Architecture Modeling and Simulation

White Paper Resulting from Architecture Forum Meeting

November 2-3, 2011, Daimler, Sindelfingen - Germany

Edited by:

Dr. Gerrit Muller, Buskerud University College, Embedded Systems Institute

Mr. Peter Korfiatis, Stevens Institute of Technology

Input was provided by the following participants in the Architecture Forum:

Name	Organization
Kurdiya Atalan	Daimler
Rob Cloutier	Stevens Institute of Technology
Lars Ivansen	Micronic Mydata AB
Peter Korfiatis	Stevens Institute of Technology
Pierre vd Laar	ESI
Bjørn Victor Larsen	Kongsberg Defense
Hugo van Leeuwen	FEI Company
Hristina Moneva	ESI
Gerrit Muller	Buskerud University College/ESI

Name	Organization
Milan van den Muyzenberg	FEI Company
Sølve Raaen	Kongsberg Maritime
Thomas Ringler	Daimler
Nico van Rooijen	Philips Healthcare
Gabriel Schwefer	Daimler
Rolf Siegers	Raytheon Company
Martin Simons	Daimler
Aristo Togliatti	NCC Roads
Bernard van Vlimmeren	FEI Company

Published on July 22, 2013

1. Introduction

Many systems engineers perceive Model Based Systems Engineering (MBSE) as the next big step in systems engineering. MBSE replaces textual documents with models to capture specification and design of systems. Proponents claim that MBSE will improve traceability, facilitate analysis and design, allow automation of early validation and verification, and enable automatic code generation (SysML forum). MBSE will enhance communication, increase the ability to manage complexity, improve product quality, enrich knowledge capture, and improve the ability to teach (Friedenthal 2007). These developments trigger questions for architects and architecting:

- What are architecture models? (preferably illustrated by examples)
- How do architects model architecture?
- Do architects model aspects (e.g. cost, performance) or "the architecture"?
- Why would architects model?
  - To validate early?
  - Because simulation helps?
- When should architects model?
  - Early, but how early?
  - How far to go?

2. Cases from practice

Automotive

The automotive industry has high expectations from modeling. However, most automotive companies are in a phase where they are gradually moving towards modeling as standard practice. They are discovering what is working while evolving. As host, Daimler provided a presentation on their status. The slides are publicly available at [Schwefer 2010].

Modern premium cars can have more than 65 Electronic Control Units (ECU) connected by about 10 buses (CAN busses, Flexray, or Ethernet). Cost is the most important driver in the automotive industry, even in the premium sector. Roughly, 25% of the cost is already in software and electronics. Modern cars are perhaps the most complex mass-produced product today, where software and electronics are leading to more and more of this complexity. On one hand, software and electronics facilitate functional integration. On the other hand, advanced innovative functions implemented in software, such as accident free driving or emissions reduction, are increasing the overall complexity. The forum participants from Daimler work on designing the E/E architecture.

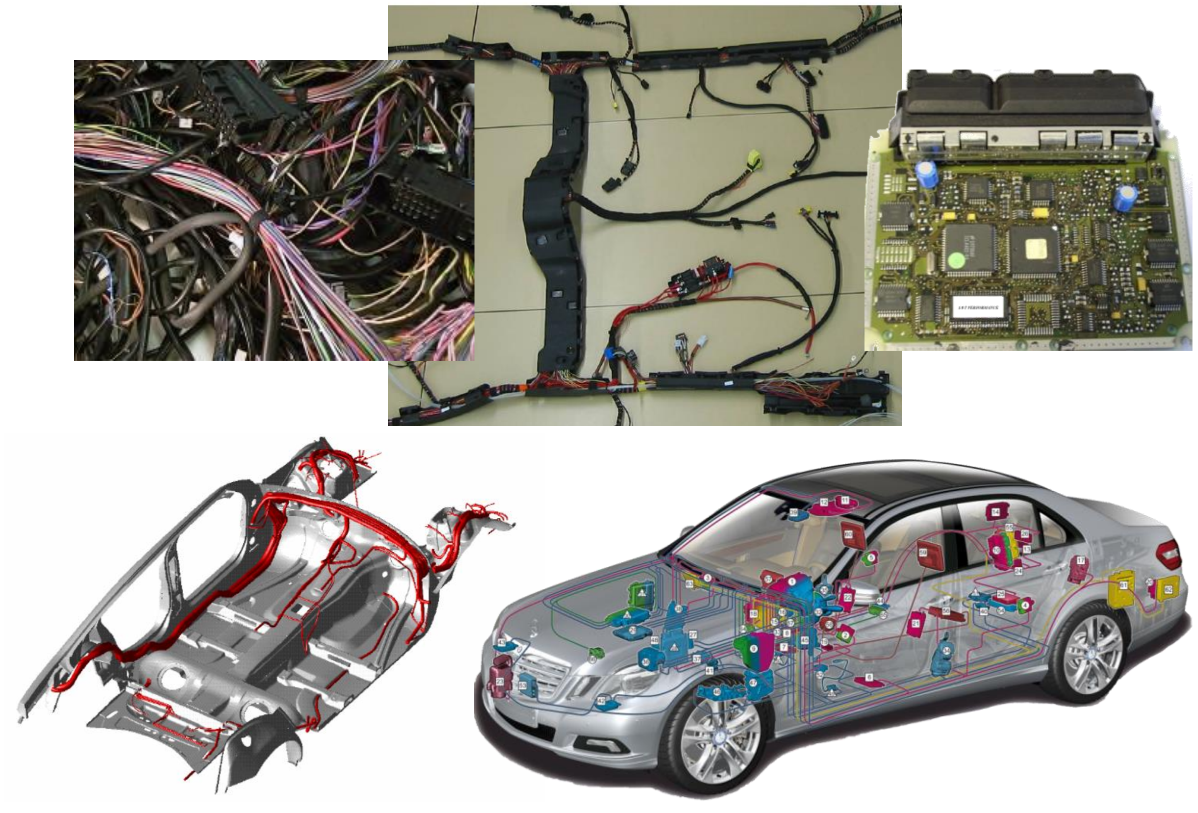


Figure 1. Typical electronics in a premium car (Photos courtesy of Daimler).

Development cycles in automotive are about 5 years, which is long compared to electronics and software technology innovations. The E/E architecture has significant impact on weight and manufacturing of an automobile; the amount of cables in a car is staggering, see Figure 1 for typical electronics in a premium car. Additional challenges include partitioning and allocation to suppliers. The automotive industry is known for its extensive supply network that is tuned towards the cost, volume, and quality needs of automotive.

