INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED15
27-30 JULY 2015, POLITECNICO DI MILANO, ITALY

INDUSTRIAL APPLICATION OF A MECHATRONIC FRAMEWORK

Torry-Smith, Jonas Mørkeberg (1); Mortensen, Niels Henrik (2); Ploug, Ole (4); Achiche, Sofiane (3)
1: Novo Nordisk A/S, Denmark; 2: Technical University of Denmark, Denmark; 3: Ecole Polytechnique de Montréal, Canada; 4: R&D Manager, Denmark

Abstract

Design of mechatronics is greatly challenging due to its multi-disciplinary nature. On one hand companies need to facilitate integration between engineering domains while they on the other hand need to manage dependencies created between the exact same domains as the development process progresses. This paper proposes a way to identify, model and clarify dependencies by use of a "Mechatronic Integration Concept". The usefulness of the Mechatronic Integration Concept has been tested in an industrial development project showing positive results of shortening the lead-time, minimizing rework and increasing the performance of the product. The literature review on the topic of dependency modelling in mechatronic products seems to reveal a lack of proposals for how to represent the dependencies graphically which can facilitate cross-domain discussions.

Keywords: Mechatronics, Dependency modeling, Design of mechatronics

Contact:
Dr. Jonas Mørkeberg Torry-Smith
Novo Nordisk A/S
Device R&D
Denmark
j.torrysmith@gmail.com

1 INTRODUCTION

Development of mechatronics is a challenging task to undertake. Nonetheless, the necessity of combining solutions from the areas of mechanical, electronics and software engineering is often what drives companies to accept a higher level of complexity in the development process. Some of the challenges companies encounter include: undesirable and unexpected dependencies are discovered too late in the design process (D'Amelio and Tomiyama 2007), product properties are dispersed onto different modules handled by different engineering disciplines without proper control and tracking (Torry-Smith et al. 2012). The nature of the development within the involved engineering disciplines is different causing synchronization problems between the domains in terms of deliverables. The research work presented in this paper aims at providing a means for modelling and representing a mechatronic concept in which dependencies between domains can be modelled and clarified and hence mitigate the above-cited challenges.

We assume that in companies the process of modelling a 'mechatronic integration concept' along with modelling important dependencies will focus the attention of the designers toward integration issues. To be able to identify and clarify dependencies, a shared understanding of the design is needed. Many good suggestions from researchers can be found on how to describe a mechatronic concept. Some suggestions aim at describing the concept seen from a holistic point of view (Buur 1990), whereas other suggestions are aimed at specific applications such as control engineering (Gausemeier et al. 2009a). In this paper, we are interested in how a mechatronic concept should be modelled when focusing on the integration issue between the domains. The research question we are trying to answer is: how can we elucidate and clarify dependencies in a mechatronic concept?

The *mechatronic Integration Concept* is a graphical modelling of the product concept and is not an automated generated description of the concept based on data structure extracted from the different engineering domains as known from 'transformation modelling' and 'meta modelling'. The intention is to use it as a graphical tool during the development process to support the engineers in the design process. Within this aspect it is just like any other graphical description needed in projects to create the desired overview to make good decisions in projects.

An example of a dependency could be that proper EMC shielding of electronic components is typical performed by mechanical engineers when the chassis of the product is designed. While for example the mechanical engineers design cooling of the product they have to keep in mind that the air inlet and outlet have to be designed so electromagnetic noise cannot enter the product. Our concept of dependencies in mechatronic development is fully described in section 4 of this paper.

The outline of the paper is as follows: first the research method is described in section 2 followed by a description of related work in section 3; the content of the Mechatronic Integration Concept is explained in section 4 and the results of testing the concept in an industrial setting are reported on in section 5; conclusion is found in section 6.

2 RESEARCH METHOD

Three mechatronic development projects aimed at the consumer product segment were investigated and documented in prior work by the authors (Torry-Smith et al. 2013); we observed and recorded trends of how the mechatronic product concepts were modelled to facilitate cross-domain collaboration. The second step was to evaluate concordance between descriptions used in the investigated projects and theory within design of mechatronic products. A detailed description of types of dependencies central to mechatronics projects was obtained via the aforementioned research by the authors. The concept description and the overview of dependencies are then combined into a proposal for how to clarify and model dependencies in mechatronic concepts. Related work is reported on to position the research to the literature. The proposal is then tested in a context of an industrial project and the results evaluated in terms of its usefulness and impact on the design activities.

3 RELATED WORK

The aim of the section is to report on prior work, in which dependencies are identified, grouped and modelled in mechatronic design. The search for related work is performed by focusing on two aspects of design of mechatronics. 1) How to describe a mechatronic product concept and 2) How to describe and model dependencies found in mechatronics projects?

