Model-Based Systems Engineering in Mobile Applications

Dipl.-Ing. Oliver Koch

Institut für Fluidtechnik (IFD), Technische Universität Dresden, Helmholtzstrasse 7a, 01069 Dresden, E-mail: koch@ifd.mw.tu-dresden.de

Professor Dr.-Ing. Jürgen Weber

Institut für Fluidtechnik (IFD), Technische Universität Dresden, Helmholtzstrasse 7a, 01069 Dresden, E-mail: mailbox@ifd.mw.tu-dresden.de

Abstract

An efficient system development needs reuse, traceability and understanding. Today, specifications are usually written in text documents. Reuse means a copy and paste of suitable specifications. Traceability is the textual note that references to affected requirements. Achieving a full context understanding requires reading hundreds of pages in a variety of documents. Changing one textual requirement in complex systems can be very time-consuming. Model-based systems engineering (MBSE) addresses these issues. There, an integrated system model is used for the design, analysis, communication and system specification and shall contribute to handling the system complexity.

This paper shows aspects of this approach in the development of a wheel loader's attachment system. Customer requirements will be used to derive a specification model. Based on this, the author introduces the system and software architecture. The connection between requirement and architecture leads to a traceable system design and produces the huge advantage of MBSE.

KEYWORDS: Mobile Applications, SysML, Model-Based Systems Engineering, Software Engineering

1. The growing complexity in mobile applications

Today, electronic is a huge driver of innovation in all our products. Looking at the construction machinery, novel technologies provide efficient drive trains and new control concepts. **Figure 1** shows the main system architecture of a modern wheel loader. Without the use of electronic control units, the implementation of an efficient power split transmission for the travel drive, a displacement controlled attachment as well as a hybrid unit will not be possible. Additionally, a separate power management

unit controls every subsystem and keeps the machine in its optimal state. /1/ showed a development of a driverless road roller. Based on GPS and the paver finisher's process data, the system plans its ideal route to compact the asphalt. This reduces the required distance travelled by the roller and hence it saves coasts and energy consumption. You find solutions for automated mining on the market. Especially, transport and unloading tasks are done automatically by the machines itself. The operator sits in a control room, supervisors and takes control in complex situations such as digging or loading material /2/. Activities to automate these tasks as well are shown for instance in /3/. In /4/ and /5/ an overall machine-communication enables the integration of construction progress optimization via simulation at run-time. Machines work co-operating. The control centre monitors the workflow and influences it through weighting of parameters.

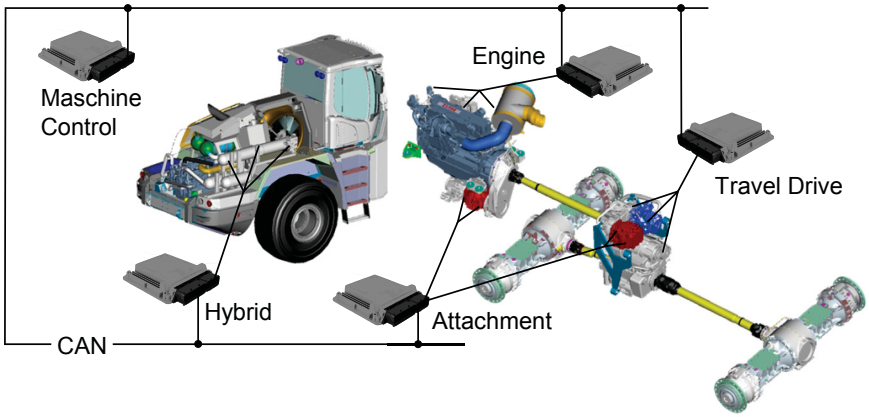

Figure 1: System architecture of a modern wheel loader, based on /6/

The European Construction Equipment Committee and Agricultural Machinery Industry Association summit in 2015 addressed the topic of “Smart Machines” /7/. Manufacturers provided an insight into development trends. Cyber-Physical Systems and the Internet of Services will also effect this industry. In future, unmanned autonomous machines will handle a wide range of tasks. The high level of automation enables self-optimization and leads to an increase of efficiency and a reduce of emissions. Hence, this will contribute to the sustainable use of our environment. Software becomes more and more important and the most competitive advantage in the development of smart technologies. The opportunity in changing the machine behaviour during operation provides flexibility and generates additional customer values. Novel business models and services will be the key to success. Until 2020, 26 bn things will be connected to the internet via IPv6. In consequence of this networking, fuzzy system boundaries increase the complexity. The developed machine

must be considered as a “System of Systems” /8/. Safety, security and reliability in context of a worldwide connectivity has to be taken into account.

