INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKHOLM, AUGUST 19-21, 2003

MODELLING INFORMATION FOR MECHATRONIC PRODUCTS

Karl Hallin, Trond Zimmerman, Daniel Svensson, Johan Malmqvist

Abstract

In mechatronic product development, there is a lack of a common understanding among different engineering domains, regarding the design. One step in improving this situation is to develop information models capable of representing both hardware and software elements. The objective of this paper is to investigate the Chromosome Model's ability with respect to the mechatronic product and also to suggest an information model which implements its principles. The approach takes its point of departure on a theoretical level by analyzing theories from mechanical and software engineering. Further, software concepts of relevance are incorporated into the Chromosome Model. The result, referred to as a metamodel, is then transformed to an information model by utilizing STEP AP214. Both the metamodel and the information model are validated with an industrial example. The conclusions drawn from the work are that the Chromosome Model can be used as a metamodel for mechatronic products and that STEP AP214 can be deployed within the scope of the metamodel. The resulting information model captures the functional, causal and spatial relationships of a mechatronic product.

Keywords: Mechatronic product, Chromosome Model, STEP, product information management

1. Introduction

Mechatronic product development is often carried out in isolation with no or poor communication between different engineering domains. Consequently, problems arise in the later stages of the development process, typically during integration. This necessitates redesigns which generates considerable costs. One step in addressing this issue is to develop better information models and processes for managing product information. One alternative for model improvement is to strive for greater integration with respect to the engineering domains in scope. Given an integrated model, activities such as requirement management, configuration management and lifecycle support will be facilitated on a shared level. The raison d'être for an integration is to increase the efficiency in information management and thus receiving fewer erroneous designs discovered in later phases. Ultimately such improvements lead to lower costs, shortened lead times and higher product quality.

This research has two main objectives; firstly, to investigate whether the Chromosome Model [1] is able to describe the concepts of a mechatronic product during development. Secondly, to suggest an information model competent of implementing the principles of the Chromosome Model.

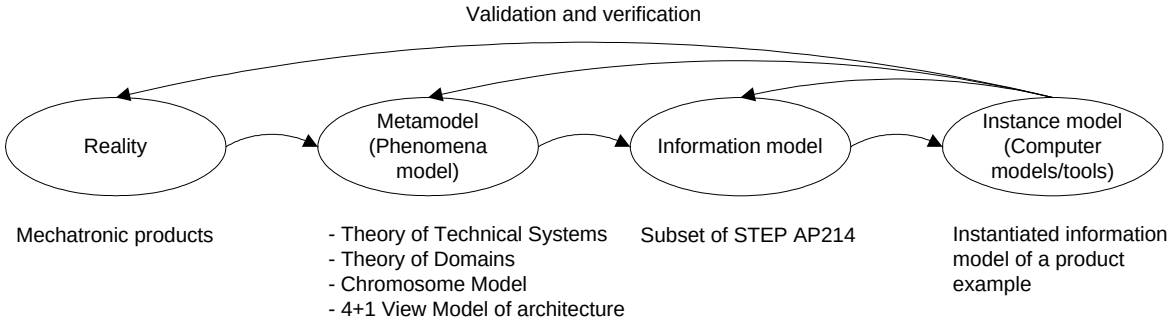

Figure 1. The research approach, adapted from Duffy and Andreasen [2]

The approach is to start from and reason about existing theories from the hardware and software domains, using them as metamodels, see Figure 1.

The hardware domain is represented by the Theory of Technical Systems [3] and the Theory of Domains [4], whereas the software domain is represented by the 4+1 View Model of architecture [5]. The Chromosome Model [1] which has evolved from the Theory of Domains is adopted for representation of the mechatronic product, including software. The resulting metamodel is then formalized into an information model by utilizing the STEP Application Protocol (AP) 214 [6]. The information model and its use represent an integrated approach towards different engineering domains, tracing dependences in the mechatronic product and supporting a coordinated management of product information. Finally, an industrial example of a mobile phone is used to validate the resulting metamodel and information model.