Many researchers propose different frameworks for mechatronics design. Buur (1990) proposes a theoretical framework for design of mechatronics, thereby extending the Domain Theory by Andreasen (1980). Within this framework he suggests to model interface 'organs' defining the boundary between the domains, i.e. dependencies. However, further definition of these dependencies is not stated. Other modelling suggestions based on functional reasoning include Contact and Channel Approach by Albers et al. (2011) and FDF by Nagel et al. (2008). Though dependencies can be modelled using these models they are limited to describing the flow of energy, material and signal between functions or functional carriers.
Gausemeier et al. (2009a) focus on control engineering when proposing a definition of a mechatronic concept. A number of views are suggested to be modelled and one of these views is called 'active structure' and relates to the control issue describing exchanged signals and energy between components. Even though interfaces (Mohringer and Gausemeier 2002) and consistency management is described (Gausemeier et al. 2007) this research is on how a holistic model can link to domain-specific models but it is not providing an overview of what to types of dependencies should be linked. The work by Gausemeier is related to research on transformation models and formal description languages. Alternative formal modelling languages include SysML (Object Management Group 2010), UML (Group 2011) and IDEF (Integration Definition Methods 2012) and AM-tool (Cabrera et al. 2011). Common for the formal languages is that they provide the possibility to model aspects of a mechatronic concept as well as dependencies due to the flexibility of the semantics provided. Thus, no systematic reporting on types of dependencies is found. Research work on transformation models presents the possibility to have one meta-model which shares parameters with domain-specific modelling to maintain consistency across domains (Gausemeier et al. 2009b; Wynn et al. 2009; Shah et al. 2010). This provides a framework for managing dependencies but does not define any dependencies to model. Instead of using a meta-model for consistency checking Hehenberger (Hehenberger et al. 2010) suggests to perform automatic consistency checking when design parameters are changed. It provides some advantages but it requires the concept to be describes via a formal model (e.g. a SysML model). This work is not aimed at providing overviews of dependencies but to suggest a method for automatically handling and checking inconsistencies (which can be interpreted as dependencies).
An alternative to formal modelling and use of meta-models is the use of informal modelling of mechatronic concepts such as A3 architecture overviews (Borches and Bonnema 2010) and sketching techniques by e.g. Buur (Buur 1990). These methods provide flexibility in describing both the concept and the dependencies. While the flexibility is the strength of these methods it is also the weakness. Only general descriptions of the content of the models are provided, thereby not touching upon descriptions of dependencies and how to model them in the concept.
Design Structure Matrix (DSM) is directly aimed at modelling dependencies in products (Felgen et al. 2005). DSM is characterized by the ability of comparing two-of-a-kind, e.g. describing which components is physically interfacing other components. Domain Mapping Matrices (DMM) (Danilovic and Browning 2007) are also aimed at describing dependencies but are capable of comparing two different entities, which could be e.g. relating functions to components. The analysed dependencies related to the product are limited to the relations between functions and components only. A further distinction between types of dependencies is not found within the methodology. The advantage of using DSM is the possibility to apply algorithms to rearrange rows and columns. Even though applicable algorithms are far more limited when using DMM, matrix representation of the dependencies is a fundamental prerequisite. Due to the focus on matrix representation, graphical representations or other visual models of the concept are lacking.
The purpose of this review was to find related work on how to identify, group and model dependencies in mechatronic concepts with the aim of facilitating a better integration between the domains. The tools and methods presented above are primarily aimed at either describing a mechatronic concept or modelling generic dependencies such as 'component-component' relationship. To the authors' knowledge, a combination of the two aspects and a more in-depth description of types of dependencies to look for in mechatronic projects revealing critical issues do not seem to be covered in prior work.

4 THE DESCRIPTION OF THE MECHATRONIC INTEGRATION CONCEPT

From investigating three mechatronic projects from Danish industry we observe that three views are predominantly used for cross-disciplinary integration meetings: (1) A common conceptual descriptions based on a functional description; (2) a view dealing with spatial relations between components, physical forces and other physical effects; and (3) a view dealing with signal processing and data processing. The views are labelled the M/E/Sw-view the M/E-view and the E/Sw-view respectively. Having established the views, the literature is consulted. It appears that these views are in concordance with mechatronics theory presented by Jansen (2007) and Tomiyama et al. (2007). Tomiyama states that a view bridging two domains can only be possible if the two domains share the same axioms. To exemplify this statement the M/E view is possible to model because they share axioms when viewing the system with regard to spatial relations or physical effects such as forces. Software does not have axioms tied to spatial relations and, hence, cannot be modelled in that view. Table 1 is showing the content to be modelled for each of the three views obtained from investigating the three cases. Since the three views are central to create cross-domain discussions we propose that they be a part of the *Mechatronic Integration Concept*.