2. Model-Based Systems Engineering

Figure 2 illustrates the V-Model approach – the typical engineering development cycle. Its use is recommended in complex and safety-critical systems /9/. While the left branch handles the design phase, the right branch includes test and integration steps. The domain specific implementation is done at the bottom. The approach follows the top-down principle, usually in a Zig-Zag pattern /10/. Based on stakeholder requirements the system architecture will be designed on different abstraction levels. They are connected through requirements of an upper system level that has to be satisfied by a deeper system level design until the component architecture level is achieved. This is always an iterative process. Several engineers usually take part in the development of complex systems. They cover different kind of roles with a different knowledge and system understanding. All of them have to communicate and exchange information. Today, “throwing these over the fence” leads to duplications of requirements and a wilderness that no one has time to check and release them all /11/. Using simple text-documents reinforces this situation. /12/ identifies this as the “three evils of engineering”: complexity, lack of understanding and communication problems. Thus, one of the big challenges in engineering is handling the changing of requirements and system architecture due to development iterations.

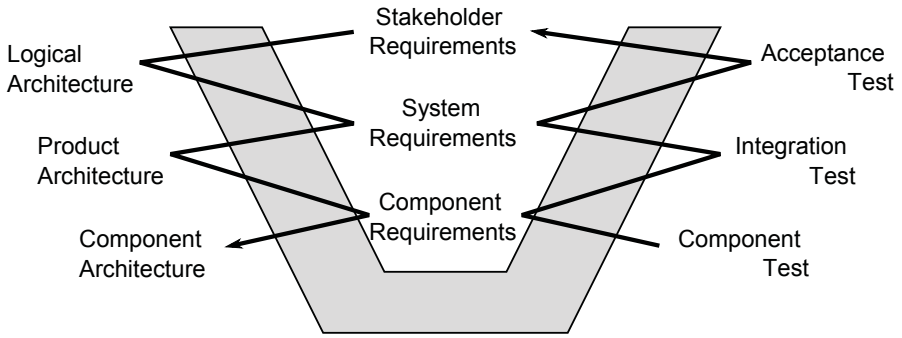

Figure 2: Zig-Zag pattern in V-model approach in systems engineering, based on /10/

Model-Based-Systems Engineering contributes to an integrated development process to minimize these issues. Compared to document-based developments, all information is stored in an entire data model. Every developer can use and change the model in real-time, thus the data is always up-to-date. The modelling of requirements and system design within one model leads to an overall traceability. The information

becomes machine-readable so that software tools are able to support the engineering process. They help to analyse and check consistency. Using a graphical language additionally contributes to a better communication and system understanding /13/. The Object Management Group (OMG) published in September 2007 its first version of the graphical Systems Modeling Language (SysML) /14/. It is a modification of the well-known Unified Modeling Language (UML). The specification includes a set of model elements and notations to describe a system from different views. SysML handles requirements, structure, behaviour and parametric relations. The language is not domain specific but it provides the concept of profiling to extend model elements to particular needs. The implemented allocation technology allows the modeller to connect several abstraction levels and system views without specifying their relation in detail. Every design decision becomes traceable. The engineer is able to check consistency within the model by defining rules of how elements should be connected. The extensive definition of relationships between model elements easily allows detecting all consequences of a changed requirement. Design patterns, own libraries and customization contributes reuse.

A suitable methodology is at least as important. There is not a common principle. It usually depends on the company's philosophy. **Figure 3** illustrates the simplified systems engineering approach that /15/ suggests. It can be integrated in the V-model approach and focuses on some basic steps. Defining the system context (1) and analysing stakeholder needs should be done at first in a system development. The results are used to derive requirements (2) and use cases (3) to represent all services the system has to offer. Their description with activities (4) results in functional aspects of the requirement model. The followed system architecture (5) design is the most creative part in systems engineering and also a challenging one. Unfortunately, there is no general approach. A good design always depends on the system architect's experience and talent /10/.

The next pages of this paper deal with these basic steps by use of the previous mentioned wheel loader example. It explains the system development with SysML for the attachment unit. The paper describes how to connect the essential views and steps to get a traceable model. At the end, we will derive the software architecture from the system description and elaborate some advantages of MBSE.