The remainder of this paper is outlined as follows: Section 2 overviews related research work. Section 3 argues the selection of the Chromosome Model as a metamodel for the entire mechatronic product and takes software into account. Section 4 focuses on a description of how software can be introduced into the Chromosome Model. The correspondence between AP214 and the Chromosome Model is analyzed in Section 5, resulting in an information model to continue with. For validation of the somewhat modified Chromosome and to illustrate how the selected subset of AP214 can be used, an instantiation of the product example is presented in Section 6. The results are discussed and concluded in Section 7 and some suggestions for future work proposed in Section 8.

2. Related work

Work that has been performed and presented within the area of engineering design research that includes mechatronic products and product development often has a technical focus. Examples are results of simulation methods and models [7, 8] or solution-modelling of the product itself [9].

An interesting approach is that of Collier [10], who proposes a common product model for hardware and software focusing on information management for the development of product and subsystem variants. His work has a very broad scope but lack in detailing. Persson-Dahlqvist et al. [11] explore the area of PDM and SCM integration, putting emphasis on similarities and differences between computer tools. Even though they raise many interesting questions regarding domain integration, they do not go into information modelling details.

Implementations of the Chromosome Model are rare. Malmqvist and Schachinger [12] have pointed out how the model could be implemented in terms of managing requirements coupled with the other artefacts of the model. The ISO standard STEP AP214 [6] offers a rather complete implementation of the Chromosome Model even though it is unspoken. Utilizing

STEP in design research is not uncommon. The work of Sivard [13] emanates from Axiomatic Design and AP214 with the purpose of improving information management in developing product families. The methodology used to perform and present this work has several similarities with research presented by Aganovic et al. [14] as it uses STEP and the Theory of Domains as its frame of reference, even though it focuses on concurrent engineering and manufacturing systems.

The research presented on the topic of mechatronic products is insufficient regarding product information management. In addition, there exist few attempts to couple information models for the mechatronic product to modern design theory. This paper aims to address this issue.

3. Justification of the Chromosome Model

The aim of this section is to justify the use of the Chromosome Model as a basis for integrated information management for mechatronic products. Even though the artefacts described by the Chromosome and the 4+1 View Model differ in the sense that software cannot be represented in terms of shape and form they have some basic similarities. In this section, the two models will be described and mapped to each other.

3.1 Two models

The Theory of Domains [4] is based on the Theory of Technical Systems [3] and describes a mechanical product by four domains; a process domain, a function domain, an organ domain and a part domain. Processes transform operands (material, energy, signals) due to effects. A function is the ability to create an effect. Effects are produced by organs. Organs are physically realized by parts. These four domains build a metamodel referred to as the Chromosome Model [1]. The domains are causally linked in the model, see Figure 2.

Shaw and Garlan [15] define software architecture as something that *"involves the description of elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on these patterns"*. In software development, these descriptions are often elaborated and communicated through various standards and specifications. The Unified Modeling Language (UML) is the most widely used standard for modelling and building software. UML adapts Kruchten's suggestions [5] of a 4+1 View Model of software architecture. All views in the 4+1 View Model include both structural models and behavioural models. The Logical view states the functional requirements of the system, i.e. what the system should do. It also defines major design packages, subsystems and classes. The Implementation view is typically deduced from the Logical view. It encompasses the organization of subsystems into hierarchy of architectural layers; software parts and files

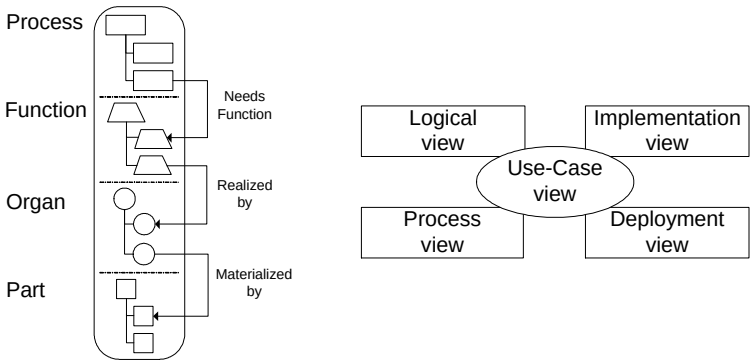

Figure 2. The Chromosome Model and The 4+1 View Model of architecture [5]

that are used to assemble and release the software. The Process view addresses concurrency and synchronization aspects of the system at run time; tasks, threads, processes and their interactions. Issues of parallelism, fault tolerance, object distribution, deadlock and response time are dealt with here. The Deployment view, sometimes referred to as the Physical view, shows how the various executables and other runtime software parts are mapped to the underlying platforms or computing nodes. It also deals with the non-functional requirements set on the software. Finally the Use-Case view has a particular role in the architecture, as it illustrates the four other view models by typical user scenarios. In the early development phases it helps to explore and encompass the architecture. At a later stage, it is used for validation of the system.