Table 1: The content of the M/E/Sw-view, the M/E-view and the E/Sw view

M/E/Sw view – Functional description	M/E view – Physical structure and spatial arrangement	E/Sw view – Data structure and signal processing
- Aspects to cover in the M/E/SW view - Task analysis for life phases - Functions and function carriers - Sequence of the functions	- Aspects to cover in the M/E view - Spatial configuration - Connectivity between components - Force and physical effects	- Aspects to cover in the E/SW view - Data and signal flow - Data structure (architecture) - Timing and sequencing

In addition to the three views presented in Table 1, we propose to include an overview of important dependencies identified in the product concept, so that the *Mechatronic Integration Concept* will be composed of these four descriptions. In the following, a description of generic dependencies is presented. From previous work, we have classified a number of dependencies, which can be grouped according to *Systems Theory* (Hubka and Eder 1988) which is equivalent to Theory of Domains by Andreasen (1980). In the framework proposed by Andreasen the functions and properties of a product constitute the behaviour of the product. Means is what realizes the behaviour of the product. In mechanical engineering the physical structure realizes the behaviour, in electronics the electronics is realising the behaviour and within software engineering the software code is creating the behaviour of the products.

Based on Theory of Domains the following dependency groups can be established (Torry-Smith et al. 2013):

- *A Function-Function dependency:* A dependency between two *functions* is described by the link that is created when a *function* reacts to a stimulus created by another *function*.
- *A Means-Means dependency:* A dependency between two *means* in the product.
- *A Function-Means dependency:* A *function* is realized by a *means* and a *means* can be further decomposed into *sub-functions*, which creates the dependency between *functions* and *means*.
- *A Property-Means dependency: Properties* are realized by *means*, thereby creating dependencies between *means* and *properties*.

An example of a dependency between *means* is physical interface between two components. The relations *Function-Property* and *Property-Property* cannot be found as direct relations in products because the relation will have to go through a *means*. An evaluation of a *property* (e.g. robustness) will include an evaluation of the *means* to which the *property* belongs. This causes the *means* to be included in the relation thereby making it impossible to observe a direct relation between two *properties* or between a *property* and a *function*.

In previous work by the authors (Torry-Smith et al. 2013) the generic categories presented above can be further classified into 13 groups of dependencies specifically directed at mechatronic products. The classification into the 13 groups has the advantage of offering a guideline to what to look for and hence making the identification of central dependencies easier to perform in projects. The identified groups of dependencies are described in Table 2.

Table 2: Overview of the classification of dependencies into 13 groups specific to mechatronic development

Categories	Id #	Name of dependency	Description of the dependency
Fu-Fu	1	Causal function	The dependency between *functions* when the functionality of the product is considered as a process flow
	2	State/time function	Dynamic dependencies between *functions*, in which the sequence and the timing is important.
	3	Sync function	Appears when functional states shift within a very short time span and synchronisation in all domains is needed.
	4	Response function	*Functions* react on stimuli from other *functions*. The size and type of the stimuli have to be matched between the *functions*.
Fu-M	5	Fu-M disposition	Proposing *means* to *functions* in one domain will often have consequences in other domains in terms of supporting functionality.
	6	Cumulative Fu-M	The realization of a *function* may require *means* from various disciplines.
	7	Adverse effect	A *means* may have an adverse effect associated to it. The undesired adverse effect can be formulated as a function (e.g. 'create vibration').
Pr-M	8	Property scheme	The realization of a *property* may be distributed on several components designed by different engineering disciplines.
M-M	9	Multi-disciplinary means	Some *means* have to satisfy boundary conditions (e.g. requirements), which are important to more than one engineering discipline.
	10	Volume allocation	Physical *means* have to be located spatially in the product and the volume may have changing restrictions during the life phases.
	11	Liveliness	The flow of information between electronics and software must be designed without causing a system-lock.
	12	Physical interface	Physical interfaces between modules and components have stakeholders from electronics and mechanical engineering.
	13	Communication interface	Communication between components whether they are analogue or digital is a dependency between electronics and software engineering.

A description comprising the three views (M/E/Sw-view, M/E-view and E/Sw-view) and the overview of dependencies constitute the *Mechatronic Integration Concept*. A modelled example of the *Mechatronic Integration Concept*, which was used in the case study, is shown in section 5. By combining the three modelled views and the overview of product-related dependencies we are able to make a description of the product concept bridging the domains. The intention is that the team members and the project manager will actively use the *Mechatronic Integration Concept* as a central tool during the development process to achieve integration. It is a graphical modelling tool and hence the views are not connected via a data structure to make it agile, easy to set up. Furthermore it does not require computer systems within different domains to be able to exchange data which is a considerable obstacle for companies. The modelling and the focus on dependencies will be the integration catalyst revealing potential challenges in the projects before they become critical problems as well as promoting synergistic solution-finding between the domains.

5 INDUSTRY APPLICATION - 'THE STRONG HAND' CASE

The usefulness and the results of applying the *Mechatronic Integration Concept* in an industrial project are described in this section. First the project is briefly described. Then we present how we modelled the *Mechatronic Integration Concept*. Lastly two examples are used to illustrate how dependencies were treated in the project and the effects of being able to model and clarify them. The modelling of one of these dependencies is shown in the first example.
The project is aimed at developing an actuated hand for patients with severe arthritis. The mechatronic hand can be fitted inside their own palm to help provide an enhanced grip. The aim is to make the device appear discrete when worn at home or in public places. The computer rendering in Figure 1a served as visualization of 'the finished product' at the beginning of the project.