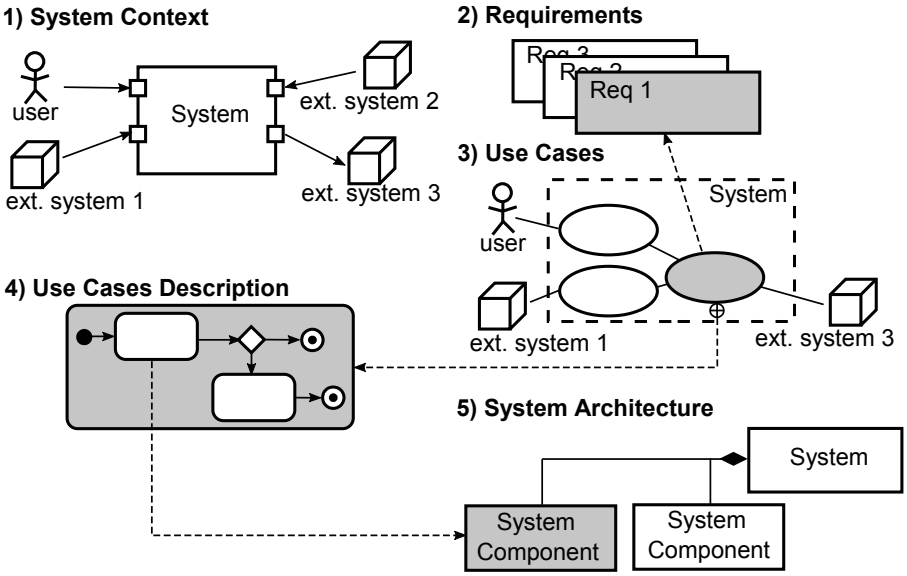

Figure 3: MBSE approach

3. System context and requirements modelling

In this example, we suppose to be a supplier for a wheel loader company. They are interested in advantages of a displacement-controlled attachment but do not have the resources to develop such a system. **Figure 4** illustrates its context view. It defines the system boundaries and contributes finding all services the attachment offers to and requests from its environment. The developed system is represented as a block that has various connections to other machine components. The triangle on each connection describes the exchange of information or energy. It is possible to specify each interface in detail to know how these parts are interacting with the attachment system. In this example, the attachment's main purpose is translating the engine's rotatory power into a translational movement that drives the kinematic. Therefore, a joystick sends user commands to control the system. An overall machine control unit (MCU) transmits control commands as well. The wheel loader's relay board is responsible for the necessary voltage supply and a separate auxiliary unit provides hydraulic oil. The attachment pumps are mounted on two different shafts. The system uses a power take-off from the transmission and from an additional pump-sharing unit. This figure is the first diagram that should be discussed with the customer. Determining the system context is an important step because it communicates responsibilities, constraints and interactions and helps to understand their needs. Many requirements can be already derived within the first iterations.

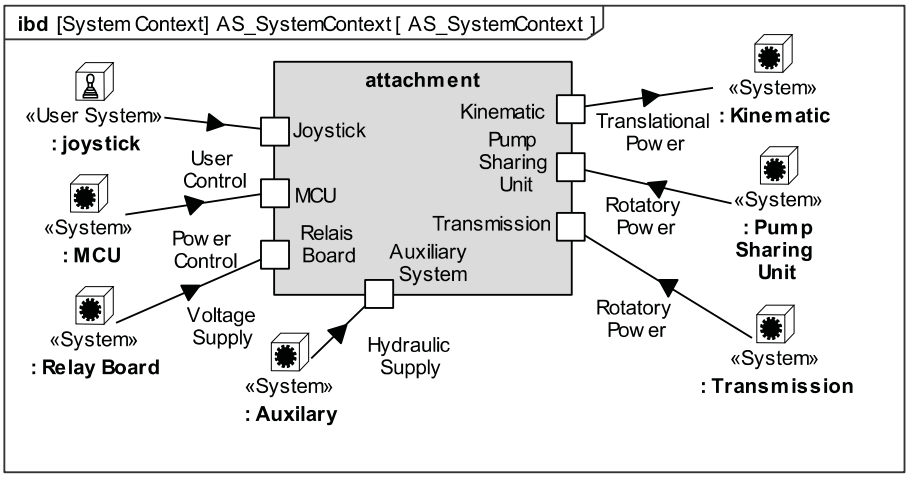

Figure 4: Attachment system context

Figure 5 shows some of the main requirements. The attachment system has to provide interfaces for a supervisor system and a power management control. Their machine provides a return-to-dig function that has to be realised as well. If the user presses a button on the joystick, the bucket shall automatically move back to its digging position. This figure represents a requirement view on our system and contains the first model elements. Each box stands for a requirement. We used the profile mechanism to extend the requirement element to a more specific one – a requirement that represents a specification from the system's stakeholders.