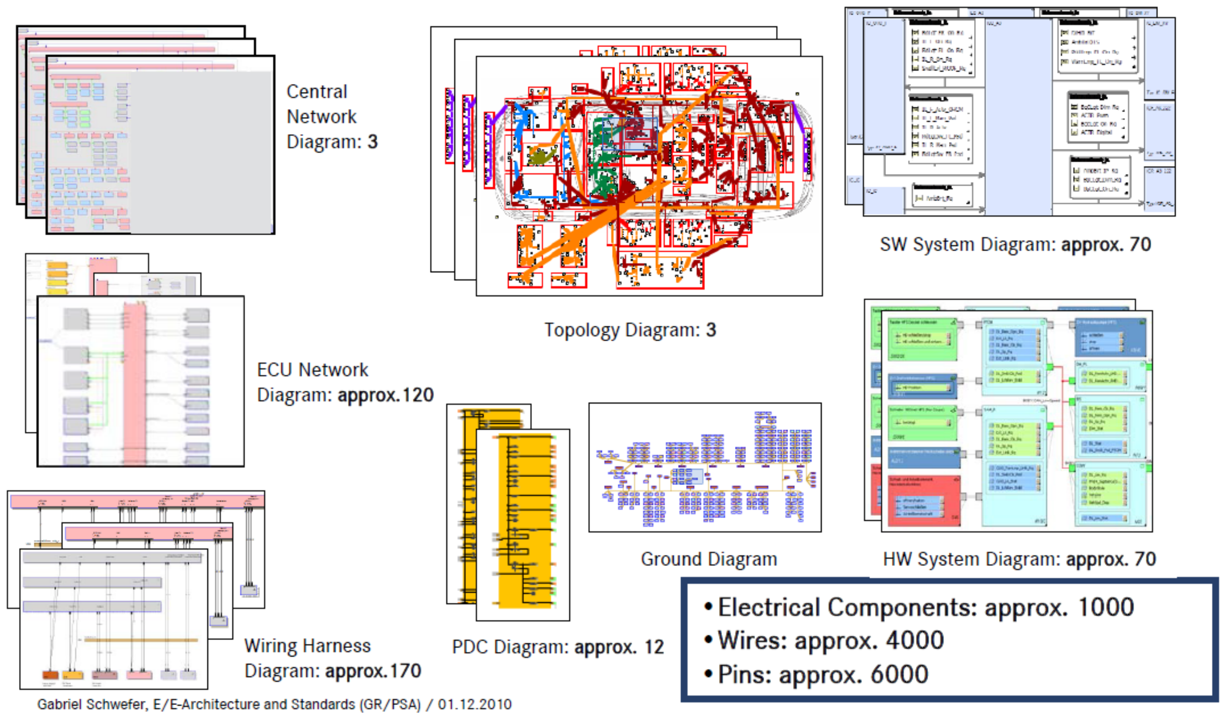

Central Network Diagram: 3
Topology Diagram: 3
SW System Diagram: approx. 70
ECU Network Diagram: approx.120
Ground Diagram
HW System Diagram: approx. 70
Wiring Harness Diagram: approx.170
PDC Diagram: approx. 12
• Electrical Components: approx. 1000
• Wires: approx. 4000
• Pins: approx. 6000
Gabriel Schwefer, E/E-Architecture and Standards (GR/PSA) / 01.12.2010

Figure 2. Some of the diagrams in PreeVision (diagrams courtesy of Daimler).

Daimler has been collaborating with the University of Karlsruhe on a modeling support environment. This tool environment was spun off as a small company, which was recently acquired by a major tool vendor. PreeVision supports physical partitioning, physical interfaces (e.g. wire harness, cables, and connectors), functional modeling, network topology, and signals (typical ca. 8000 in a car); Figure 2 shows a number of diagrams as an example. The tool can generate artifacts that can become part of development specifications. The tool at

Daimler is mostly used for metrics-based architecture evaluation. Other OEMs use the tool with a focus on architecture documentation. Daimler uses other tools in the systems engineering workflow that are not part of the architecting workflow, such as DOORS. The use of PreeVision is evolving and the organization is still learning how to embed it in its processes. For example, some of the challenges include who is responsible for updating models, how can integrity be maintained and what release policy should be used?

High precision manufacturing machines

The speaker presented some of the models that his company created during the development cycle of a new product. The specific product is an example of radical innovation with challenges in feasibility (will the concepts work), viability (can we make a business case), and organization (how to fit new concepts in an existing organization).

Figure 3 shows a time line of the project with the various models that developers produced and the typical tools used to produce them. Developers typically create many domain specific models to explore specific performance related questions. In this time line, we see that developers use a variety of tools for this purpose, e.g. Excel, Matlab, Zemax, and Comsol Multiphysics. Architects introduced SysML models in this project to capture customer context, requirements, and system structure (partitioning and interfaces). The idea was for these SysML models to serve as a shared understanding of the system architecture. About half way through the process, the development team created an animation of the main concepts to be used for analysis and communication. At the beginning and at the end we see that the project team has made business related models, in Excel and @Risk, to understand and validate the value and business propositions behind this new product.

