Model Based Systems Engineering for Aircraft Systems – How does Modelica Based Tools Fit?

Ingela Lind Henric Andersson
SAAB Aeronautics
SE-581 88 Linköping, Sweden
ingela.lind@saabgroup.com henric.andersson@saabgroup.com

Abstract

Saab Aeronautics has chosen Modelica and Dymola as part of the means for model based system engineering (MBSE). This paper will point out why a considerable effort has been made to migrate models from other simulation tools to Dymola. The paper also shows how the models and tools are used, experiences gained from usage in an industrial context as well as some remaining trouble spots.

Keywords: MBSE; Dymola; Aircraft simulation; Model integration; Modelica

1 Introduction

Engineering aircraft systems is a complex task partly due to the factors; expensive equipment, expensive tests, long lead times, safety constraints, varying environmental conditions, e. g., temperature, pressure, and g-loads but also weight and space constraints, which may lead to high interaction level between systems, interaction between engineering domains, and finally, sensitivity to shortage of technically broad and experienced staff.

By using Model Based Systems Engineering (MBSE), much of the information regarding a system can be collected into an executable description, a model. This helps information sharing between people, encouraging cooperation over technical disciplines such as, fluid mechanics, electrical engineering and software engineering thereby helping the definitions of interfaces between systems and algorithm development of embedded systems. Integrating models from different disciplines into the executable model forces focus on the system boundaries. Models are also good tools to increase the in-depth understanding of a complex system and for training new staff. The most likely pay off is that by using MBSE, problems can be detected earlier than by using document based systems engineering, where many problems may be detected when the first test aircraft has been built.

At Saab Aeronautics, several projects for increasing the use of MBSE are ongoing. A few of these projects are partly EU financed and performed in cooperation with most of European aircraft industry (Crescendo [3], Clean Sky [4]). The ambition is to:

- detect problems within a system or an engineering domain early,
- detect problems between systems and engineering domains early,
- increase the ability to optimize design for different purposes (such as total fuel consumption),
- detect ambiguous and/or conflicting requirements early,
- reduce the amount of implementation errors detected late,
- reduce project risks,
- gain better control of model variants, model fidelity and approved usage of models,
- get more effective system engineering,
- reduce testing time and cost,
- effectively use data from tests,
- get better secondary products, such as training simulators for pilots and technicians,
- give more fun for systems engineers as happy engineers perform better.

The rest of the paper is outlined as follows. Section 2 contains a description of the model based process and different aspects of models and Modelica usage within this process. A deeper discussion on models and tools integration is presented in section 3, section 4 reports the potential future needs and section 5 concludes the paper.

2 Model Based Development

The model based development process can be described from many different viewpoints, see e. g. [8]. Here, the viewpoint from a systems engineer specialized in systems modeling and simulation is taken. In Figure 1, an overview of the process is given. The basic idea is that as much as possible of system functional and nonfunctional aspects should be tested as completely and cheap as possible to find system design and implementation errors as close to their origin as possible. Each engineer takes responsibility that his or her work is well done and checked. The practice follows typically ARP4754 [5], but the NASA Standard 7009 [7] give many useful ideas. Another driving factor is that MBSE supports multidisciplinary optimization, a fact utilized in research projects as CleanSky [4]. To make MBSE possible, a set of tools is needed. The applications for these tools cover many aspects of the engineering tasks; from requirement tracking, construction, configuration management, specification, modeling, simulation, report writing, archiving, project planning etc. There are many technical disciplines involved, most with specific specialist tools. This paper mostly focuses on the technical discipline of fluid dynamics with embedded hardware and software, but might be applicable to any mechatronic system.

The major reason to migrate from previous generation of simulation tools, which was made for simulation only, to Dymola is that Dymola supports all the mandatory aspects of tool integration, as described below. The choice of Dymola in front of other Modelica tools that also fulfills the technical requirements depends mostly on two aspects. It is owned by a large tool vendor and can thereby be trusted to live long enough (10-30 years) and there are consultants available that speaks Swedish.