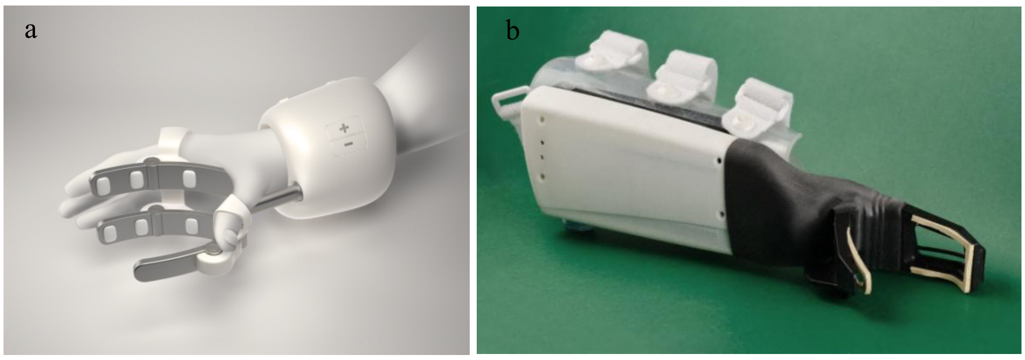

Figure 1: a) Computer rendering of the product concept b) The functional model of the product concept (fully operational)

The project set-up was a joint venture between several companies comprising engineers representing the mechanical, the electronics and the software domain. The project team also included user experience experts, industrial designers and a board of practitioners (arthritis specialists) in addition to the project management group.
The *Mechatronic Integration Concept* was deployed in the conceptual phase where the overall functionality had been determined and solutions in terms of suggestions for technology building blocks have been proposed. Thus the *Mechatronic Integration Concept* was used as a tool during the development process to support the engineers in the design process. Only a rough modelling of the concept has been performed in CAD showing an outline of the subassemblies. A suggestion of the *Man Machine Interface* was proposed but not tested by users so far. Figure 2 is a sketch of the concept (document from the project) illustrating the clarification level. Figure 1b shows the functional model three months later which is fully operational. The project being half-way through the concept development phase creates a purposeful option to test the usefulness of modelling and using the *Mechatronic Integration Concept*.

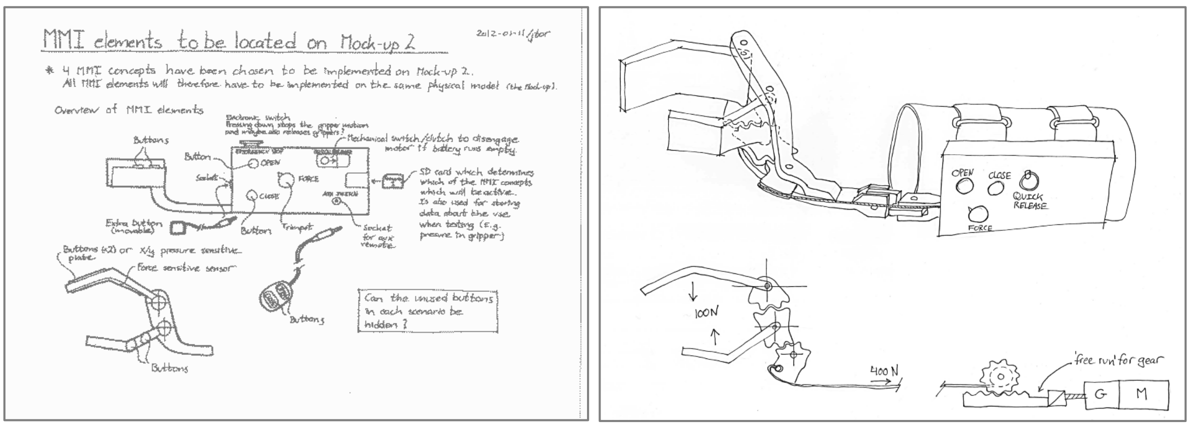
MMI elements to be located on Mock-up 2
2012-01-11/jbor
* 4 MMI concepts have been chosen to be implemented on Mock-up 2.
All MMI elements will therefore have to be implemented on the same physical model (the Mock-up).
Overview of MMI elements
Electronic switch
Pressing down stops the gripper motion and maybe also releases grippers?
Mechanical switch/clutch to disengage motor if battery runs empty.
Buttons
Button
Socket
OPEN
FORCE
CLOSE
SD card which determines which of the MMI concepts which will be active. Is also used for storing data about the use when testing (E.g. pressure in gripper)
Extra button (movable)
Button
Trimpot
Socket for aux remote
Buttons (x2) or X/y pressure sensitive plate
Force sensitive sensor
Buttons
Can the unused buttons in each scenario be hidden?
Buttons
OPEN
CLOSE
QUICK RELEASE
FORCE
100N
400 N
'free run' for gear
G
M

Figure 2: Sketches of the product concept

By using the classification of dependencies from Table 2 we are able to identify 54 central dependencies in the product concept, which have to be addressed in the project. These dependencies were identified and gathered during the conceptual design phase as part of the normal activities in the project. Team members were asked to note and report if they came across dependencies in the course of the project. To facilitate the clarification and further handling of the dependencies the *Mechatronic Integration Concept* is modelled (see Figure 3).