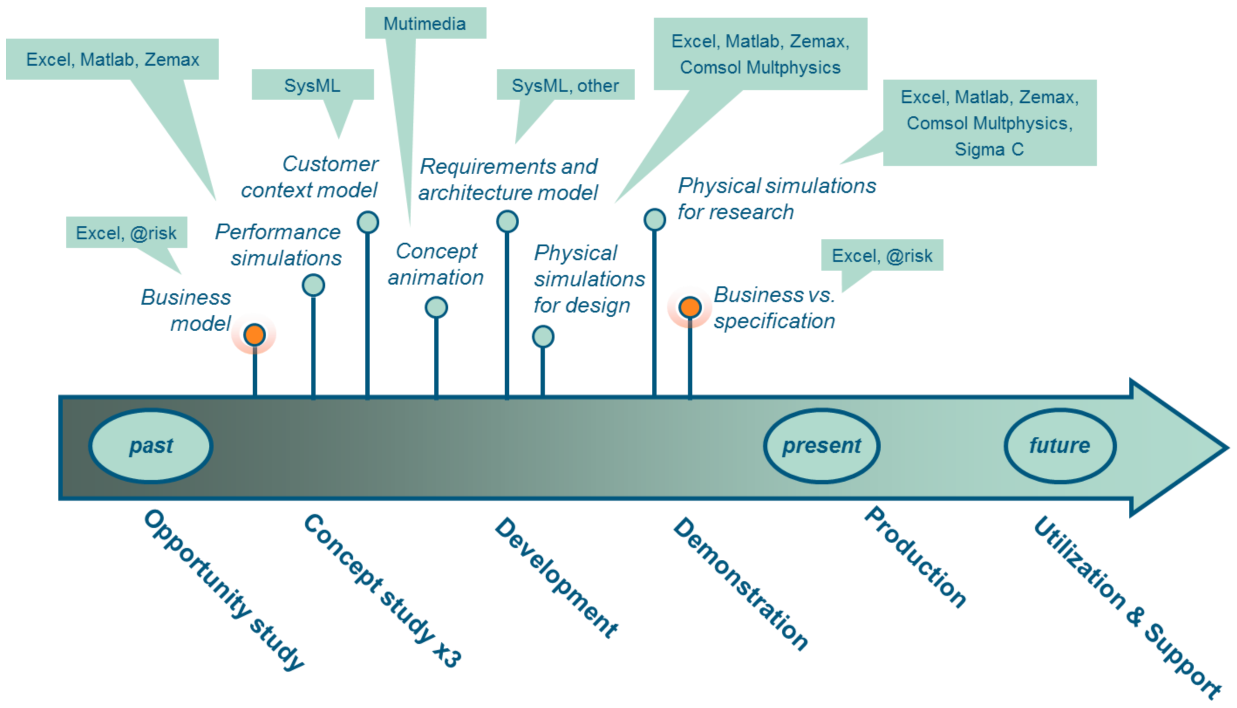


Figure 3. The development cycle of a new product and the type of models made during development.

The architects used SysML in the project in an attempt to create a shared “reality” for the entire team of developers, from marketing to technical experts. Figure 4 shows the top-level model in SysML that developers saw as “the architecture”. The figure uses pictures instead of blocks to improve recognition for stakeholders less familiar with SysML.

The original intent was that everybody on the team would use SysML (e.g. create, modify, and read). However, there was not enough time to train all of the team members. Therefore, a small group of people was responsible for creating and modifying SysML artifacts to support the broader team. Domain experts only needed to be able to read SysML. As shown in Figure 3, these experts, create their own domain specific models using other tools and formalisms.

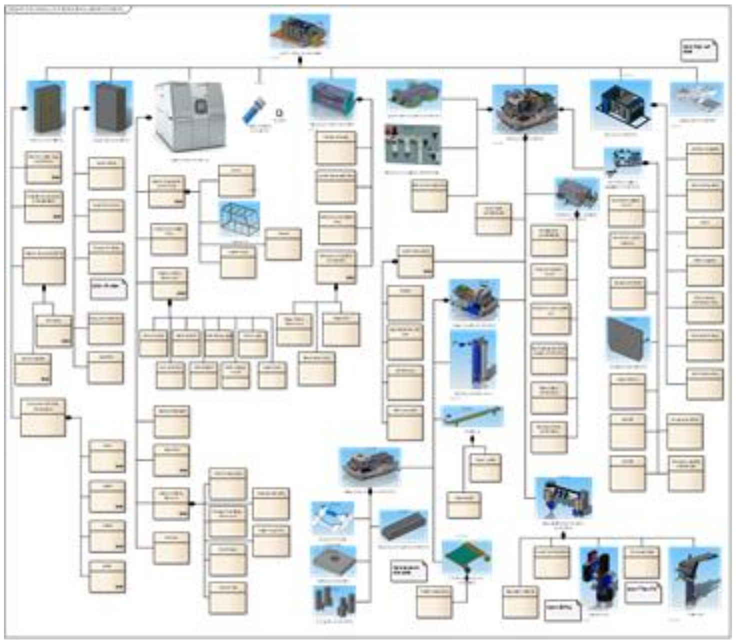


Figure 4. The architecture of the high precision manufacturing system in SysML.

The speaker made the following observations:

- Many block diagrams, only few behavior, requirement, and use case diagrams
- Weak impact of cost of goods
- Information was acquired and created outside the model

He offered as possible explanations:

- Experienced engineers break down problems in their mind
- Training and templates came late
- Method was lacking
- Vision and purpose was vague
- Navigation was hard
- The culture in the company was performance dominated, not cost

Defense

The presenter from the defense industry shared an inventory of models used in missile development and posed the question "What models help in architecting?" He used the following definition of architecting: "Architecting is the activity to create the right architecture, and involves communication with all the stakeholders in order to produce one or more alternative architectures where identified tradeoffs are handled."

For this speaker, the starting point is the question "How can modeling and simulation support this process in order to arrive at the right architecture?" The presenter offered a priori the following answers:

- Modeling and simulation may be used in order to gain insight into a domain in order to determine quantifiable properties of the architecture and to identify substantial issues that need decisions.
- One or more models may be built of the system or parts of the system with structure, properties and behavior to test or validate the system in the anticipated environment.
- Often, a model of the anticipated environment must also be created in order to get useful insight.

A prime challenge in missile development is that developing flying objects is expensive. Launching early versions of a missile with a high risk of error may result in loss of the expensive projectile, which can cause the program to be delayed . The approach in this sector is to have different test configurations, where a sub set of aspects is tested in each configuration. Disadvantages to this approach include the risk that one or more aspects may be left out, or that crossover behavior may not be tested. In early phases, there are known crossovers, but also unknown crossovers.

Missile development uses a mix of physical prototypes and models at different system levels to verify design and validate specifications. The following (hybrid) models are available:

- a host based Missile Simulator (MisSim),
- a HardWare In the Loop (HWIL) simulator,
- a "Missile On A Board" environment called IRMA,
- a POD tank equipped as a missile mounted on a F15 aircraft.

A Mission Simulator provides an environment where end-to-end functionality may be investigated. In addition, there are several other environments both at the system and subsystem level, for example, in Matlab/Simulink and sensor emulators. Various models ranging from digital models to mockups were created to investigate forms and other physical characteristics.

MisSim is conceptually a heritage from an earlier missile platform. The designers identified that it would be beneficial to have a system where application code runs 1:1 in simulator vs real world. That was not the case in the previous system. Physical aspects and surrounding environment are modeled to the level necessary for development and testing of the missile control loop.