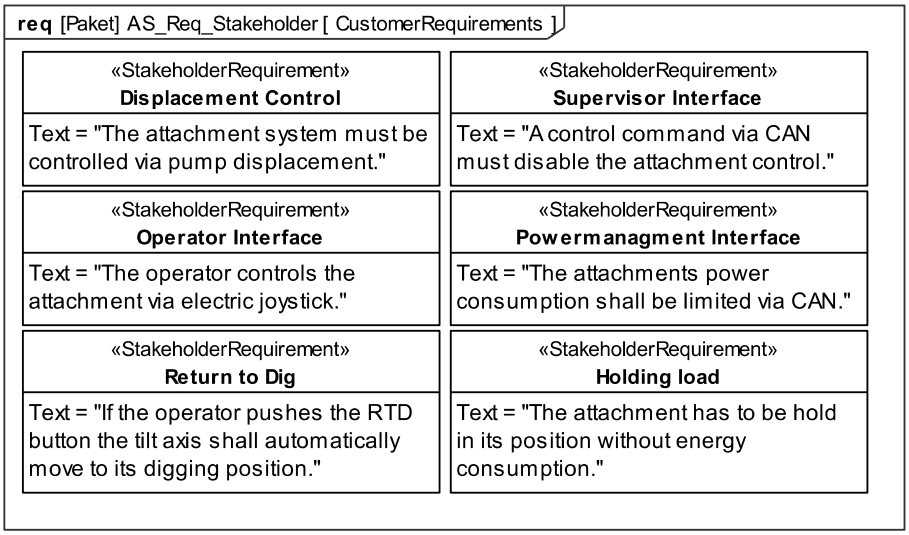

Figure 5: Customer requirements of the attachment system

Identifying the services, based on the system context and the customer requirements, is the next fundamental step. They are usually described in a use case diagram. /12/ gives some interesting hints concerning the modelling of these functional aspects and explains when it is useful to decompose a functionality or a system actor. The use case diagram in **Figure 6** contains some of the identified attachment's services. They are represented by an ellipse and connected to the system actors. This figure also characterizes different dependencies between use cases. The similarity of lifting and rolling can be described through a generalization. It has a white triangle at the end of the connection. The include-association is used if a service is part of another use case. In the example of Figure 6 driving the axis also includes the limitation of power. An extend-relation considers the fact that the functionality of a use case may change in some way like enabling the attachment through a signal coming from the MCU.

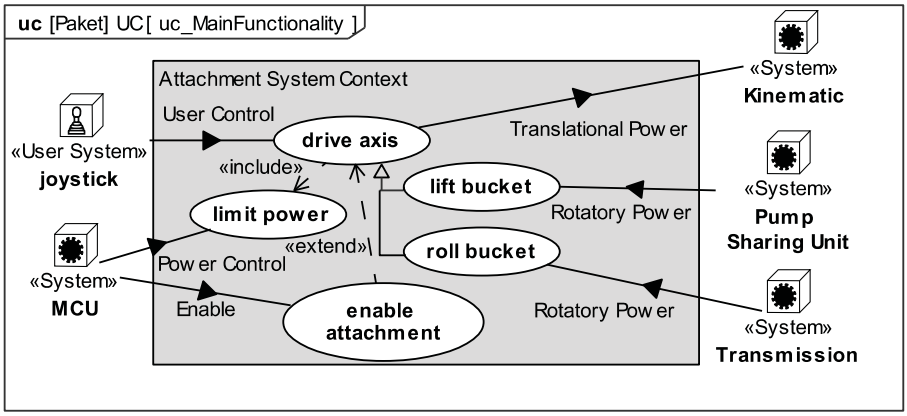

Figure 6: Use case “drive axis”

The next step should describe every use case in detail. It can be used for system developers in the design to choose suitable components as well as for test engineers to derive tests for verification and validation. SysML provides two diagram types that are predestined. A state machine expresses changes of system states through a specific use case. Within the state machine you define the conditions for changing to another state. On the other side, an activity diagram shows flows of processes, information or physical subjects. **Figure 7** models the drive axis use case as an activity. Rectangles at the diagram boarder expresses its in- and outputs (engine speed, pressure signal, torque power, etc.). The rounded rectangles represent actions that do something with the objects flowing into them. These actions themselves can also be an activity as well that another diagram explains in detail. It allows to group functions in a hierarchical model with various abstraction levels and to reuse activities in different

contexts. This figure is divided into two parts. The energy stage characterizes the connection between the rotational speed of the pump shaft and the translational speed of the hydraulic axis. The information side specifies how the input data is to be transformed to a ratio command. Based on the joystick deflection a speed demand will be calculated. Considering the engine speed and a ratio limit that considers the power limit, a ratio command will be derived.