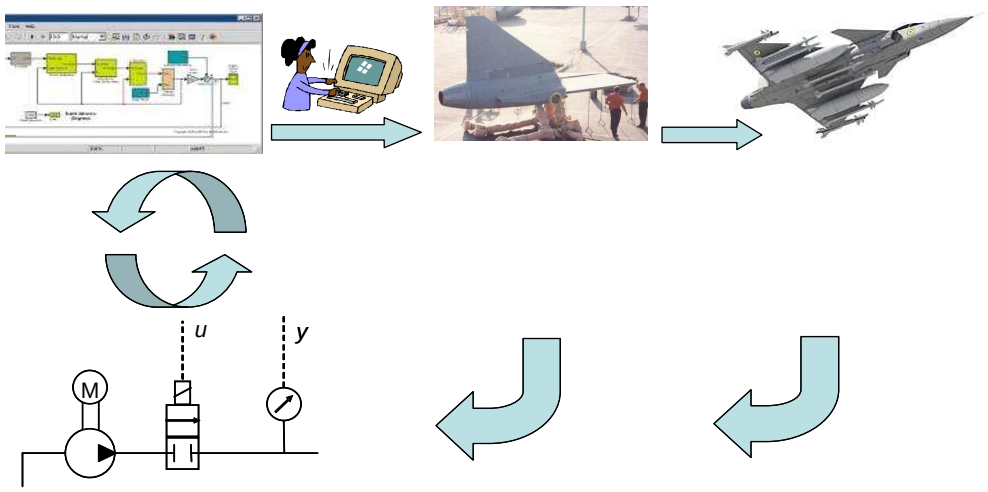

Figure 1 Model based development process. The first loop is small and fast and involves desk top simulation of one model, either control software specification or the model of the physical part, with the other model run as hosted simulation. The second loop demands more work to close the loop. Code should be made for the target computer, code for the physical part of the system might need to be exported to the simulator platform, and/or the physical parts of the rig or simulator prepared. To feed back results to the models good comparison and tuning facilities are needed. The third loop involves the airplane, which means typically expensive tests. For successful feedback, measurement, comparison and tuning facilities are needed.

2.1 Typical Parts in an Aircraft System

Aircraft systems typically involve three major types of parts; equipment, avionics with embedded software and surrounding. Equipment is things like gear boxes, valves and pipes, batteries, sensors, heat exchangers, reservoirs, etc. Avionics is computer hardware and software which fulfills requirements for aviation use. The surrounding is everything from connected aircraft systems, ambient conditions to pilot commands.

2.2 Modeling systems of physical equipment

First, the physical part of the system is modeled, from first principles using supplier data, bench test data, previous experience, geometrical data, and sometimes also results from continuous fluid dynamics computations. To make the modeling effective, it is important to have a library of mod-

eling components built for actual purpose and in an appropriate level of detail. For environmental control systems and fuel systems, Saab has together with Modelon AB developed libraries based on the stream construct in Modelica. These are commercially available [1]. It is also important to have a good tool for tuning parameters of model components to fit well with available data. Identifiability, which means that there exists a unique parameterization of the model or solution to the optimization problem of fitting the model output to measurement data [2], is often an issue, since physically parameterized components often have several ways to affect the same property. It can be hard to determine which parameters that are affected by identifiability problems for the identification problem at hand, so any means to support this task would be welcome.

The model is (partly) reused in several different environments on different platforms with and without real time requirements and with different interfaces. This requires special care in designing the model architecture and the libraries. The Modelica construct of replaceable classes is useful for making desktop simulations faster when parts of the model can be switched off before translation, which implies that whole sections of equations don't turn up as compiled code. A typical example is the model of a fuel drop tank that can be either present and attached or not present. We miss the possibility to do the same type of switch in the generated code, so that the choice of variant can be done after translation, preferably in run time, but without the code related to those sections of equations always slowing down simulation, see further discussion in Section 3.3.

It is also useful to have model switches that can turn off a complete section of the model, as the slow temperature dynamics in the fuel system model, which makes it easier to reach real time performance for the model without rebuilding it using another component library.

As the model is reused for many different purposes, the concept of power ports inherent in Modelica is essential. The higher abstraction level used for implementing models with non-fixed causality compared to other languages means that modeling effort can be used elsewhere.

It is important to build all components and the model such that an analytical Jacobian can be created. The reason is that the models for aircraft fluid systems tend to be large with several hundred time-continuous states, to include nonlinear equation systems, and tend also to be stiff. Our experience is that a completely analytical Jacobian decreases simulation time compared to a partly numerical Jacobian with at least a factor 5-10, but that still means a single typical simulation run takes between 5 and 30 minutes. We have also experienced a tendency that the solvers fail more easily if the analytical Jacobian is not available.

Using the model for the system of physical equipment, tasks as first concept validation, equipment sizing, sensitivity analysis, and performance estimation can be performed. But the involved systems are so complex that it is not sufficient to base further design decisions on these results. The control software is needed to e.g. close the loop.

