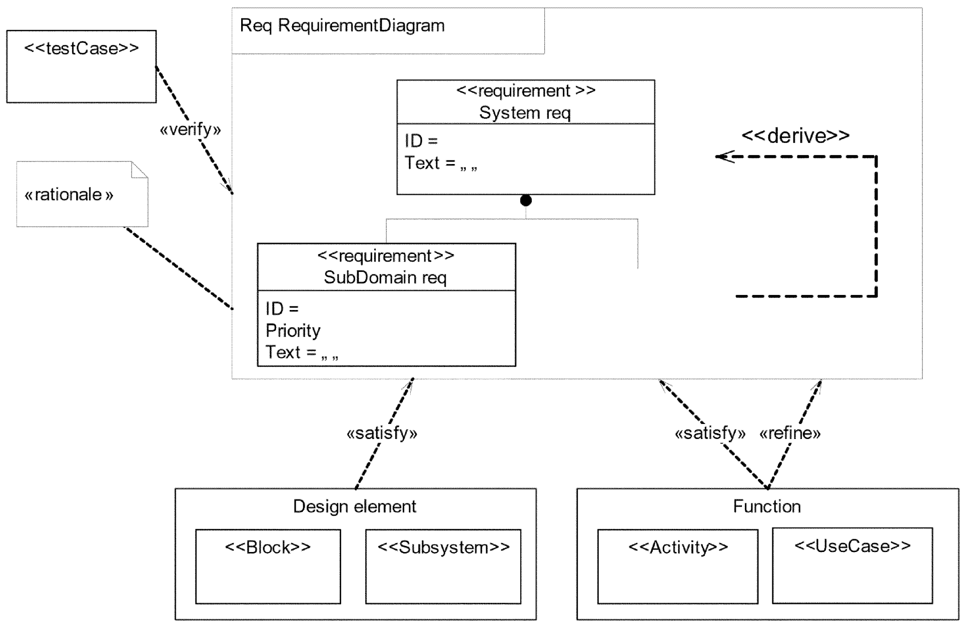

Fig. 2. Metamodel of the requirements.

templates. Templates are defined according to the specific product domain. In this paper we focus on the general electrical vehicle platform design. Templates are still general enough to be adopted to other subdomains.

A requirement is defined by ID, textual representation and parameters. Default parameters are *Weight*, *Risk*, *OptimizationDirection*, and *Source*. Optional parameters are *ConsistentStandard* and *MaxCost* (Fig. 3). Every single requirement has to be verified on some level by one of the following methods: test, demonstration, analysis, inspection. Therefore every single requirement has a relationship with the activity *TestCase*. If a requirement needs multiple tests, the requirement should be decomposed into multiple subrequirements.

When the process proceeds, the requirement can be connected with design elements as block, assembly, activity, etc. This relationship indicates specific design elements, which satisfy the particular requirement.

Templates are intended to be used for effective and professional requirement modelling. The engineer can pick the best matching template according to the design scope and start to bind the predefined requirement parameters with real values. Requirement templates consists of activities like standardized *TestCases*. These activities are collected to the knowledge base library, where they can be extracted and redesigned if necessary.