The simulator (including the “system under test”) is run on a host computer platform, where the application code is “connected” to the emulated system through an infrastructure interface. Time is simulated to allow for hardware emulation. Time delays are configured into simulation in order to develop and test the control loop.

MisSim provides early validation of functionality. Designers can observe how the missile behaves in its environment and determine what capabilities and characteristics are needed for the missile to perform as expected.

One of the advantages of using (hybrid) models is the possibility of collecting a wide variety of data for analysis. Analysis and visualization of these data is a crucial aspect while using these tools.

Missile On a Board environment, seen in Figure 5, provides real sensors and electronics to verify interfaces and analyze how frequencies and sensor data transfer characteristics propagate through the system. This environment provides a reference platform for the HWIL simulator.

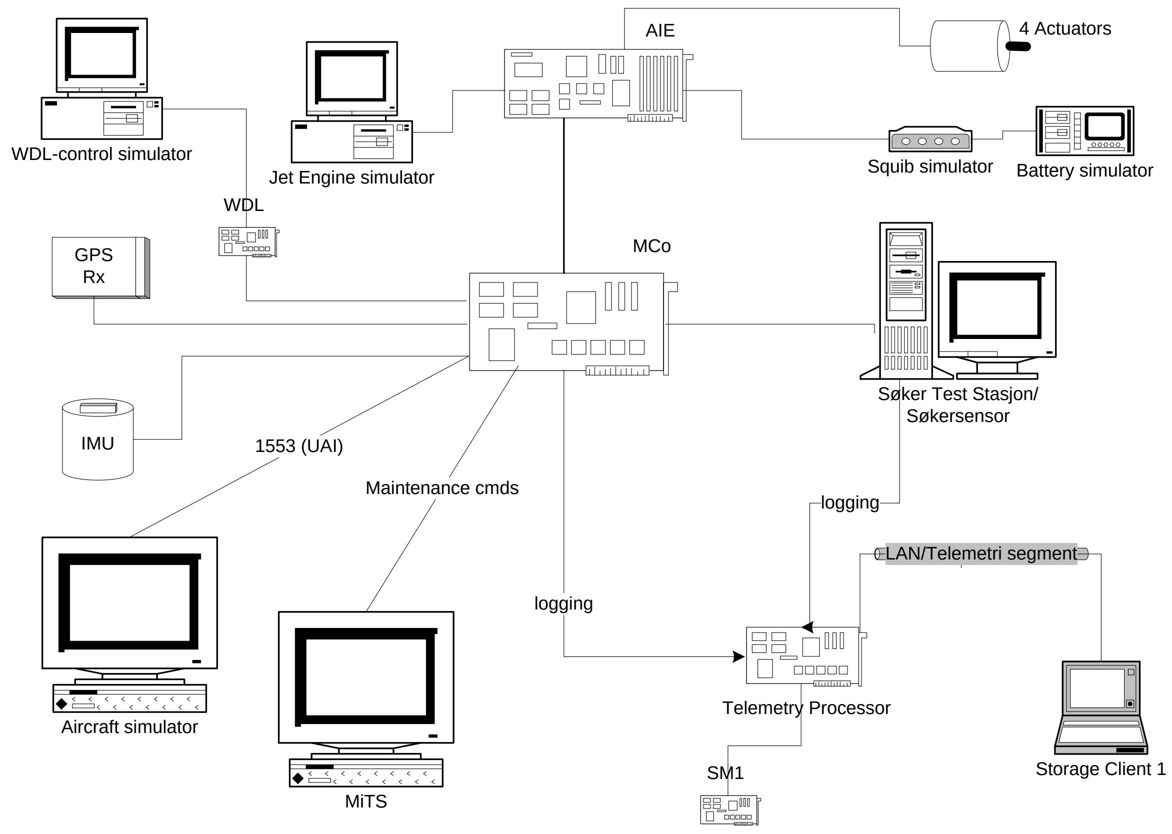


Figure 5 Block diagram of the Missile On a Board environment.

The **HWIL simulator**, seen in Figure 6, provides target electronics to run software. Sensors, physical characteristics and environment are simulated to the required level of precision. In this way, the simulator provides the software with sensor input through correct interfaces, at the correct rate and representative value variance. A dedicated computer system executing the simulation environment closes the loop for the control system. The HWIL simulator is used to verify the characteristics of the electronics hardware platform: does the platform react correctly and does it have sufficient resources to deliver output within the required time limits (externally visible characteristics).

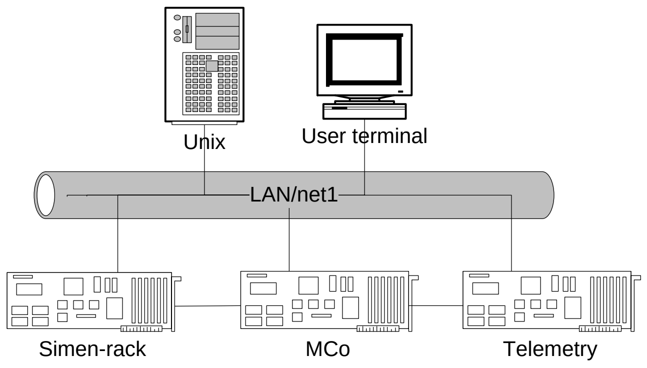


Figure 6 Hardware In the Loop (HWIL) simulator.

A POD tank, see Figure 7, is filled with sensors, electronics software, and other hardware to be tested under "real" conditions. An F15 AC runs preplanned trajectories, and attacks reference targets in order to test identification and sensors under real conditions. POD flights provide the equivalence of 800 to 1000 test fires.