2.3 Software specification

The control software is used to make sure that the system reaches its control goal and to perform safety functions such as functional monitoring (FM), redundancy management (RM), and built-in test (BIT), see further [9]. Physical limitations often make it necessary to introduce new control actions. An executable model of the control software is developed in a modeling and simulation tool. In vehicle systems design at Saab Aeronautics, Mathworks Simulink and Stateflow is used together with UML tools to develop software. The tools are used to build an executable specification of the code.

Depending on conditions such as criticality assessments, target avionics, review process, tool integration, and license model, code for target is either auto generated from the tool or hand coded using the model as specification, as reported in [12]. As all information about requirements tracing, purpose and descriptions are included in the model, it is possible to automatically generate parts of the software documentation from the software models.

2.4 Close the loop

By closing the loop by hosted simulation both in Dymola and in Simulink several tasks can be performed, as further described in [10]. By using the FMI standard [6] when generating code for the hosted part an efficient handling of the connected models is achieved. It might seem as double work to make a closed loop environment in two tools, but this means that engineers can perform their

tasks respectively in the tool which they are most acquainted with and which is most suited to do the tests and changes their task depends on. The closed loop simulation is useful for rapid prototyping both for the physical part of the system and for the control software. It supports safety assessment of the system and can be used to give input to computation of static and fatigue loads as well as for performance evaluation and detailed design of the system.

2.5 Large scale simulators

An aircraft consist of a large number of interacting systems. There are needs ranging from early interaction tests between systems to training of pilots, technicians and maintenance staff that can be met using large scale simulators. The large scale simulators involve many of the aircraft systems to a varying level of detail. Some large scale simulators include target avionics (that is, are partly hardware rigs) with real time performance constraints while others are completely software based without real time constraints. Control software can either be included as a model or as target code. Equipment models are needed, but often simplified models are sufficient for the use. The highest demands on accuracy on fluid mechanical system models in this context often comes from usage in training simulator for technicians and maintenance staff or from development tasks where the interaction between several complex systems is investigated. An interesting aspect of simulator models is that correct behavior when system faults occur is required. This means that all sensors and actuators need to have several different types of faulty behavior implemented [11].

2.6 System test rigs

The aircraft systems for environmental control and fuel management are so complex that system test rigs are necessary. The system test rigs are used to test that avionics and equipment have correct electrical interfaces, that all equipment has correct mechanical interface and that pressure drops, and other functional characteristics live up to the given specifications and work well together. Depending on the flight criticality class of the system and status of the system test rig, the system test rig can also be used for flight safety checks, often in combination with ground tests in the aircraft. As the avionics is also used in large-scale simulators where the equipment is not available, the equipment model must have an interface compliant with the real equipment.

The closed loop simulation can be used to run test before they are run in the system test rig to make sure the test will give the information needed. The results from the system test rig should be fed back into the equipment and software specification models, to improve confidence and quality.

2.7 Ground and flight tests

Ground and flight tests are used primarily to ascertain flight worthiness and for validation of requirements on aircraft level. As the use of MBSE increases we see an increased usage of ground and flight tests in order to get measurement data for model validation. At the same time, the need for ground and flight tests decreases, partly due to that more validation can be done using the model based techniques.

To use model based techniques to support the certification of an aircraft, that is, ascertain flight worthiness and validate all requirements from authorities such as EASA, is partly treated in the ongoing research program Crescendo [3].

2.8 Feedback data to models

This task needs careful planning and consideration of many aspects. First, the placements of sensors in the aircraft need to be decided upon several years to several months before use. Sensor placement is expensive to change with long lead times due to the mechanical, electrical and installation work needed, if the optimal placement is not reached at the first try. Sensors are subject to a constant revision and trade off between usability versus weight and signal storage availability, which means planned sensors are easily removed, but not easily reinserted. This makes measurement data scarce.

Most flights are non-informative from a model validation perspective. The informative flights in the outskirts of the normal envelop can be hard to reach due to weather and climate conditions, safety constraints and complex conditions to be fulfilled to reach a given state. One example is ice build up in heat exchangers, which happens in some weather and flight conditions but can be close to impossible to achieve in a dedicated flight test.

To find informative data sets in the ground and flight test measurement data base is a task that requires experience and special care to make sure that the data set is fit for use. It is often less complicated to use data from a dedicated test since the control of all conditions can be monitored with the validation task in mind.