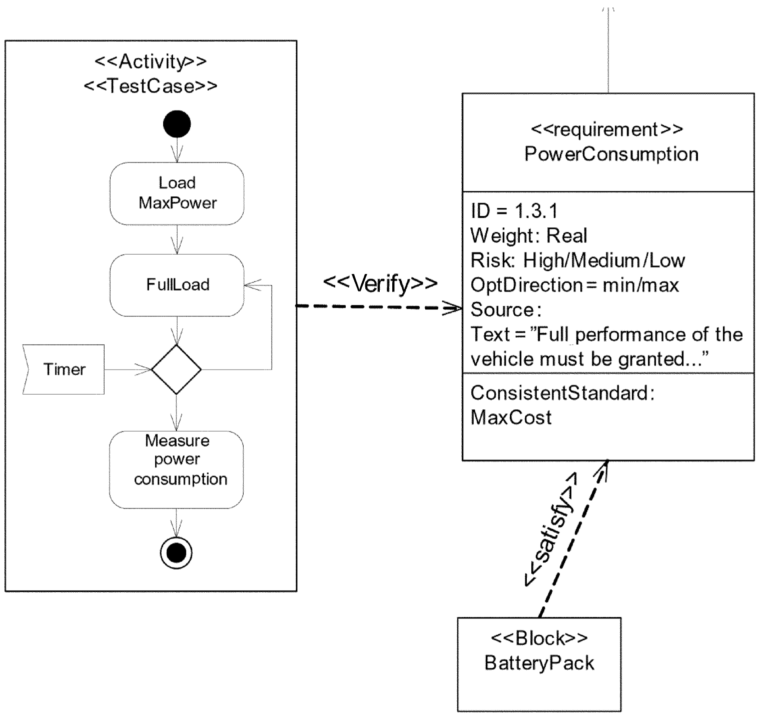

Fig. 3. Requirements metaclass and *Activity* relationship.

4. CONCEPTUAL MODELS

4.1. Conceptual design

Conceptual design is actually the first stage, where engineers start to develop the target system, corresponding in an optimal way to requirements. This stage is tightly related to the previous one, to the requirements modelling part. After starting to develop actual concepts of the system, often new aspects arise and very often they cause some changes in initial requirements. This is an interactive process and must be well coordinated.

A frequent mistake in this interactive process, made by beginners, is that the very fist idea is taken as the best one and is developed into a product [3]. This may be a very costly approach. Correction of design mistakes and conceptual changes in a later, product development or integration stage, will cost much more than in the conceptual stage. Therefore it is very important to develop more than one candidate for the solution. Methodical comparison and initial simulation will ensure that the optimal solution is selected and the risk of project failure is reduced.

4.2. Development of the conceptual solution

Development of a candidate for a conceptual solution is the next step after the requirements model is established. Our concept is described by the static structure, interaction of components, behaviour and dynamic parameters. All these aspects are modelled with a corresponding diagram. At the same time, the diagrams can interact with each other and one diagram can consist of parts from other diagrams. In the concept design process, the requirements model must be kept in mind and relationships with the requirements diagram are allowed and strictly suggested through the <<satisfy>> relationship. Diagram template library incorporates categories for concept design shown in Table 1. Diagram templates are divided logically according to SysML diagram types.

The interaction levels are as follows:

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

Level II – Subsystems are opened and defined in general 'black box' components; parameters of subsystems are initiated, subsystem inner activities are distinguished.

Table 1. Diagram template library

Name	Diagram (acronym)	Level
For the whole concept		
Hierarchical structure	Block Definition (bdd)	I
System usage	Use Case (uc)	I
For every solution candidate		
Component interaction	Internal Block diagram (ibd)	I
Behaviour	Activity (act)	II
Dynamics	Parametric (par)	II

The first two diagrams (bdd, uc) are common for all solution candidates because in the context level they do not differ very much for various solution candidates.

System usage at the context level is defined by the Use Case diagram. Use Case diagram defines the usage of the system under the development. This is particularly important in the early design stage. The diagram describes high-level behaviour and bounding of the system at the same time. This diagram can be compared with the context diagram in the data flow diagram (DFD). We use the original structure of the Use Case package. The syntax is analogous to UML. In that way the software engineers and mechanical engineers use the same syntax to describe the system usage and context. This ensures that the gap in understanding will be reduced to minimum.

The concept level structure of the system is described by the Block Definition diagram (bdd). A Block is defined as a modular unit of the system and depending on the design detailization level the block describes different things. In the beginning of the conceptual stage, the hierarchy of the components is defined and disclosed to assembly level, e.g. vehicle drive. The Block encapsulates its contents, which include attributes, operations and constraints.