3.2 Conceptual similarities between the models

The use of the Chromosome Model as an opening reference for integration requires some understanding for the similarities between the domains in scope. Here, a conceptual mapping from the 4+1 View Model to the Chromosome is proposed.

The Process view: This view deals with aspects of interaction and has time as a variable. It clearly belongs to the process domain.

The Logical view: Since the view treats the definition of what is to be accomplished, it is apparent that it fits into the function domain. The view also maps to he organ domain, since it is used for the definition of fundamental elements, carrying function, within the software structure.

The Implementation view: In terms of its management of software artefacts, it is evident that the mapping is preferably done towards the part domain. Noticeable is that not all aspects of this view can be mapped unambiguously. For instance the view also deals with internal requirements to ease the programming.

The Deployment view: This view assigns software parts to hardware. The view thereby map to the part domain. However, the non-functional requirements do not belong in this domain.

The Use-Case view: The view communicates the actual use of the software. We can see upon it as an abstraction of the Process view. With the same certainty as with the Process view, we state that it belongs to the process domain.

To sum up, there are noticeable conceptual similarities between the two models. Even though the mapping is not completely unambiguous in every aspect, it could be used as a foundation for a further exploration of the Chromosome Model's capabilities regarding software.

4. Using the Chromosome Model for mechatronic products

In this section we propose some additions to the Chromosome Model, extending its capabilities for managing software information. The discussion will have its point of departure in the Theory of Technical Systems.

4.1 Software from a Theory of Technical Systems perspective

From a Theory of Technical System's point of view an operand is transformed in a transformation process by effects from a technical system. In the process, the operand changes state from existing to desired. If for example the process is to boil water, the technical system may be a stove, the operand the water to be boiled and the effect produced by the technical system may be the heat from the stove.

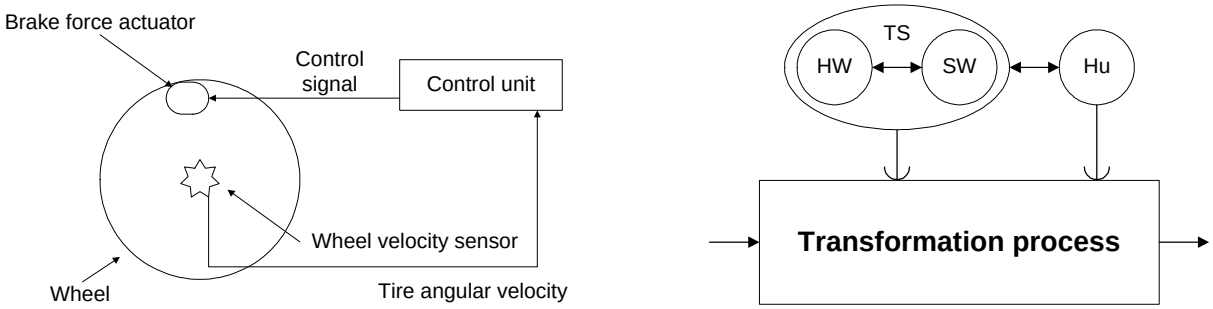

Figure 3. Hardware and software together effects the transformation process

Software can in the terminology of Theory of Technical Systems be described in two different ways. On a high level of abstraction, only the controlling effects of the software are apparent and software can then be described as a part of the technical system creating effects on the transformation process. A further decomposition of the transformation process entails that the software system must actually define the process, giving the process its desired characteristics. Software is then a set of rules or instructions, which prescribe the actions to be taken and which output to produce when given certain input.

The most reasonable approach is to regard software as a part of the technical system. After all, software and hardware together carry the functions of a product. This can be illustrated by giving an example of an Antilock Braking System (ABS), schematically shown in the left part of Figure 3. We can notice how the actuator and the software in the control unit together carry the function *"apply brakes without locking"*. In the right part of the figure we see how the technical system is divided into software and hardware subsystems and how they together effects the transformation process. Of course, software can only have effect on an information level.