In the following the modelling is described. The functionality of the product is modelled via a task flow analysis in which the technical process including the involved sensors and actuators are described for each step. This description is broken down into two functional descriptions: 1) A functional description where *functions* are related to the principle solutions and 2) A functional overview describing which functionality is active depending on the state of the product. Based on the functional description the M/E-view and the E/Sw-view are created. The M/E-view contains descriptions of the spatial arrangements of the modules and components in the product as well as central physical effects considerations, which are force calculations for the electro-mechanical transmission. In the E/Sw-view we choose to model the data and signal flow in a Data Flow Diagram. In addition we model the data architecture by defining the hierarchy and interaction between main modules of the software.

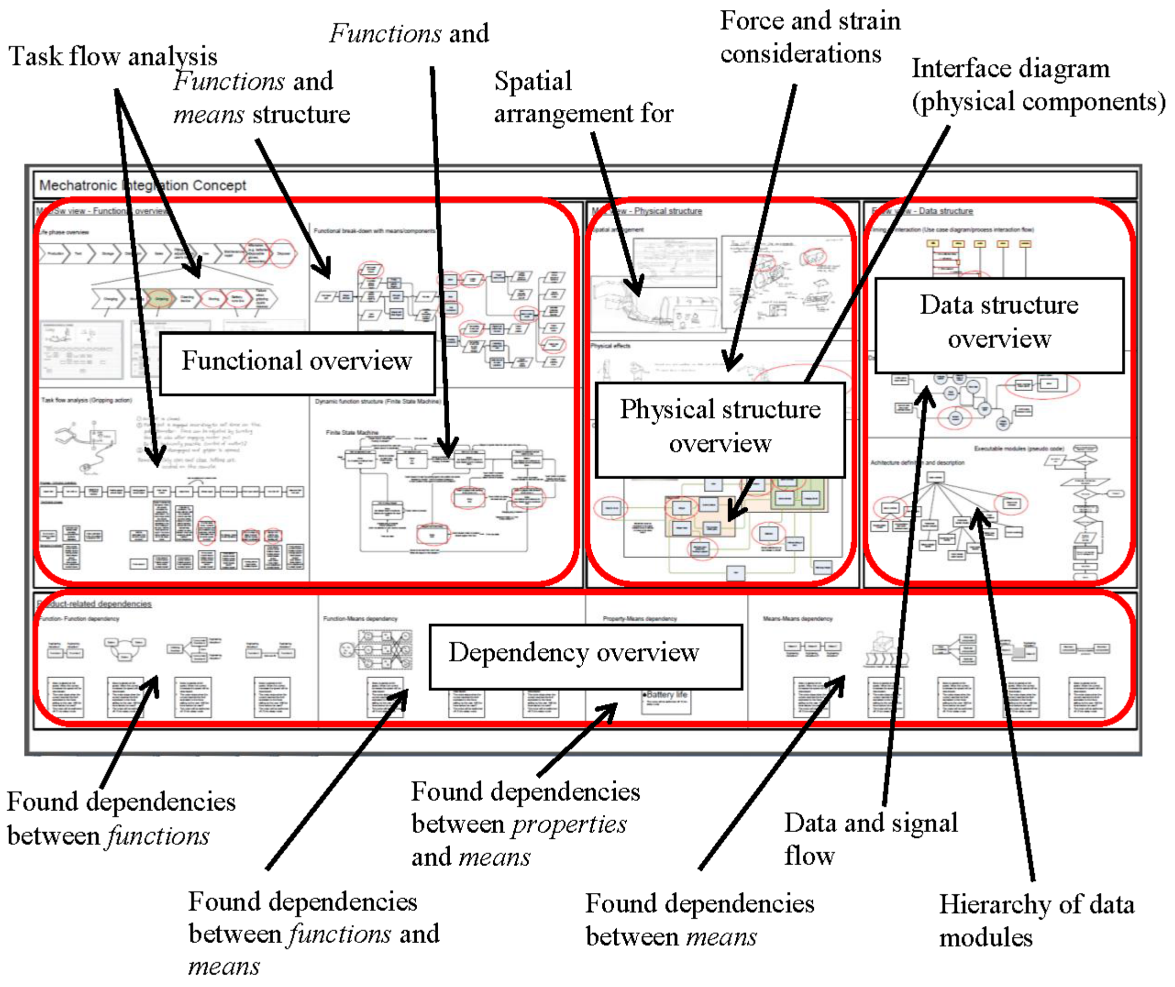
Task flow analysis
Functions and *means* structure
Functions and
Spatial arrangement for
Force and strain considerations
Interface diagram (physical components)
Mechatronic Integration Concept
Functional overview
Physical structure overview
Data structure overview
Dependency overview
Found dependencies between *functions*
Found dependencies between *functions* and *means*
Found dependencies between *properties* and *means*
Found dependencies between *means*
Data and signal flow
Hierarchy of data modules