Every system has interconnections between its blocks. These connections can be quite different, e.g. flow of energy or material, software operations, data exchange, analogue signals, etc. In the system hierarchy a model generalization relationship is used instead of interactions between the components. The system is decomposed and linked with parts or assemblies from the model library.

Both the Block Definition diagram and the Use Case diagram are common for all solution candidates due to the reason that these diagrams describe a general view of the perspective system and are derived directly from the requirements. Solution-specific diagrams are more detailed and describe the specific solution.

Three different diagrams are defined for the solution-specific development in the conceptual stage:

- Internal Structure, describing the interaction between the components in terms of service and flow;
- Parametric, describing the key parameters and system dynamics;
- Activity, describing the behaviour.

4.3. Conceptual design templates

In the framework of the electrical vehicle profile, several design templates for the Block Definition diagram have been developed. The system hierarchy on the concept level has one main template with mostly common blocks. When starting to use the template, the engineer can select or not select these pre-defined blocks and their attributes. This enables to start quickly the system description without missing relevant components.

Most commonly used components in electrical vehicle design are presented with common attributes and the generalization relationship. Attributes are in most cases optional and can be added or removed depending on the design characteristics. The Internal Block diagram template describes the interrelationship between the movement components. Service and flow ports are defined but kept open for most cases.

Activity diagram templates are composed with the Swim Lane technique, where activities are mapped with blocks on the basis of functions. An example of this is shown in Fig. 4. Templates based on functions and behaviour describe more specific activities, e.g. *MotionTraction*, *Brake*, *Accelerate*, etc., and *TestCase* for the satisfaction of the requirements.

The general dynamic model [20] of the system or subsystem is shown schematically in Fig. 5. The system S is characterized by a set of state variables X that are influenced by a set of input variables, U, representing the action of the system's environment on the system. The set of output variables, Y, are observable indicators of the system's response.

The general dynamic model can be applied for all subsystems or components. The dynamic model can be executed and by feeding different input parameters different system behaviour is achieved. The system and its model can be linear or

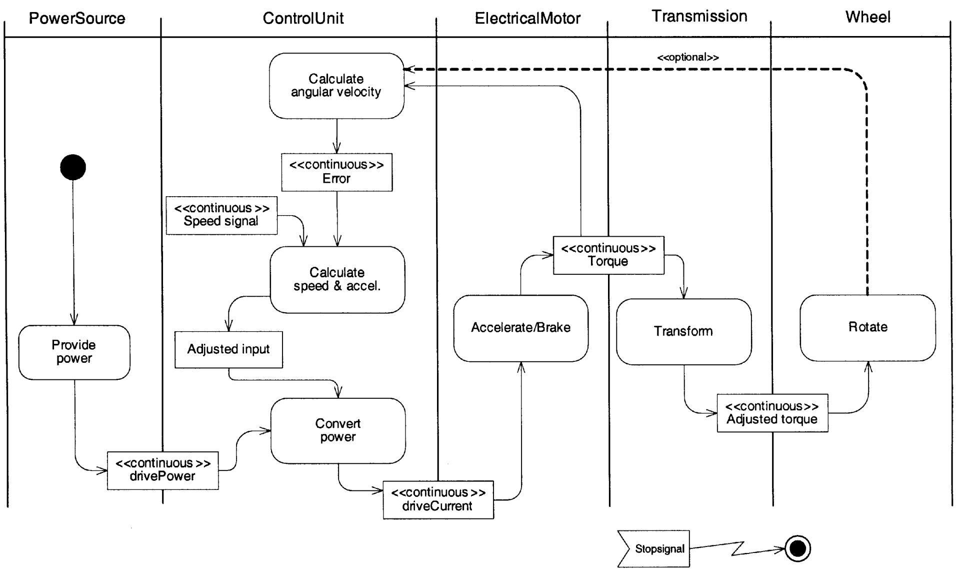

Fig. 4. Example of an Activity diagram.