4.2 Impact on the Chromosome Model

Due to the fact that software and hardware together create effects on the transformation process, we propose the addition of software as well as mechatronic organs to the Chromosome, see Figure 4. If a functional dependence exists between a hardware organ and a software organ, it is defined by their causal relations to the function that they realize. In the part domain we subsequently propose the addition of software parts. Mechatronic parts may also be necessary in order to manage artefacts that comprise both hardware and software. Dependence between a hardware part and a software part imply that the software is housed in the hardware, or more correctly loaded onto the hardware. This type of dependence can be classified as spatial and is created by including hardware and software parts in the same assembly.

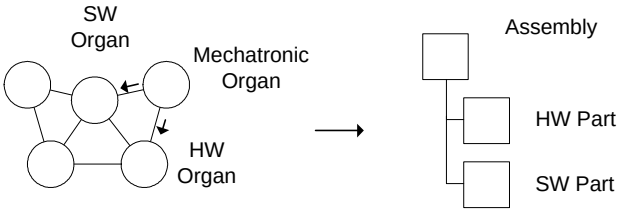

Figure 4. Mechatronic organ materialized by parts

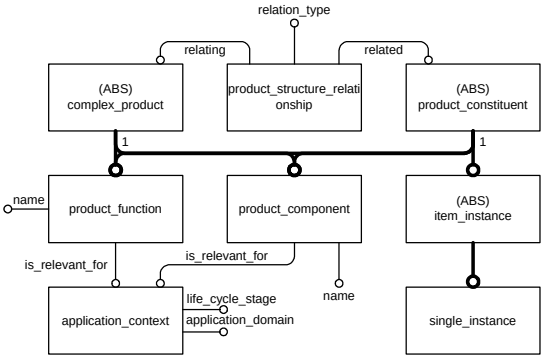

Figure 5. The utilized subset of AP214 [6]

5. STEP-based implementation

The aim of this section is to show how the most essential subset of AP214 implements the principles of the Chromosome Model (including proposals of section 4). STEP is an abbreviation *for STandard for the Exchange of Product model data*, which is a family of ISO standards [16]. The STEP *Application Protocol 214* scopes *Core data for automotive mechanical design processes* [6]. Due to its generality and Chromosome outlook on design AP214 is a good candidate as an information model for mechatronic products.

In AP214, three out of four Chromosome domains have direct correspondence; the function, organ and part domains. The only process domain concepts that are found in the AP regard the manufacturing system and its meeting with the design. The subset of AP214 that has been utilized is illustrated in Figure 5 and described in the subsequent section.

Starting from top left, a *complex_product* can be realized by, decomposed into or specialized as *product_constituent* in a functional, logical, or physical way. A *product_structure_relationship* relates a *complex_product* to a *product_constituent*. Consequently, a *product_constituent* is an object that may participate in the functional, logical, or physical break-down of the product. These three entities represent a very important mechanism for building product structures. A *product_function* is defined as *"a behaviour or an action expected from a product"*. A *product_component* is defined as an element in a conceptual product structure, which is a somewhat vague definition. The possible relationships between *product_function* and *product_component* reveal more of the nature of the two entities. A *product_function* may be considered as the functionality to be fulfilled by a *product_component*, or a *product_component* may be considered as a means to realize a *product_function*. Both *product_function* and *product_component* are specializations of the *complex_product* and *product_constituent* entities. An *item_instance* is an occurrence of an *item* (part) in a product structure (whose semantics are defined elsewhere in the AP). The entity is also a specialization of the *product_constituent* entity. The valid contexts for *product_function* and *product_component* are defined by the entity *application_context* and its attributes *application_domain* and *life_cycle_stage*. *Application_domain* identifies the context of the entity that it is relevant for. The standard suggests amongst others 'electrical design' and 'mechanical design', where applicable. We here propose a small extension including 'mechatronic design' and 'software design' as possible options. The *life_cycle_stage* attribute specifies the stage for which the definition is valid; 'design', 'manufacturing' or 'recycling'. The specialization of the *item_instance* highlighted in this work is the *single_instance* entity.