Figure 3: The content of the Mechatronic Integration Concept

The three views are not linked in a data structure but draw upon normally used graphical models used in each of the domains, e.g. 'Function/Means Tree' (Andreasen 1980) or 'Use cases' (Object management Group 2010). This property of the *Mechatronic Integration Concept* has been carefully considered in order to create a tool which is easy to setup for companies and requires only little expertise to create and maintain. The found dependencies are grouped according to the 13 categories, and the four main categories from Table 2. There are also dependencies within each engineering discipline such as an interface between two mechanical parts. However, these are not modelled since the scope is the cross-disciplinary dependencies. The dependencies may be identified as short statements, but it is likely that they are not complete or fully clarified. As each dependency reaches across at least two engineering disciplines, the presence of representatives from each discipline is needed to achieve a clarification. Hence, it allows us to check if assumptions made by the engineers regarding each dependency are correct.

We use the poster as presented in Figure 3 to facilitate the discussion among representatives from each domain and to keep track of the created explicit knowledge. We have selected two dependencies to exemplify the potential of clarifying the product-related dependencies, which will be presented in the following. The examples have been selected based on the criteria to be fairly explicable while still showing the complexity of the dependencies. In addition, an example is given in the section labelled I for how to model a dependency.

I. This example illustrates a dependency linked to the battery life-time. The '*battery life-time'* is a *property* identified to have contributing elements in the mechanical, electronics and software domain. Due to the link between the *means* contributing to the *property*, the dependency is of the type: Property scheme dependency (according to Table 2). The device is powered by a battery and *battery life-time* is therefore a central concern. Firstly, the obvious components affecting the *battery life-time* is identified: battery size (capacity and technology); power consumption of motor and power for the electronics. When these are broken down further and new aspects are discovered, the picture is far from simple. The stiffness of the structure and the mechanical advantage of the system play an important role. The type of battery technology (e.g. polymer-Ion) is linked to the capacity and the capacity can vary over time as a consequence of how the charging is performed and monitored. In the software domain sleep modes can be introduced but should be carefully designed to allow for monitoring of user inputs in-between sleep modes. These are just some of the *means* which influence the *battery life-time*. The modelling process begins by identifying and highlighting the contributing elements. This is simply done by circling the elements in all three views which will or may influence the *property* (see Figure 4). One by one the influence of the elements is discussed between the involved stakeholders and thereby clarified at a meeting where representatives from each domain are present. The task for the involved engineers is to figure out how the target specification can be met and what *means* to optimize to use the allocated resources most ideal not to inflict the development time negatively.

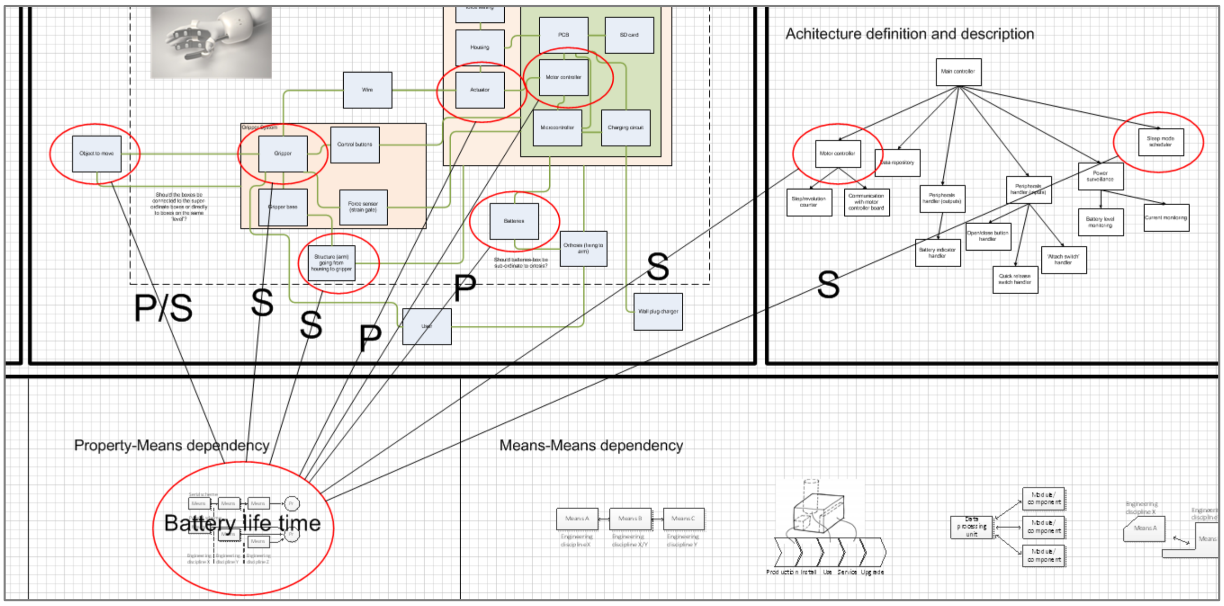
Achitecture definition and description
Property-Means dependency
Means-Means dependency
Battery life time
P/S
S
S
P
P
S
S