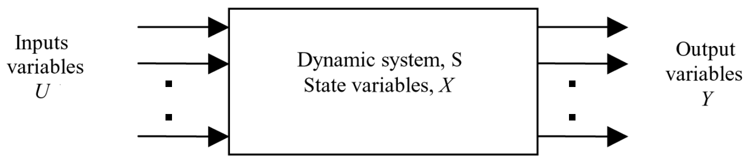

Fig. 5. Dynamic model of the system.

non-linear. Linear systems generally can be described by a set of linear first-order differential equations and it is possible to obtain detailed solutions of the system response. If one of the components or a subsystem is non-linear, the overall system is non-linear and conventional analytical tools do not work any more [20]. Solving the non-linear system, the simulations according to time steps must be carried out. 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 dynamic model of the system is not fully defined yet. The early stage system dynamics is determined by the Parametric diagram. This defines how one value of the structural property affects other values. Parametric constraints are tightly connected with the system structure and are used in combination with Block diagrams.

The Parametric diagram is the main input for the system simulation model. Different conceptual solutions can be obtained and simulation results used for the improvement of the design. For example this is used for the performance and reliability analysis as well as for meeting all the requirements, specified by the Requirement diagram.

The Parametric diagram template for basic performance of an electrical vehicle is taken as an example (Fig. 6).

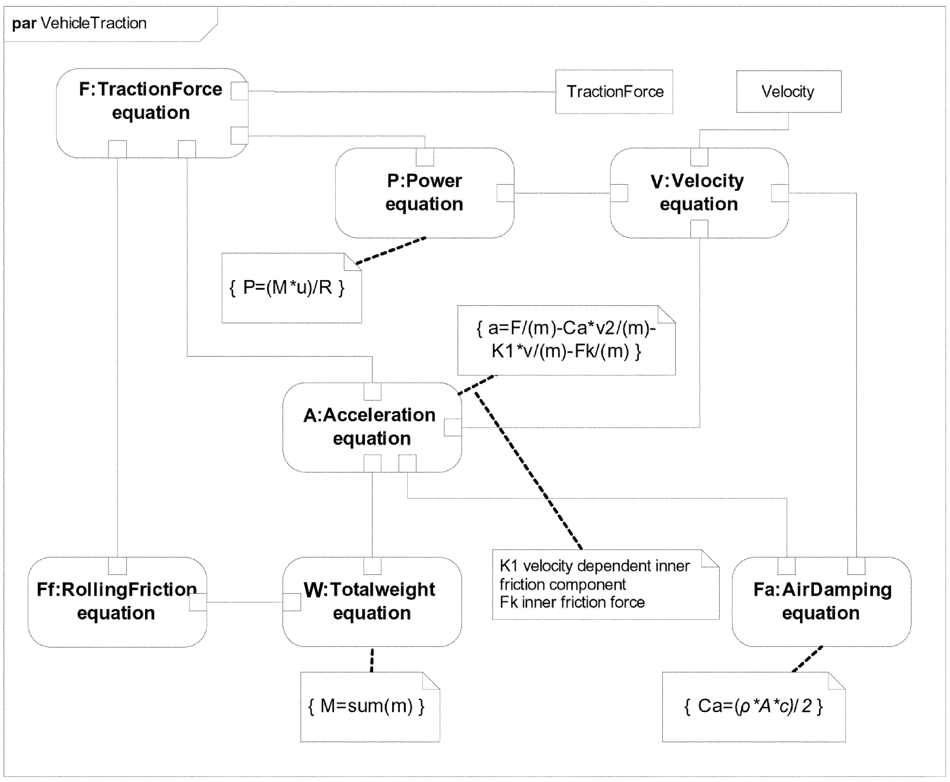

Fig. 6. Parametric example.