In summary, the discussed subset of the AP is the most vital part for representing core Chromosome concepts. Three domains are satisfactorily covered by the AP. *Product_function* is the key entity in the function domain, *product_component* in the organ domain and *item_instance* in the part domain.

6. Validation

This section validates the work presented in this paper with product information of a mobile phone, namely the T28 of Ericsson. Firstly, it will be illustrated how the Chromosome model can be used as a basis for addressing the mechatronic product information management issue and secondly, parts of AP214 will be instantiated with product information of the phone.

Ericsson has chosen to describe the T28 by a software architecture and a hardware part structure. These two structures are separate from each other but there is a generally expressed desire for integration. During development the software is divided into modules which are implemented by a set of source code files and managed in a SCM system. The final outcome of the software development process is a so-called *load module*, a software-executable built out of several modules and specially packaged for loading onto the phones memory. The hardware part structure is managed in a PDM system.

The use of the Chromosome is illustrated in Figure 6 by the two T28 high level functions *Set Frequency* and *Send Signal*. Noticeable is that the soft and hard organs together realize their corresponding functions. To realize these functions, three hardware organs and one software organ are deployed. A network communication organ, controlling the transmitter/receiver, a logic organ executing software operations, a transmitter/receiver organ transforming an electrical signal to a radio signal and vice versa and an antenna organ for signal reception and transmission. The hardware organs are realized in the parts structure by a CPU, a transmitter/receiver assembly and an antenna.

The software organ is realized by the load module. The example shows that software modules are possible to treat as software organs and included in a common model together with the hardware of a mechatronic product. A functional dependence between the transmitter/receiver, logic and the network communication organs originates in that they together realize the *Set Frequency* function. A spatial dependence between software and hardware exists due to the fact that the load module is included in the parts structure.

The above discussed functions, organs and parts can be instantiated in the AP214-model. In the centre of Figure 7, the entire instantiation (for details see [17]) of the information presented in Figure 6, and to the left/right a zoom of the two highlighted regions.