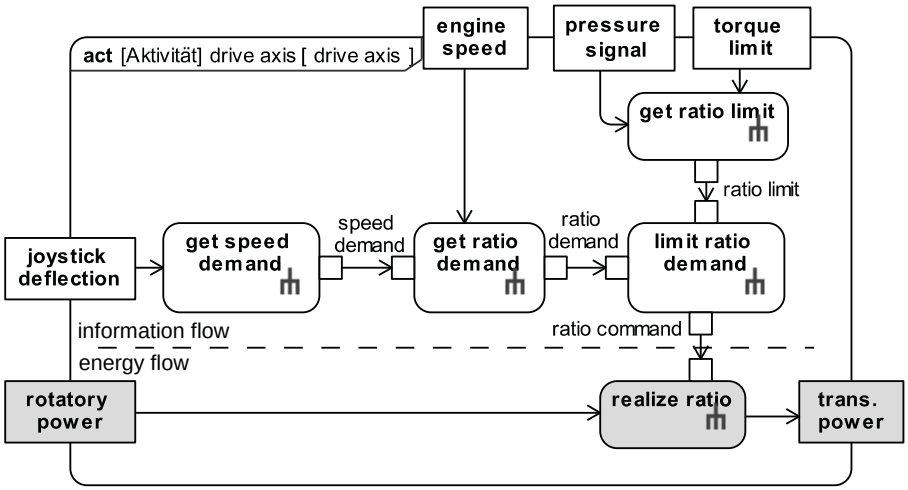

Figure 7: The drive axis activity describes its use case

4. System design

A block definition diagram realizes structural aspects of a system model and their relationships. **Figure 8** presents some of the attachment's main components, which are determined during the first design stage. This diagram only uses the composition association to express its configuration. In this example, the system consists of three cylinders, a control unit, two pumps and a wiring harness. They are directly connected to the block Attachment. Although each cylinder also consists of several parts like rod, piston, seal or bearing it will be bought as an assembly. There is actually no need to decompose it into smaller units. On the other side, the pumps for the displacement control own additional valves and sensors integrated in their end caps. There, it is useful to model these parts in more detail. SysML offers the internal block diagram to define the detailed relation between all components of a block definition diagram. It is possible to describe the electrical or mechanical connection as well as the information that is exchanged. SysML modelling tools usually detect wrong connections. While designing, the system modeller gets feedback on consistency.