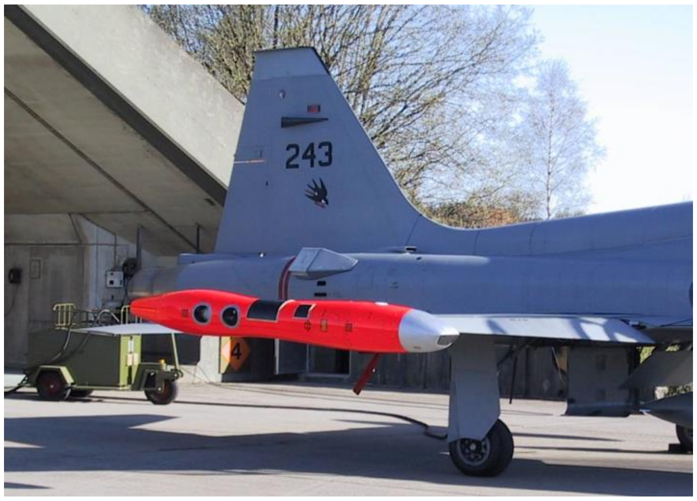
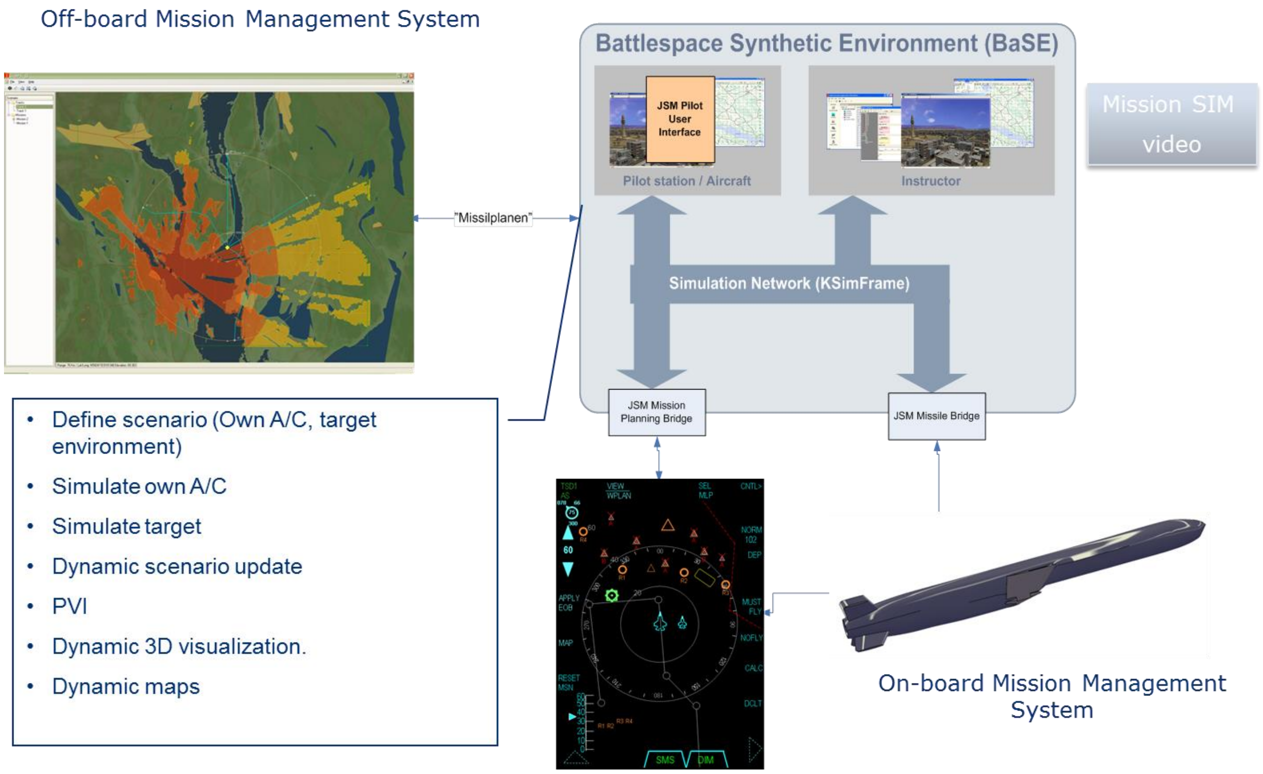


Figure 7 Photo of POD tank. (Photo courtesy of KDS)

Specific systems that are tested and verified in this way are the seeker sensor, navigation system, missile computer, missile flight software, and telemetry system.

Digital and physical mockups are complementary. The shape and weight distribution of the airframe including wings and control surfaces are designed and analyzed digitally. This digital model needs to be consistent with MisSim. Static scaled models of the missile are run in wind tunnels to validate form with respect to airflow, and acting forces.

A **Mission Simulator**, see Figure 8 has the broadest scope, including an Off-board Mission Management System, a Battlespace Synthetic Environment, and the On-board Mission Management System. Scenarios can be defined for the aircraft and targets, which are simulated and dynamically visualized in 3D and on maps that are provided.


Off-board Mission Management System
Battlespace Synthetic Environment (BaSE)
JSM Pilot User Interface
Pilot station / Aircraft
Instructor
Mission SIM video
"Missilplanen"
Simulation Network (KSimFrame)
JSM Mission Planning Bridge
JSM Missile Bridge
On-board Mission Management System

- Define scenario (Own A/C, target environment)
- Simulate own A/C
- Simulate target
- Dynamic scenario update
- PVI
- Dynamic 3D visualization.
- Dynamic maps

Figure 8. The Mission Simulator

After explanation of the variety of models that are used in missile development, the issue of architecture relevance was revisited. The presenter provided the following insights:

- How does the Simulation Environment support architecture development?
    - Architecture-relevant models and simulations are those that **help validate that the product has necessary capabilities and characteristics in order to solve its problem in the real world**.
    - To be able to conduct this validation at such an early stage where the real world cannot be used, we need an **adequate model of the real world**.
    - Adequate in this context means that it **supports the aspects of the product we desire to validate**.
    - The adequacy of the model is an issue throughout the early stages of a project.
    - Models must be **adjusted** when better knowledge of the actual situations is gathered. For example, wind tunnel tests provide data to adjust simulated forces in the simulator.
- Architecture relevance of **MisSim**
    - The MisSim, running on a host computer cluster, provides a development environment for the control loop, but also provides **architecture validation of the frequencies and time constants of the control loop against the characteristics of the airframe**.
    - Developing a model of the environment also provides a way of **determining which parameters of the "real world" are important and which can be treated as noise**.
- **Mission on a Board Environment** provides **limited relevance to architectural issues.**
    - This environment is essential in verifying interfaces and analyzing how frequencies and sensor data transfer characteristics propagates through the system.