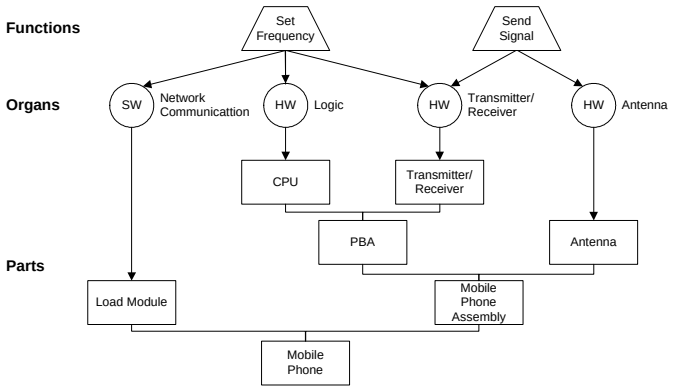

Figure 6. Schematic representation of the mobile phone in Chromosome notation

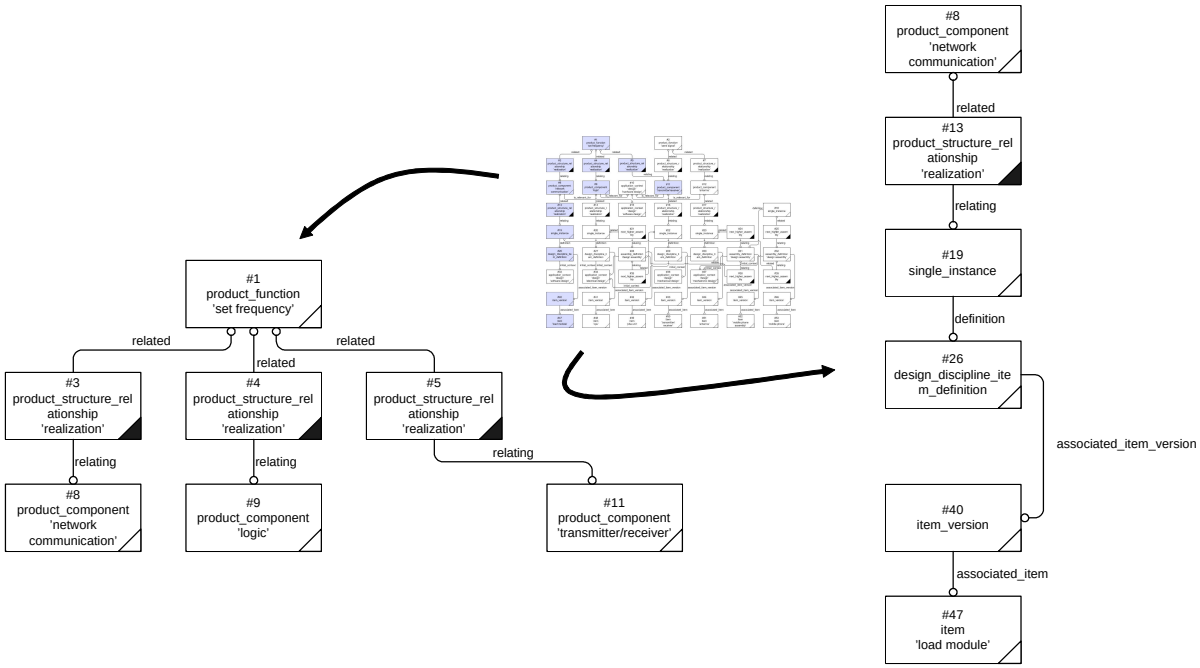

Figure 7. An instantiation of the information model describing the mobile phone example.

The exemplifying functions *set frequency* (#1) and *send signal* are represented by instances of the entity *product_function*. The functions are realized by instances of *product_component*, i.e. the organs, as in the cases of network communication (#8), logic (#9) and transmitter/receiver (#11). This illustrates how AP214 captures the functional organ-organ relations. The realization of the organs into parts is done through instances of the *product_structure_relationship* entity (#13), which point out the specific instances of the *single_instance* (#19) that realize the organs. A *single_instance* is as mentioned in Section 5 only an occurrence of an *item*, the *item* is the actual part that is version controlled by an *item_version* (#40). Notice that the *single_instance* (#19) of load module (#47) realizes the network communication organ (#8).

The instance diagram illustrates that AP214 without any extension is capable of handling the main elements of the mechatronic product: The product can be represented on the functional, organ, and part level. Assemblies comprising mechanical as well as electrical and software parts can be built indicating their relations from a functional or manufacturing viewpoint through the *assembly_definition* construct (excluded in Figure 7). Where applicable, a PDM-system implementing AP214's information model could act as the only information holder at the system level. But it is also possible to let other information systems such as SCM systems take control at appropriate points. This will most probably be the case for the source code files and the building of the load module.

7. Conclusions

This paper has approached the area of product information management for mechatronic products with two objectives as stated in Section 1. In this section the work will be discussed and concluded in accordance with these objectives.

Objective 1: *Investigate whether the Chromosome Model is able to describe the concepts of a mechatronic product during development.*

The Chromosome Model was originally designed to clarify the concepts of mechanical engineering artefacts (machine systems). However, the number of pure mechanical products being developed today is decreasing, due to an increased use of software and electronics. Thus a further elaboration of the Chromosome Model is desired. This paper has showed that software can be viewed as a part of the technical system and that it together with hardware produces effects on the transformation process, at least on an information level. With this in mind it becomes clear that the concepts of mechatronic products can be explained in the terminology of the Chromosome. We have seen that function carriers can be of various types and that it is of no significance for the final product if they are implemented in hardware or software, it is just a matter of means for reaching the goal of the design. For example both a slide-rule and an electronic calculator implement the functionality of addition.

This paper has focused on conceptual integration, delimiting from details of what information that should be managed in what supporting system. Despite this fact, it can be concluded that the Chromosome, with modest modifications, is possible to deploy in the efforts to consistently describe the concepts of a modern mechatronic product.

Objective 2: *Suggest an information model competent of implementing the principles of the Chromosome Model.*

In this paper it has been shown that the STEP AP214 is a suitable information model for representing mechatronic product information. There are several benefits of a standardized approach towards information management, implementing already agreed solutions and conventions. The use of standardized formats eases access and increases the ability to exchange information, which is certainly not the case of a non-standard alternative. Even though the STEP Application protocol 214 was developed for and by the automotive industry and without formulated intents of implementing the Chromosome Model, we have found significant similarities as shown in section 5 and 6.