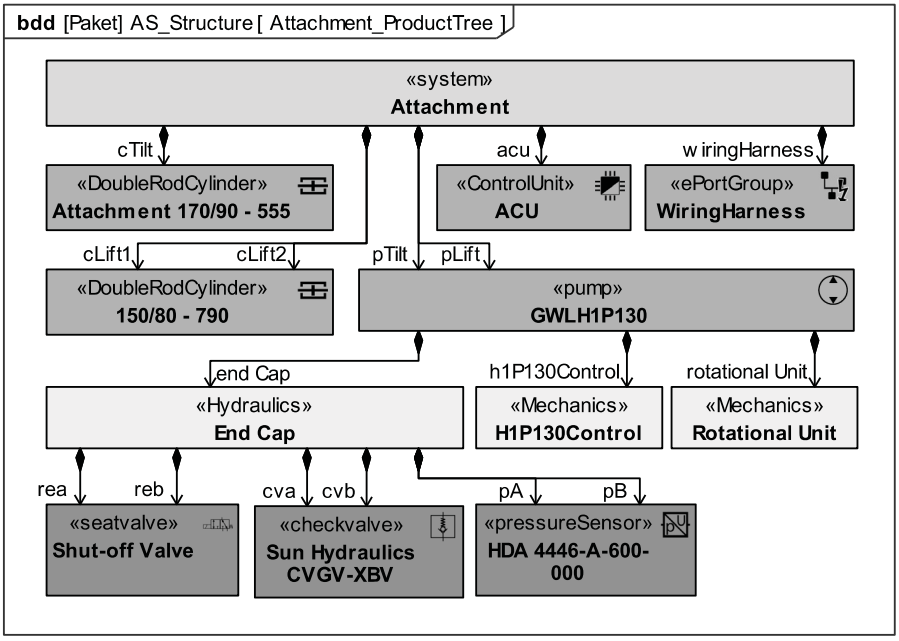

Figure 8: Attachment composition

Based on the main system architecture, it is possible to derive a basic software design for the control unit. We decided to use an object-oriented approach. It encapsulates functionality into software components, which provide interfaces to change their states. A direct access to internal data of an object is protected. This contributes handling software complexity and enables using a wide range of design patterns /16/. They describe approaches for solving similar problems especially in software engineering. Its dedication results in a flexible software design with a minimum of dependencies. A reaction to requirement changes will be eased. Although most embedded devices are programmed in C that is not a typical object-oriented programming language, /17/ explains a set of design patterns for embedded systems that can be used in C-applications. **Figure 9** shows the three-layered controller's software architecture. The lowest level describes the Hardware Access Layer. It includes a set of manufacturer libraries to get access to the specific controller hardware like analogue inputs and outputs, CAN controllers, timers, watchdog and so on. On top, the Application Layer contains all application specific components. They have to fulfil the functional requirements. In our case, the Joystick module evaluates the user data. Tilt Control and Lift Control are responsible for the axes' speed demand calculation. They use the same Speed Controller module but realize a different state machine to consider that both axes support diverse additional functions. For instance, just the tilt axis is involved in

the return-to-dig function. The implementation of the state machine follows the same-named design pattern. The class State defines the overall interface for all derived states. Each handles its execution and the transition to other states. This approach lets us reuse the Speed Controller's functionality in different situations and enables an easy implementation of new features that are currently unknown. Based on the speed demand calculation, the Pump Controller determines the control values for the pumps. The Supervisor monitors and will intervene if any error occurs or the overall machine controller disables the axes. The Operating System Layer lies in between. It includes an abstraction of the hardware access to achieve a hardware independent design. A change in the controller hardware shall only need a modification of this layer. This application uses a CAN controller to control the CAN network access, a separate IO module for the analogue inputs and outputs as well as a task manager that calls application modules within a defined time interval.

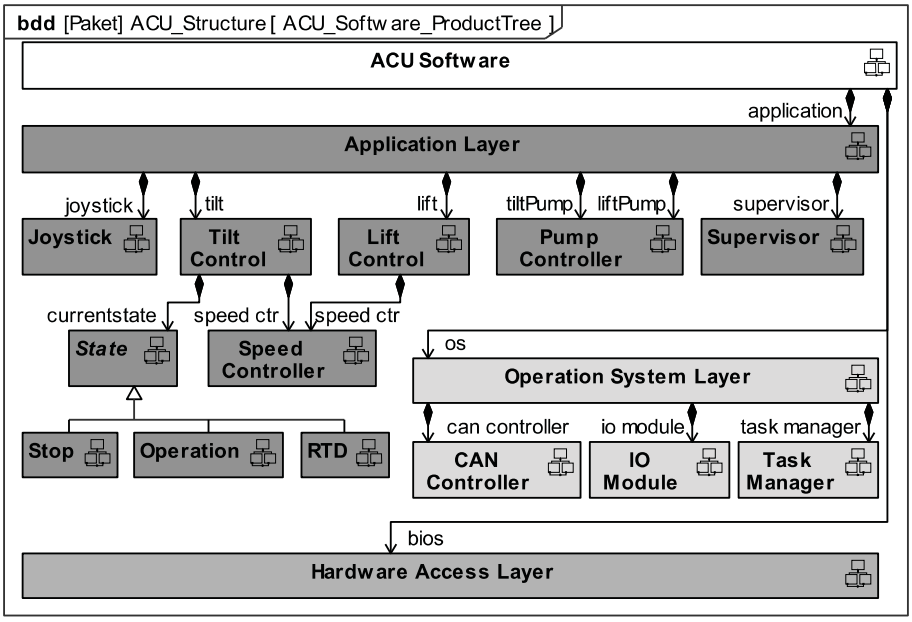

Figure 9: Software architecture

Figure 10 illustrates an internal view on the software architecture concerning the Application Layer and the Operating System Layer. It explains how different components interact. They are represented as blocks and own a set of ports, which are shown as small squares. Ports are only connectible if they have compatible interfaces. For instance, the CAN Controller component follows the observer pattern principle. It provides an interface, which allows registration of other modules. These clients will be

notified by the CAN Controller if a registered CAN message arrives. Therefore, the CAN Controller has to provide operations for clients to register a message and CAN clients must also support an operation that allows the notification. This interface description is expressed within a connection between CAN Provider and CAN Client ports.