5. SIMULATION OF THE ELECTROMECHANICAL (PHYSICAL) MODEL

To describe interaction, a pair of variables should be used. One variable is an input, which describes the effect and another is the result or feedback, showing the reaction (or vice versa). The instantaneous power can be calculated from this pair of variables. Equations of the mechanical behaviour of the motors and voltage–electromotive force equations are placed in different blocks [21] according to the structure of the energetic macromodels. For example, electrical subsystems, described with transfer functions, have as an input the instantaneous value of voltage and the reaction is calculated as the instantaneous value of the current. Mechanical subsystems can have as input the linear or angular velocity and the calculated reaction is the force or torque.

A simulation model should be simplified as much as possible because of energetical, technical and time restrictions. A model is always a simplification of the reality for a certain purpose. The simplified electromechanical model does not always describe electrical parameters of windings, supply network, controllers and converters. The main purpose of modelling of the control object and load is virtual testing of control methods for the simplification of the development process. Models allow the verification of control methods and software in different operation modes, mode-altering conditions and different control modes [22].

For the verification of the model and comparison with the real system, it should be divided into subsystems using subsystem macromodels. These subsystem macromodels can be also divided into subsystem and component models. MatLab/ Simulink simulation model has a hierarchical structure and each block can be flexibly composed from configurable subblocks. The grouping of the blocks should take into account different configurations of the drive hardware, such as different motor–wheel configurations, different compositions of supply converters, motors, etc. This can be done via parametric (Fig. 6) and operation mode transition (Fig. 7) diagrams.

Events that cause changes in the control structure can be defined as transitions in the operation-mode transition diagram shown in Fig. 7. Each state describes a different set of control structures and algorithms.

Conceptual models, developed with design tools in Mobile Platform Toolbox, will be linked with the MatLab/Simulink object in the template model. Toolbox consists of several predefined models for multiple purposes. For example, the simulation models of electrical vehicle performance, current consumption, efficiency, etc., are stored in the simulation template library. A template is a general simulation model for a specific simulation target. Appropriate modification will be done and correct parameters assigned for the picked template model. The following example shows a simple simulation model template for electrical and performance simulation in different operation modes. The Simulink model shown in Fig. 8 uses torque reference values and state variables of wheels as arrays (multiple numbers) indicated with bold lines in the figure.