8. Future Work

This paper has focused on integration on a shared level. Further research is needed to clarify the details of how information management in software and hardware engineering can be linked. Topics such as requirement management, versioning and lifecycle issues still have to be addressed for the mechatronic product. Finally, there are more than two engineering domains (electronics, mechanics, software, hydraulics et cetera), which means that there is a need for addressing the general problem of multi-domain products.

9. Acknowledgements

This work was financed by Vinnova (Swedish Agency for Innovation Systems) and Ericsson, whose support is gratefully acknowledged. We wish to thank Sony Ericsson Mobile Communications AB in Lund for providing the validation example used.

References

[1] Andreasen, M.M., "Designing on a "Designers Workbench" (DWB)", Proceedings of 9th WDK Workshop, Rigi, 1992.

[2] Duffy, A.H. and Andreasen, M.M., "Enhancing the evolution of design science", Proceedings of International conference on engineering design - ICED 93, 1995, pp. 29-35.

[3] Hubka, V. and Eder, W.E., "Theory of technical systems : A total concept theory for engineering design", Springer-Verlag, 1988.
[4] Andreasen, M.M., "The theory of domains", Proceedings of EDC-workshop: Understanding Function and Function to Form Evolution, Cambridge, United Kingdom, 1991, pp. 21-47.
[5] Kruchten, P., "The 4+1 View Model of Architecture", IEEE Software, Vol. 6, 1995, pp. 42-50.
[6] ISO, "ISO 10303-214:2001(E) - Industrial automation systems and integration - Product data representation and exchange - Part 214: Application protocol: Core data for automotive mechanical design processes", 2001.
[7] Krastel, M. and Anderl, R., "Managing mechatronic simulation models of technical products with PDM-systems", Proceedings of International conference on Engineering Design - ICED01, Glasgow, 2001.
[8] Zhang, W.J. and Li, Q., "Design for control: A proposed methodology for developing mechatronic systems", Proceedings of International conference on Engineering Design - ICED 2001, Munich, 1999.
[9] Gausemeier, J., Flath, M. and Möhringer, S., "Modelling and evaluation of principle solutions of mechatronic systems, exemplified by tyre pressure control in automotive systems", Proceedings of International conference on Engineering Design - ICED 2001, Glasgow, 2001.
[10] Collier, W., "A Common Specification for Systems-Based Product Modeling", D.H Brown Associatives, Inc., 1999.
[11] Persson-Dahlqvist, A., Asklund, U., Crnkovic, I., Hedin, A., Larsson, M., Ranby, J. and Svensson, D., "Product Data Management and Software Configuration Management - Similarities and Differences", The Association of Swedish Engineering Industries, 2001.
[12] Malmqvist, J. and Schachinger, P., "Towards an implementation of the chromosome model - focusing the design specification", Proceedings of International conference on engineering design - ICED 97, Tampere, 1997, pp. 203-212.
[13] Sivard, G., "A Generic Information Platform for Product Families", Doctoral thesis, Royal Institute of Technology, Division of Computer Systems for Design and Manufacturing, Department of Production Engineering, Stockholm, 2000.
[14] Aganovic, D., Nielsen, J., Fagerström, J., Clausson, L. and Falkman, P., "A concurrent engineering information model based on the STEP standard and the theory of domains", Proceedings of International design conference - DESIGN 2002, Dubrovnik, 2002.
[15] Shaw, M. and Garlan, D., "Software Architecture", Prentice-Hall, Inc., 1996.
[16] ISO, "ISO 10303-1:19994 - Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and fundamental principles", 1994.
[17] Svensson, D., Hallin, K., Zimmerman, T. and Malmqvist, J., ”An information model for mechatronic products focusing on early development phases”, Proceedings of the 6th Workshop on Product Structuring, Copenhagen, 2003.

For more information please contact:

Karl Hallin Chalmers University of Technology, PPD SE - 412 96 Göteborg Sweden
Tel: Int +46 31 7728285 Fax: Int 46 31 7721375 E-mail: karl.hallin@me.chalmers.se
URL: http://www.ppd.chalmers.se/~hallin/