    - It provides a reference platform for the HWIL simulator where the characteristics of the electronics hw platform is validated: that the platform reacts correctly and has resources to deliver output within the requires time limit (externally visible characteristics).
- **HWIL** provides **limited relevance for architectural issues.**
    - HWIL gives feedback on sizing of electronics and efficiency of implemented algorithms when executed on correct hardware.
    - HWIL is essential in verifying and analyzing how frequencies and sensor data (with "real value ranges"), transfer characteristics, and value ranges propagate through the system.
- **Mockups** are **highly relevant for architectural issues**.
    - Digital Mockups can be used to validate structural characteristics against a digital model of the context the product will deployed within.
    - Physical Mockups may be used to validate structural properties as airflow (through wind tunnel tests), and handling (by maintenance and operational personnel).
- POD runs are **relevant to architectural issues**
    - POD runs are important because they provide a near to correct environment for the test platform. They do not close the control loop, but provide accurate sensor data from the real world.
    - Provide an early opportunity to validate how the real world affects the system.
    - Examine if the system is robust enough for what is characterized as noise.
- A **Mission Simulator** has **high architectural value**
    - The Mission Simulator provides a framework for
        - testing prototypes of the off-board environment

        - Planning missions
        - Preparing for onboard activities
    - testing pilot interactions with the missile
        - Pilot Vehicle Interface
            - Weapon administration
            - Engaging based on preplanned missions
            - Target of opportunity
            - Weapon in-flight administration

3. Academic Vision on Modeling

Stevens Institute of Technology presented research conducted in their Visualization, Modeling, and Computation Lab. The core objective for modeling is to reason about the problem, to understand the complexities, and to communicate with others. The presentation began by suggesting that we begin modeling as kids, using such modeling tools as Lego bricks, model cars, airplanes, and ships, and sewing patterns. As a result of this, the coming generation of architects is growing up in an era where computer assisted modeling and simulation are commonplace. An example was the Lego "Design by me" website that provided a complete CAD based building environment to design Lego creations, as well as a service that delivered the physical blocks to build the design. The site itself has been closed since January 2012; however, the design program is still available at http://ldd.lego.com/download/. Figure 9 shows an example of designing with the Lego Digital Designer software program.

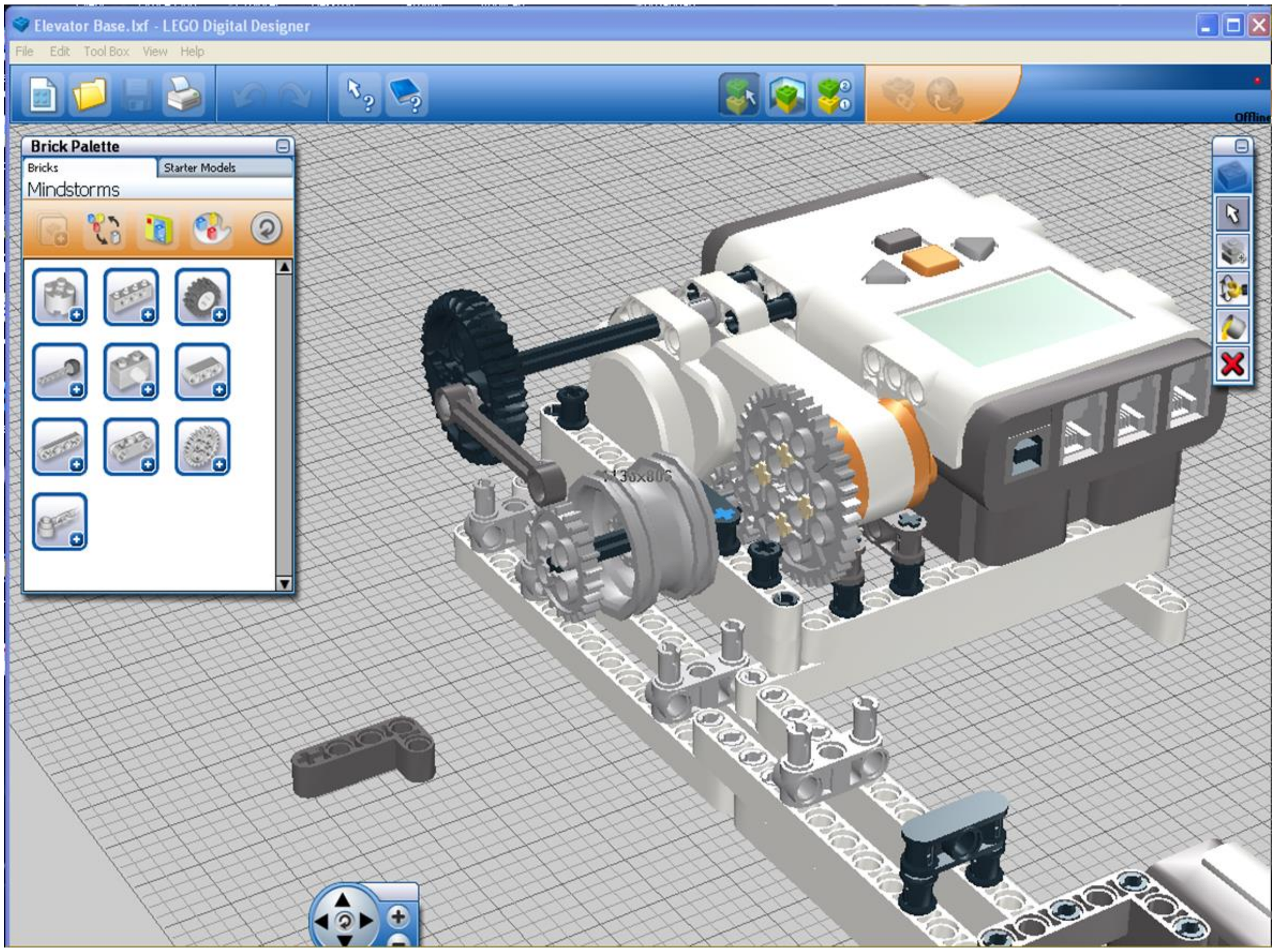

Elevator Base.lxf - LEGO Digital Designer
File Edit Tool Box View Help
Brick Palette
Bricks
Starter Models
Mindstorms

Figure 9. An example of designing with the Lego Digital Designer.

This Lego example illustrates how new generations of architects have been growing up in a world where models and simulations are ubiquitous. Their experiences as kids tells them that designers should always model systems before building the real thing.

The Lego example illustrates the MBSE principles. Taking a step back, we see MBSE as the combined use of modeling languages, development processes, and tools. Designers may use various approaches, e.g. functional, Hatley Pirbhi, or object-oriented. MBSE can be applied with any modeling language; the popular SysML is not the only possibility.