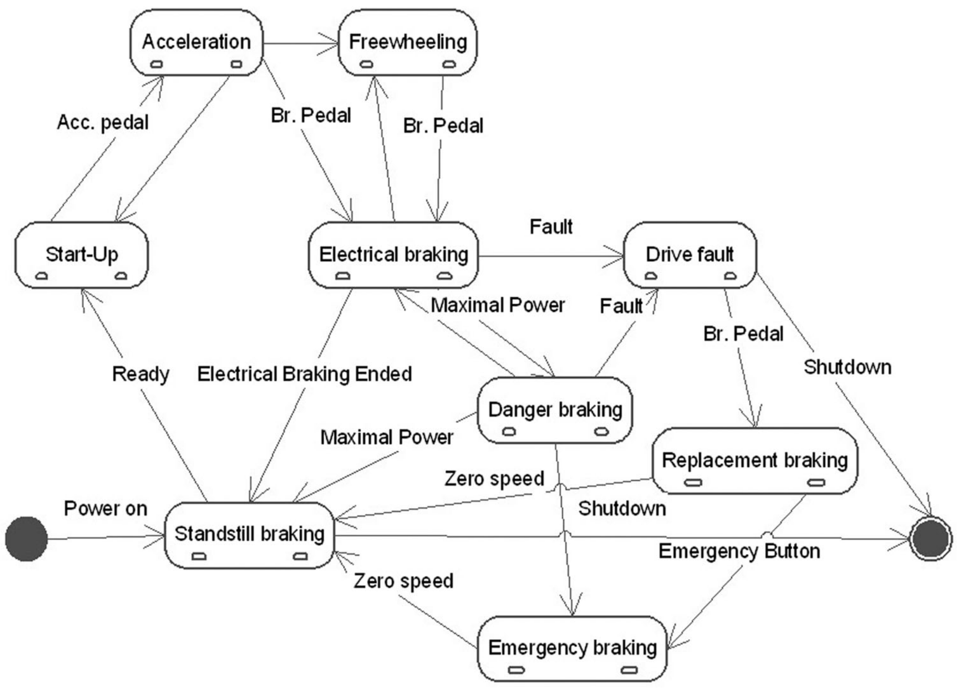

Fig. 7. Operation mode transition diagram of the drive.

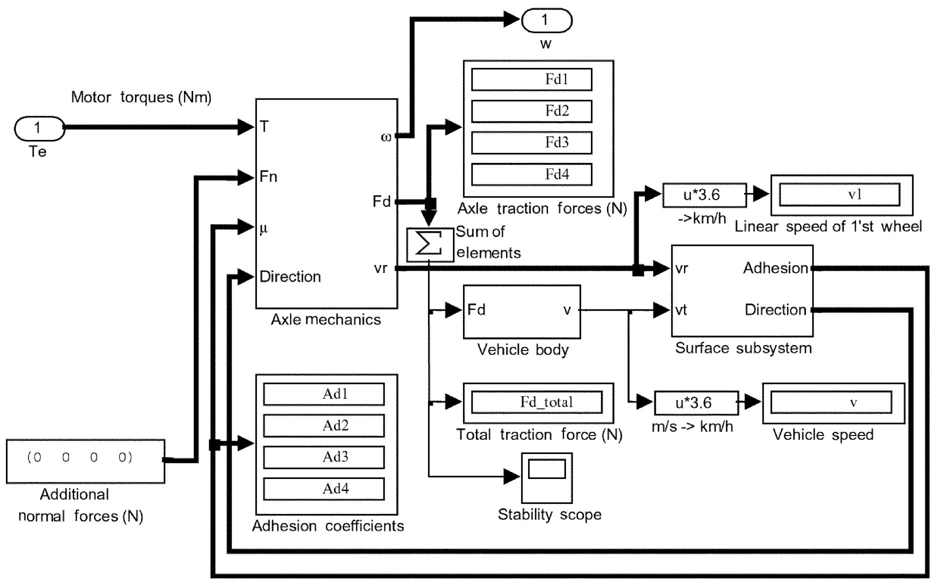

Fig. 8. Simulation model of a multimotor drive describing axial weights, adhesion and wheel diameters.

The models of the mechanical part and control circuits together form the model of the control object, including load and motor electromagnetic models and models of the electronic part of the converter, hardware of the control part and feedbacks. The model should be simple because of energetic, technical and time limitations and limitations of the available computer hardware. But it should include important properties and parameters needed for the control system design.

Using the model of the mechanical part, shown in Fig. 8, and adding the detailed model of the electrical part, gives the opportunity to observe different operation modes and ranges of the system.

These operation ranges, shown in Fig. 9, include the torque ramp, constant torque operation, motor field weakening in constant power operation and maximum speed limitation due to the end of field weakening. The torque ramp limits maximal available motor current and is needed for limiting the acceleration. The start of the motor field-weakening process depends on the maximum available supply voltage for drive systems. Control principles (methods) are an important part for system modelling. Most of modern systems are based on a microprocessor control system, thus a dynamic model should also contain descriptions of software based controllers, control algorithms, torque controllers, speed controllers or reference integrators, ramp-functions, anti-slip systems, control of the field weakening, control of the operation mode and control of the braking.