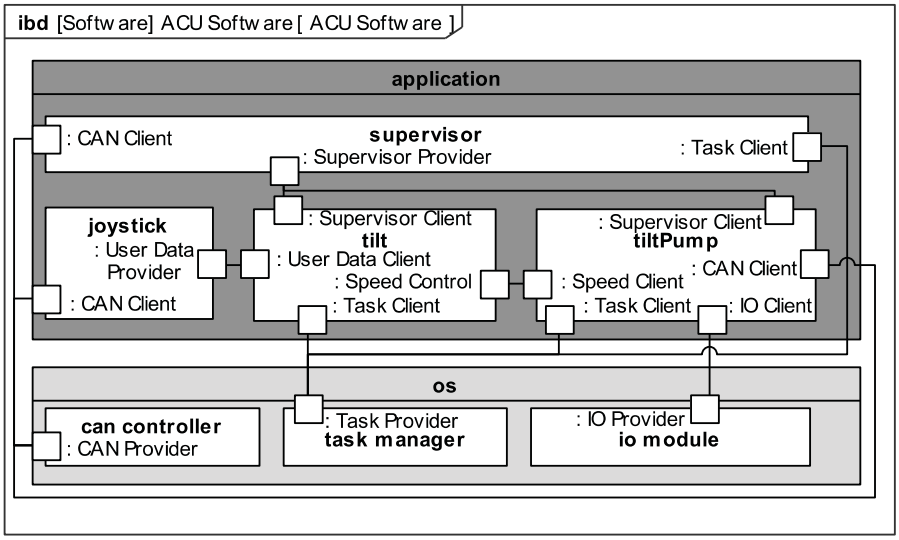

Figure 10: Dependency of software modules for tilt axis

5. Traceability

A diagram of model elements represents just one view on the system. A requirement diagram contains aspect of requirements, a block definition diagram shows issues of the systems composition and activity diagrams describe the system behaviour. Different points of view allow engineers to achieve a complete system understanding. **Figure 11** illustrates another view on the attachment system. It contains dependencies between requirements, use cases and components that the previous pages introduced. They are the key to a whole traceability. We started to define the system context and analysed stakeholder needs. We determined requirements that led to the demand for an electronic controlled pump as well as an electronic control unit. The refinement through use cases and its description with activities resulted in an extensive functional requirement model. It was the basis for the system design that has to fulfil all demands. Having found a solution, we allocated aspects of the activity model to system components and mapped system elements to requirements they satisfy. For instance, the software component Tilt Control as an integrated part of the ACU controller's software uses the Speed Controller that is responsible for calculating the ratio demand.

Its state machine implements a return-to-dig state to satisfy the same-named requirement. These modelled dependencies let us analyse the impact on a changing requirement.