Figure 4: Modelling the dependency linked to the property 'battery life-time' (Section of the Mechatronic Integration Concept)

Means can contribute in either a serial or in a parallel manner. The physical size of the battery will have the character of contributing in a serial manner, meaning that the size of the battery is almost proportional with the capacity, which will extend the *battery life-time* proportionally. Sleep modes will have an effect on the drain of the battery but the effect is not as straight forward as the battery capacity and will e.g. be influenced by the ratio between the device gripping and the device being inactive. During the clarification process the type of contribution scheme (parallel or serial) is marked in the

views as can be seen in Figure 4). Having modelled the dependency in detail, decisions on how to treat it further in the project can be made on a sound basis. If the relation between the contributing factors is not clarified, the risk of sub-optimization is immanent and resources will be wasted as well as unnecessary design changes being made in the name of ‘optimization’ (perform the optimization blindly). Resources allocated to a project are always limited and therefore it is very important that we use them in the best possible way.
II. The second example is about how to adjust the gripping force after the user has gripped an object. To do that the electronics and software measure the current in the DC motor. This measurement is an indirect measurement of the gripping force. Prior to gripping, the user can adjust the gripping force by turning a knob on the device. If the user experience the grip is not firm enough after gripping the user can increase it. If the force has to be increased the motor is activated until a higher threshold limit for the current has been reached. Adjusting the force up or down after gripping represents a dependency since solutions from every domain is required to realize the functionality (Fu-M cumulative dependency). The effort has to be coordinated and the dependency has to be understood by all involved domains. When discussing this functionality of the device, a concern arose: increasing the force would most likely be feasible but decreasing the force would be a problem. The reason being that the current will not reflect (be proportional to) the gripping force when decreasing the force. Due to the gripping in action there is a pull in the actuator. Thereby the actuator can reduce the gripping force by a very small reverse current since the tension in the system is helping the movements of the system decreasing the gripping force. After having revealed the dependency the team could choose between two options: i) remove the functionality of reducing the gripping force after gripping, ii) solve the functionality by use of other means. One possibility, which was discussed among the engineers was to count the pulses to the motor when decreasing the force and then based on experiments assess how many steps the motor should reverse to obtain a certain decrease in gripping force. It was decided to put this function ‘on hold’, and wait for the test of the demonstration model to evaluate if the strategy of ‘counting steps’ would be sufficient to control the decrease in gripping force. By revealing the dependencies at an early stage, the team or the project manager is able to make the decision up front of removing the functionality or allocate resources to find an alternative solution. The result is improved project planning, better use of resources and enhanced monitoring of the predicted performance of the product.
The *Mechatronic Integration Concept* was actively used in the development to support the engineers in the design process and it aided in clarifying the dependencies. The application in the industrial setting showed that it is possible to model the dependencies and that the *Mechatronic Integration Concept* can facilitate a cross-disciplinary discussion. The result from applying the concept in the project was a potential cut in the lead-time, increased efficiency of resources used thereby pointing in the direction of being able to help reduce costs in a development project.

6 CONCLUSION

This article introduces the *Mechatronic Integration Concept* and demonstrates the use of it in an industrial setting. It is a graphical model representing the product concept without an underlying data structure to auto-generate it. It is a graphical overview which is equal to other essential graphical tools such as sketching, drawing and modelling concepts or conceptual aspects throughout the development process (e.g. visualisation of a potential product-service system). The *Mechatronic Integration Concept* ensures a common and shared understanding of the product concept with its inherent dependencies. The structured cross-domain clarification of the dependencies enables the team to resolve integration issues early on in the project, which has many positive effects on the product development process. When applying the *Mechatronic Integration Concept* in an industrial project and using it actively in the development process we observed strong indications of effects comprising: reduced lead-time and better utilization of resources due to avoidance of re-work as well as increased performance of the product. The benefit for companies in the long run from having the dependencies revealed and monitored is that it enables them to run a concurrent process in which the mechanics, electronics and the software activities can be aligned. The advantages of a concurrent process include even shorter lead-times and an increased potential for innovative solutions due to the achieved integration.

ACKNOWLEDGEMENTS

We wish to thank the company Invencon for allowing us to participate in and conduct research during the course of the development of the Strong Hand device. We also wish to thank the consultancy company IPU and the Technical University of Denmark for funding the research. The research and the case study do not have any connection to Novo Nordisk A/S despite the affiliation of one of the authors to this company.