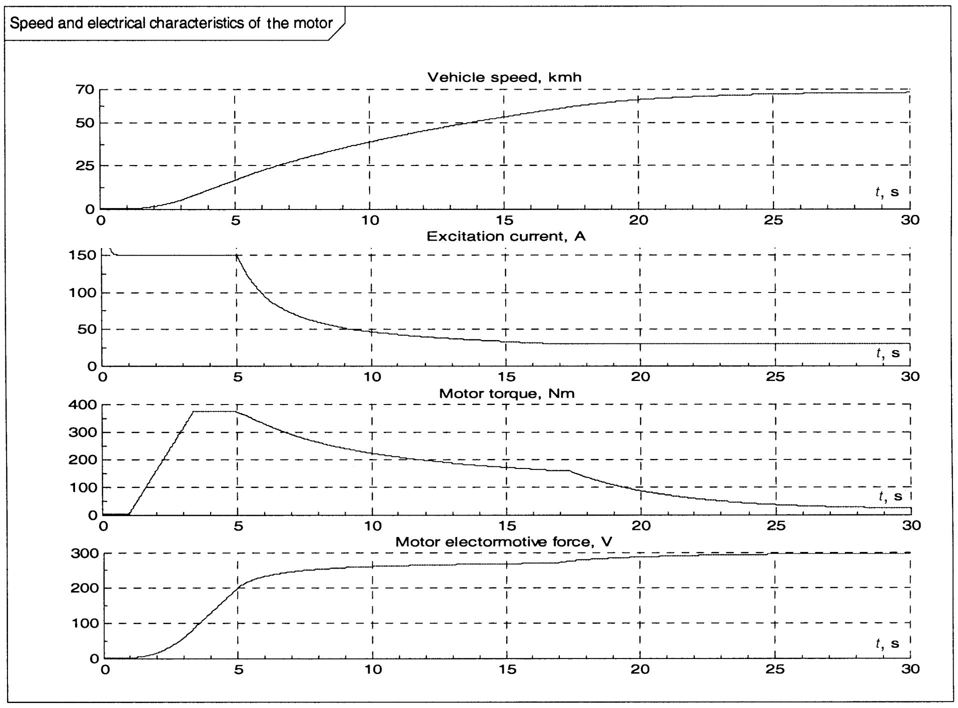

Fig. 9. Acceleration process, calculated using a simulation model.

6. CONCLUSIONS

Conceptual modelling for the design of mechatronic systems is described. The method utilizes new rapidly advancing SysML as the modelling language. The concept consists of models of the requirements, structure and behaviour. An important feature of the approach is instantaneous analysis and simulation link to get the fast response of strengths and weaknesses of the developed mechatronic system design candidate. The described method is bounded and specified for the mobile electric vehicle design. Some examples and metamodels for this case are presented. Design templates for modelling and simulation are required to start a fast product development process. Therefore the implemented methodology relies on predefined templates from the template library. The next step is to widen the template library for different design cases. Together with the development of the design templates, simulation templates (Simulink models) will be designed. Further work lies in the integration of the solution with an automatic mechatronics system development framework, where design concepts can be generated from the requirements diagram in a semiautomatic way.

ACKNOWLEDGEMENT

The work is part of the robotics projects supported by the Estonian Ministry of Science and Education, grant No. 0142506s03.

REFERENCES

1. Hubka, V. and Eder, W. *Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge*. Springer, London, 1996.
2. Pahl, G. and Beitz, W. *Engineering Design – A Systematic Approach*. Springer, Berlin, 1996.
3. Ullman, D. G. *Mechanical Design Process*. McGraw–Hill, New York, 2002.
4. French, M. *Conceptual Design for Engineers*. Springer, London, 1999.
5. *Design Methodology for Mechatronic System – VDI 2206*. Beuth Verlag GmbH, DI, Düsseldorf, 2004.
6. Gurd, A. *Using UMLTM 2.0 to Solve Systems Engineering Problems*. Telelogic, 2003. http://whitepapers.zdnet.co.uk
7. Kukkala, P., Riihimäki, J., Hännikäinen, M., Hämäläinen, T. D. and Kronlöf, K. UML 2.0 Profile for Embedded System Design. In *Proc. Design, Automation and Test in Europe Conference*. Munich, 2005, 710–715.
8. Gawthrop, P. *Metamodelling: for Bond Graphs and Dynamic Systems*. Prentice Hall, London, 1996.
9. Desel, J. and Juhas, G. What is a Petri net? – Informal answers for the informed reader. *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, 2001, **2128**, 1–25.
10. Davoren, J. M. and Nerode, A. Logics for hybrid systems. *Proc. IEEE*, 2000, **88**, 985–1010.
11. Rzevski, G. On conceptual design of intelligent mechatronic system. *Mechatronics*, 2003, **13**, 1029–1044.