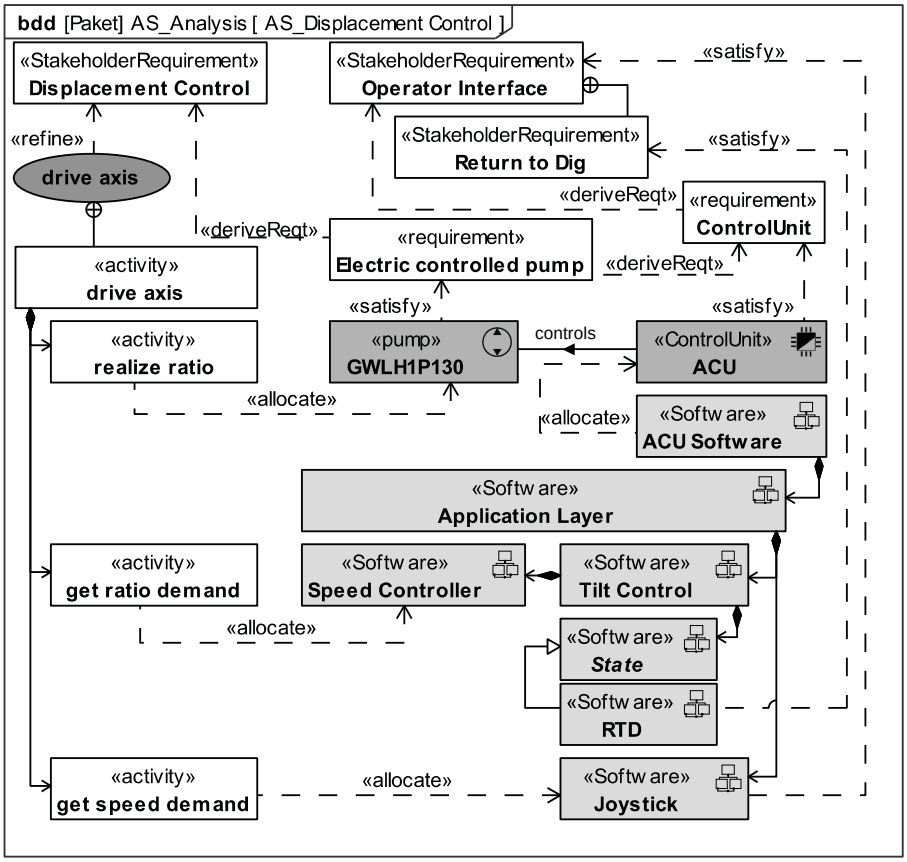

Figure 11: Traceability within the model

6. Conclusion

Is it worth it? Unfortunately, there is not any precise answer. You can hardly evaluate measurements. Every project is different and has its own challenges. At the end, everybody has to decide this by himself. /18/ has answered the question with a clear “yes”. The paper tells about the introduction of MBSE in a company's development process to achieve a systematic support for the system design, a basis for the software development and the system architecture documentation. To get a benefit greater than the effort, the engineers simplified their methodology. Within the first project, they had used less language constructs and a minimum set of diagram types. The author reported that the model itself still became the communication object. Due to this, they

achieved a fast collection of all relevant information across all disciplines. He is sure that some design faults were definitely avoided. During the project from /6/, we also used parts of the presented approach within the development of the whole wheel loader. We modelled the main system and software architecture, the CAN communication between the different subsystems and derived the Hardware-in-the-Loop test bench configuration for software tests. We have learned that is very important to understand the main idea of MBSE first and not to try to model everything that is possible. Because it is possible to model everything that costs a lot of time without any additional value.

However, when considering the model purpose, MBSE is the right approach and very helpful in analysing relations between requirements and design aspects. Engineers always use models to analyse problems and solutions. Expressing the system specification in a model as well will be solely consistent and can be a further step to an efficient engineering. Maybe the effort in developments concerning MBSE is a significant indicator to its worth. For instance, market-leading SysML tools are able to simulate activities and state-machines to get validations in an early design phase. They can also generate and integrate Functional Mock-up Units that enables a combined simulation of detailed physical models with SysML behaviour descriptions. Especially in safety-critical systems, there is an increased usage of SysML. /19/ describes activities that cover SysML-based safety analysis.

7. References

/1/ Halbrügge, Christoph & Johanning, Bernd: Automatisierte Asphaltverdichtung mit Tandemwalzen. ATZoffhighway, Nr. 2/2015, S. 56-67, 2015

/2/ http://www.miningandconstruction.sandvik.com/ (06.11.2015)

/3/ Lehnen, Bernd: Implementierung einer (Teil-) Automatisierung von Arbeitsspielen in der Steuerungssoftware von Hydraulikbaggern. 6. Fachtagung Baumaschinentechnik 2015, Dresden, 2015

/4/ Ansger, Jacob et al.: AutoBauLog Zweijahresbericht – Stand und Ausblick. 2011

/5/ Müller, Marcus et al.: Autonome Steuerung in der Baustellenlogistik: Modelle, Methoden und Werkzeuge für den autonomen Erdbau. 2013

/6/ Schneider, Markus et al.: Green Wheel Loader – Development of an energy efficient drive and control system. 9. International Fluid Power Conference, Aachen, 2014

/7/ CECE-CEMA Summit, http://www.cece-cema-summit.eu, (24.11.2015)

/8/ Muggeo, Christian & Pfenning, Michael: Die Rolle von MBSE und PLM im Industrial Internet. Tag des Systems Engineering 2015, Ulm, 2015

/9/ Verein Deutscher Ingenieure: Design methodology for mechatronic systems, VDI2206, 2004

/10/ Weilkiens, Tim et al.: Model-Based System Architecture. Wiley series in systems engineering and management, New York, 2016

/11/ Hood, Colin: Using Agile Methods to improve Efficiency in Requirements Engineering. Tag des Systems Engineering 2015, Ulm, 2015

/12/ Holt, Jon & Perry, Simon: SysML for Systems Engineering. The Institution of Engineering and Technology, London, 2008

/13/ Alt, Oliver: Modellbasiertes Systems Engineering und seine Technologien als Schlüssel für Industrie 4.0. Tag des Systems Engineering 2014, Bremen, 2014

/14/ Object Management Group: OMG Systems Modeling Language, http://www.omg.org/spec/SysML/1.4/, 2015

/15/ Weilkiens, Tim: Systems Engineering with SysML/UML. Elsevier, 2006

/16/ Gamma, Erich et al.: Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994

/17/ Douglass, Bruce, Powel: Design Patterns for Embedded Systems in C. Elsevier, 2011

/18/ Rogalski, Thomas: Beschleunigt oder bremst MBSE die Entwicklung von 500 Kilowatt Bremsleistung? Tag des Systems Engineering 2015, Ulm, 2015

/19/ Weilkiens, Tim et al.: System Safety in SysML. Tag des Systems Engineering 2015, Ulm, 2015