INCOSE has been using the SE^2 challenge to demonstrate solutions to challenging problems using MBSE. One such challenge sees a group of systems engineering professionals attempting to model the Active Phasing Experiment technology demonstrator for the future European Extremely Large Telescope, as seen in [Karban 2008, 2011] and http://www.omgwiki.org/MBSE/doku.php?id=mbse:telescope.

Challenges in transitioning to MBSE include:

- most instructors in academia do not have experience with MBSE and therefore it is difficult to teach
- there is a large up-front cost associated with tools and training
- there are many tool vendors with little compatibility between tools
- management is often afraid of the 'new' and may be averse to exploring the unknown

The presenter posed the following proposition as closure: no modern company would think twice about outfitting their civil architects with high-cost CAD software rather than drafting tables, why is this so different for the systems architect?

The **Embedded Systems Institute** (ESI) presented the "Design Framework", a framework to support architects in designing and modeling. The underlying assumption is that organizations design complex systems in multi-disciplinary teams applying concurrent engineering. The consequence is that the teams create and modify many models using various formalisms with tools that fit the specific problem area. Problems that may arise are:

- Designers make implicit assumptions that become part of models. If these assumptions are not checked against the current state of design later in the design process, the analysis of these models may be misleading or even wrong.
- Models are related, but what are the implications of changes on other models or the entire system?
- How can models of various system parts or in various formalisms be connected?

- How can various models be synchronized?
- How can the rationale behind decisions be documented?
- For what reason was a model introduced?

The speaker posed that it is not cost effective to model the complete system in all its aspects, but only the non-trivial parts of the system (new functionality and/or reengineering existing parts). As rule of thumb, the speaker indicated that about 10% of a system should be modeled. Part of the audience wondered if 10% is too little.

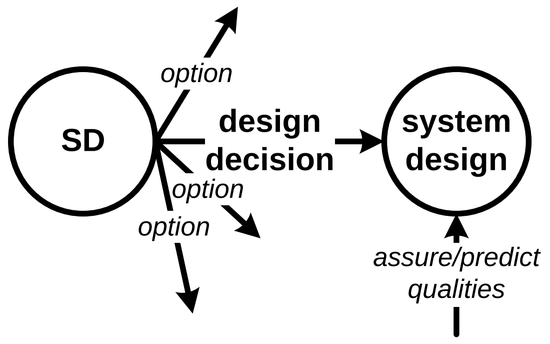
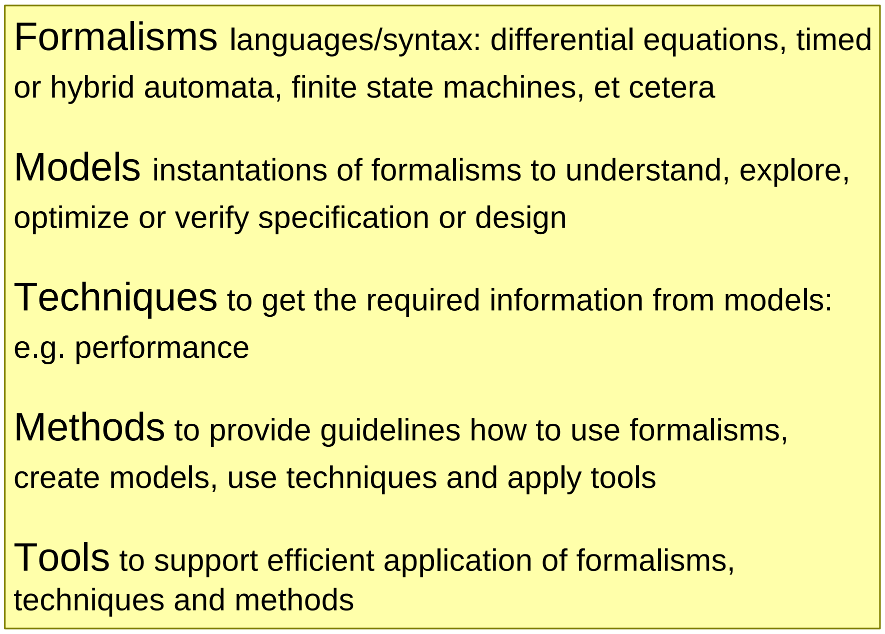


Figure 10 Basic concepts in the Design Framework

Researchers at ESI designed and built a prototype of the design framework as part of the Multiform project (http://www.esi.nl/multiform/). The goal of this research was to create a tool that would support architects by keeping track of design decisions and linking them to the modeling effort done in a variety of tools and formalisms. The vision of the framework is to be agnostic of specific processes, methods, tools, and formalisms. However, the framework requires a number of design and design flow concepts at its core. Figure 10 shows a few of the basic concepts of the framework. The idea is that system design consists of many small steps where design options are considered and design decisions are taken to progress the system design to its next incarnation. The framework is described in [Moneva 2011].

4. Terminology

As usual, the terms that we use are overloaded and their meaning is context dependent. We discussed the terminology that the Boderc project (see /www.esi.nl/boderc) agreed upon, as shown in Figure 11. The five key terms that Boderc participants agreed upon are *formalisms*, *models*, *techniques*, *methods*, and *tools*. Figure 11 shows that these concepts are related. Designers create a *model* expressed in a *formalism*. Tools can operate on *formalisms*. Designers apply a *technique* on a *model* to get results, e.g. to analyze performance. *Methods* provide guidelines how to use *formalisms* to create *models*, use *techniques*, and apply *tools*.


Formalisms languages/syntax: differential equations, timed or hybrid automata, finite state machines, et cetera

Models instantations of formalisms to understand, explore, optimize or verify specification or design

Techniques to get the required information from models: e.g. performance

Methods to provide guidelines how to use formalisms, create models, use techniques and apply tools

Tools to support efficient application of formalisms, techniques and methods

Figure 11. Terminology as agreed upon in the Boderc project.

5. Why modeling and modeling tools?

The forum members discussed this question several times in the breakout sessions using the presentations as input. The primary answer to this question is that models and tools help to structure technical discussions. For example, models can be used to:

- Reason about systems

- Discuss and analyze options and tradeoffs
- Validate variants
- Analyze communication relationships and dependencies
- Record results of meetings
- Aid communication about systems

Analysis typically requires a degree of formality. A formalism, e.g. notations and definitions facilitate computer based interpretation and manipulation.