12. Seo, K., Fan, Z., Hu, J., Goodman, E. D. and Rosenberg, R. C. Toward a unified and automated design methodology for multi-domain dynamic systems using bond graphs and genetic programming. *Mechatronics*, 2003, **13**, 851–885.
13. Granda, J. J. The role of bond graph modeling and simulation in mechatronics systems. An integrated software tool: CAMP-G, MATLAB–SIMULINK. *Mechatronics*, 2002, **12**, 1271–1295.
14. AMESim: Modeling & simulation environment for systems engineering. http://www.amesim.com
15. Dymola – dynamic modeling laboratory with Modelica (Dynasim AB). http://www.dynasim.com
16. 20-sim, the dynamic modeling and simulation package for iconic diagram, bond graph, block diagram and equation models. http://www.20sim.com
17. System modeling language (SysML) specification. Version 1.0, Draft. OMG document ad/2006-03-01, 2006. http://www.sysml.org
18. Sell, R. and Tamre, M. Integration of V-model and SysML for advanced mechatronics system design. In *Proc. Research & Education on Mechatronics Conference REM05*. Annecy, 2005, 276–280.
19. *Systems Engineering Handbook*. INCOSE-TP-2003-016-02, version 2a. Technical Board of International Council on Systems Engineering (INCOSE), 2004. http://www.incose.org
20. Karnopp, D. C., Margolis, D. L. and Rosenberg, R. C. *System Dynamics – Modeling and Simulation of Mechatronic Systems*. J. Wiley, New Jersey, 2006.
21. Popa, I. S., Popescu, M. O. and Popescu, C. Energetic macroscopic representation applied to an electrical urban transport system. In *The Annals of "Dunarea de Jos", University Of Galati Fascicle*, 2002, **III**, 34–39.
22. Lehtla, M. *Microprocessor Control Systems of Light Rail Vehicle Traction Drives*. Tallinn University of Technology Press, Tallinn, 2006.

Mobiilse elektrisõiduki kontseptuaalse modelleerimise metoodika

Raivo Sell, Mart Tamre, Madis Lehtla ja Argo Rosin

On loodud elektrisõiduki modelleerimise metoodika, mis kasutab modelleerimiskeelena uut, kiiresti arenevat keelt SysML. Metoodika on suunatud kontseptuaalse modelleerimise faasi kiiremale ja efektiivsemale projekteerimisele. Selleks on välja arendatud eeldefineeritud mudelite süsteem, mida arendaja saab andmebaasist valida sõltuvalt süsteemi spetsiifikast. Eeldefineeritud mudelid on nii süsteemi nõuete kui ka lahenduse kirjeldamiseks. Kirjeldatud metoodika abil saab modelleerida süsteemi struktuuri ja käitumist ning siduda erinevaid lahendusvariante simulatsioonimudelitega. Väljatöötatud lahendus võimaldab kiiresti ja efektiivselt alustada mehhatroonikasüsteemi arendusprotsessi ning juba kontseptuaalses faasis simuleerida erinevaid lahendusvariante. Töö on üks osa mehhatroonikasüsteemide projekteerimisega seotud uuringust, mille eesmärk on automatiseerida tootearenduse kontseptuaalset faasi.