REFERENCES

Albers A, Braun A, Sadowski E, Wynn DC, Wyatt DF, Clarkson PJ (2011) System Architecture Modeling in a Software Tool Based on the Contact and Channel Approach (C&C-A). Journal of Mechanical Design, Vol. 133, No. 10, pp. 101006-101008.

Andreasen MM (1980) Machine Design Methods Based on Systematic Approach. PhD thesis, Lund University, Sweden.

Borches PD, Bonnema GM (2010) A3 Architecture Overviews - Focussing Architectural Knowledge to Support Evolution of Complex Systems. 20th Annual INCOSE Symposium (IS2010), Chicago, 2010.

Buur J (1990) A Theoretical Approach to Mechatronics Design. PhD thesis, Technical University of Denmark, Denmark.

Cabrera AAA, Woestenenk K, Tomiyama T (2011) An Architecture Model to Support Cooperative Design for Mechatronic Products: A Control Design Case. Mechatronics Vol. 21, No. 3, pp. 534-547.

D'Amelio V, Tomiyama T (2007) Predicting the unpredictable problems in mechatronics design. International Conference on Engineering Design (ICED'07), Paris, 2007, pp 703-704.

Danilovic M, Browning TR (2007) Managing Complex Product Development Projects With Design Structure Matrices and Domain Mapping Matrices. International Journal of Project Management, Vol. 25, No. 3, pp. 300-314.

Felgen L, Deubzer F, Lindemann U (2005) Complexity Management During The Analysis of Mechatronic Systems. International Conference on Engineering Design (ICED'05), Melbourne, 2005, p 409.

Gausemeier J, Frank U, Donoth J, Kahl S (2009a) Specification Technique for the Description of Self-optimizing Mechatronic Systems. Res Eng Des, Vol. 20, No. 4, pp. 201–223.

Gausemeier J, Giese H, Schäfer W, Axenath B, Frank U, Henkler S, Pook S, Tichy M (2007) Towards the design of self-optimizing mechatronic systems: Consistency between domain-spanning and domain-specific models. Guidelines for a Decision Support Method Adapted to NPD Processes.

Gausemeier J, Schafer W, Greenyer J, Kahl S, Pook S, Rieke J (2009b) Management of Cross Domain Model Consistency During the Development of Advanced Mechatronic Systems. International Conference on Engineering Design (ICED'09), Palo Alto, 2009, pp 1-12.

Group OM (2011) OMG Unified Modeling Language (UML) Specification V2.4.1. http://www.omg.org/spec/UML/2.4.1/.

Hehenberger P, Egyed A, Zeman K (2010) Consistency checking of mechatronic design models. Proceedings of the 2010 ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2010, Montreal, 2010, pp. 1141-1148.

Hubka V, Eder WE (1988) Theory of Technical Systems: A Total Concept Theory for Engineering Design. Springer Verlag.

Integration Definition Methods (2012) IDEF. Integration Definition Methods. www.idef.com.

Jansen S (2007) Eine Methodik zur modellbasierten Partitionierung mechatronischer Systeme. PhD thesis, Ruhr-Universität Bochum, Achen.

Mohringer S, Gausemeier J (2002) An Interface Specification for Principle Solutions Supporting the Cross-Domain Design of Mechatronic Systems. International Design Conference (Design'02), Dubrovnik, 2002, Vol. 7, pp. 533-538.

Nagel RL, Vucovich JP, Stone RB, McAdams DA (2008) A Signal Grammar to Guide Functional Modeling of Electromechanical Products. Journal of Mechanical Design, Vol. 130, No. 5, pp. 051101-051110.

Nonaka I (1994) A Dynamic Theory of Organizational Knowledge Creation. Organ Sci, Vol. 5, No. 1, pp. 14-37.

Object Management Group (2010) OMG Systems Modeling Language Specification V1.2. http://www.omg.org/spec/SysML/1.2/PDF/.

Shah AA, Kerzhner AA, Schaefer D, Paredis CJJ (2010) Multi-View Modeling to Support Embedded Systems Engineering in SysML. Graph Transformations and Model Driven Engineering- Lecture Notes in Computer Science 5765/2010:580-601. DOI:10.1007/978-3-642-17322.

Tomiyama T, Amelio VD, Urbanic J, ElMaraghy W (2007) Complexity of Multi-Disciplinary Design. Annals of the CIRP, Vol. 56, No. 1, pp. 185-188.

Torry-Smith JM, Qamar A, Achiche S, Wikander J, Mortensen NH, During C (2012) Challenges in Designing Mechatronic Systems. Journal of Mechanical Design, Vol. 135, No. 1, pp. 011005

Torry-Smith JM, Mortensen NH, Achiche S (2013) A Proposal for a classification of product-related dependencies in development of mechatronic products. Res Eng Des, Vol. 25, No. 1, pp. 53-74.

Wynn DC, Nair SMT, Clarkson PJ (2009) The P3 platform: an approach and software for developing diagrammatic model-based methods in design research. International Conference on Engineering Design (ICED'09), Palo Alto, 2009, pp. 559-570.