We made an inventory of the types of models that presenters have shown. This inventory shows a wide variation in the kind of models shown in these presentations:

- Requirements modeling, traceability
- Concept selection models
- Architectural models
- Logical models
- Behavioral models
- Physical decomposition, modularity
- Control model
- Electronic schematic models
- Mechanical CAD model, e.g. stress
- Performance simulation
- Customer context model
- Business model
- 3D animations, communication, alternative designs, as functional model

- Models for safety qualification

Architecting teams typically make and use many different kinds of models with different purposes and different formalisms. Architects need to combine data and insights from multiple models. A hot topic is to what degree are models connected or integrated. One of the outcomes is that models are connected on a need basis, mostly with manual conversions to make the connection work. The consequence can be that different aspects, analyzed in disjoint models, are weakly related. Combining model results requires shared semantics; e.g. this is a form of tight coupling. At the same time, architects prefer loosely coupled models. One participant remarked that architects own the data that go into the models.

Architects struggle with the question, what level of modeling effort is affordable? How much effort can we afford to develop models and to integrate them? One idea is to look for a stop criterion that an inexperienced person is able to understand. The participants expect that this effort depends on domain knowledge and experience. Risk reduction is seen as a justification for modeling expenses. High risk and new initiative may justify higher expenses. Models facilitate analysis and assessment of risks, technical feasibility, and commercial viability.

The discussion triggered some new questions:

- Are people afraid of modeling?

  Most architects have experienced some expensive failures in modeling. Models promise understanding, exploration, analysis, communication, decision support, and risk reduction. However, not all models bring these expected benefits. The quality of the models depends on the competence of the model makers and the quality of the data going into the models. One of the findings of the Boderc project [Boderc 2006] was that stakeholders trust models when they understand what the model does. Lack of insight in the model is a reason to distrust its outcome.

- Are systems engineers afraid of formal modeling?

There seems to be some reluctance in the current systems engineering population to embrace formal modeling. Does conservatism cause this reluctance or are there other underlying concerns? This topic deserves another workshop.

In the last breakout session we discussed what support architects need and at what level of interaction. The support question resulted in the following wish list:

- Model integration, exchanging shared semantics
- Common data repository
    - Configuration management
    - Version control
    - Links to information sources outside the models
- Application version consistency of tools
    - Including OS versioning
- IT support
- Tool smith, efficiency; the tool smith concept comes from the software engineering world. A tool smith is someone who is fluent with tools and supports the organization in effectively applying them, among others by adapting tools where required. This concept probably can help to improve efficiency of architecture modeling.
- Degree of interaction: ideally roundtrip and concurrent engineering

We finally made a quick inventory of tools used by participants. The result is a long list (and probably far from complete): PreeVision (architecture), Matlab, ML/Sl/SF, E3cable, Catia, NX4, XDIS (communication), Symta/S, Sparx Enterprise Architect, Eclipse, IEEE architecture decision template, TFS (Microsoft), MS office, Team Track workflow, Zachman, DoDAF, MoDAF, UDDM, TOGAF, Rhapsody, Zemax, @risk, Model Centre,

6. Conclusions

Let us revisit the questions that drove the discussion of modeling:

- What are architecture models? (preferable illustrated by examples)

  This paper shows a large number of different models. A subset is considered to be architecture models. Architecture relevance, or the usefulness of the model for the architect are criteria to classify models as architecture models.

- How to model architecture?
- Do we model aspects (e.g. cost, performance) or "the architecture"?

  The examples show a wide variety of models, e.g. aspect models, technology related models, business models, structure (e.g. partitioning, interfaces, behavior, function allocation) models. We model single aspects and we create more integral models to combine multiple aspects. Architecture modeling requires a variety of models, driven by domain, business, project or program, and organization needs.

- Why would architects model?
  - To validate early?
  - Because simulation helps?

  A recurring answer is to reduce risks. However, to gain understanding, to explore, to facilitate analysis and discussion, to support decision making, are all answers that pop-up.

- When should architects model?
  - Early, but how early?
  - How far to go?

  We did not discuss the question of how early. The how far to go question did not get a full answer either. Affordability relates to the degree of risk reduction. The architect's intuition is that the effort should be balanced to the amount of reduced risk.

Acknowledgements

Several companies permitted the use of photos and diagrams in this paper, which improved readability and helps readers by showing examples. We thank Daimler, Micronic Mydata AB, and Kongsberg Groupen for their willingness to share these pictures.

Literature

[Boderc 2006] M. Heemels and G. Muller (editors) Boderc: Model-based Design of High Tech Systems, the Embedded Systems Institute, 2006, http://www.esi.nl/dotAsset/ee6a410f-0ae9-4f9f-80df-fc64cd7c8556.pdf retrieved April 22, 2013

[SysML forum 2006] http://www.sysmlforum.com/faq/what-is-MBSE.html retrieved August 6, 2012.

[Friedenthal 2007] Sanford Friedenthal, Regina Griego, Mark Sampson INCOSE Model Based Systems Engineering (MBSE) Initiative, INCOSE 2007 in San Diego http://www.incose.org/enchantment/docs/07docs/07jul_4mbseroadmap.pdf, retrieved 6 August 2012

[Karban 2008] R. Karban, M. Zamparelli, B. Bauvir, B. Koehler, L. Noethe, A. Balestra Exploring Model Based Engineering for Large Telescopes - Getting started with descriptive models SPIE Astronomical Telescopes and Instrumentation 2008, http://mbse.gfse.de/documents/SPIE-SysML-with-Copyright.pdf

[Karban 2011] R. Karban, T.Weilkiens, R. Hauber, M.Zamparelli, R. Diekmann, A. Hein, Cookbook for MBSE with SysML http://mbse.gfse.de/documents/SE2PracticesAndGuidelines.pdf

[Moneva 2011] H. Moneva R. Hamberg, T. Punter A Design Framework for Model-based Development of Complex Systems, IEEE-AVICPS 2011 in Vienna, http://www.esi.nl/publications/multiform/2011_Moneva_etal_A_design_framework.pdf

[Schwefer 2010] Gabriel Schwefer, Tool Supported E/E Architecture Development at Daimler, Vector Congress 2010, Stuttgart, 01.12.2010. http://www.vector.com/portal/medien/cmc/events/commercial_events/VectorCongress_2010/ProcessTools_1_Schwefer_Lecture_V8.pdf