When appropriate data sets are found these should be compared with model output. The easy part of this is to simulate the model using the same ambient conditions and the same inputs from the surrounding systems. As the complete signal interface between equipment and avionics (up to 200 signals for a single system) is not subject to monitoring during ground or flight tests it is not possible to fully provide the same conditions for the equipment model as for the real equipment. The output can be compared to the measurements to give an idea of the model quality with regard to the measured quantity. To generalize the estimate of the model quality to other quantities of interest in the complex nonlinear models is more difficult. We find the tool support at the moment not sufficient for our needs. There is ongoing research on what validation methods are industrially useful for that purpose e. g. within the Swedish National Aviation Engineering Programme (NFFP), [13].

For tuning of model parameters to make the model better fit the measurement data the Dymola Model Calibration Tool has been useful to some degree, even if a larger variety of system identification methods and better control and possibility to select optimization objectives and optimization methods is desired. When using identification, different methods sometimes give different results and which method is preferred depends on the application and the validation result.

3 Integration of models and tools

In simulation of an aircraft or any other complex product, the total model is usually composed of several sub models to be manageable. The aircraft model architecture is created and maintained in order to get explicit and clear model interfaces. It is convenient to map the "model breakdown structure" onto the breakdown structure of the represented product so that e.g. system interface definitions and responsibility allocation can be reused more easily. An example of several simulation models connected to form a larger integrated model is shown in Figure 2.

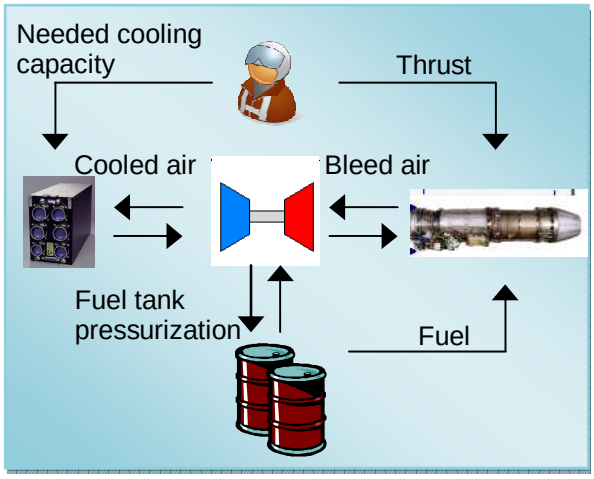

Figure 2. This is a simplified view of an integrated simulation model consisting of models for Engine, Fuel system, Environmental control system, Avionics and a Pilot model.

A smaller set of aircraft subsystem models may today be integrated in Dymola, but for larger sets of models some more specialized and powerful integration framework is still needed.

3.1 Configuration handling

Several aspects regarding configuration of models has to be handled. A simulation model is a representation of a (specified or built) aircraft system which itself is under configuration control in e.g. a Product Data Management (PDM) system. A PDM system is used for e.g. structuring, storage, change and validity control of product data related to delivery and maintenance of the products. Equipment data such as specifications, change requests of parts or documentation for certification is mature in these systems. Simulation models that represent systems/parts of the product are normally not kept within the PDM system. Information such as specification of interfaces and equations, model change requests or status accounting of models is not part of the traditional PDM scope.

Software Configuration Management (SCM) systems are suited for code management and as the simulation models are code in some format, the support for model management on code level (e.g. revision and release management) is best supported by a SCM system. Dymola supports version control of models using e.g. the SCM tool subversion (SVN).

The division of data between the PDM and SCM domains is unfortunate for many reasons, not only storage and control of simulation models. It is especially problematic in the vehicle systems modeling domain though, because data about the systems and equipment to be represented in the models are basically handled in PDM, but the model software "belongs" to SCM. So this is one of the challenges for the future to solve.

3.2 Variability

Two concepts in variability and configuration handling are variants and versions. A variant is "an option of an item which customers can choose". Versions are sequential revisions replacing each other. Variants exist in parallel, whiles versions exist in a series. If an error occurs in an item it needs to be revised, and a new version is created.

As aircrafts are developed and maintained for a long period of time, the systems usually exist in several variants and versions. Also the models exist in variants and versions, but there is usually not a straight-on mapping between the PDM development tree and the model development tree. Rigor tracing has to be maintained for models used for verification, certification, and training purposes.

Models are often parameterized, meaning that one model can be used to represent a set of aircraft system variants/versions. This implies that both the parametric model (interfaces, equations, algorithms etc.) and all the parameter sets need to be under configuration control.

One driving force for parametric models or other kinds of variability is to enable reuse. In aircraft simulation it is of major importance to reuse existing models (if possible) because the verification effort for each single model drives time and cost. The number of variants should therefore be minimized, but there are situations where the requirements are incompatible and model variants are unavoidable, such as:

- different level of fidelity
- customer specific equipment models
- security (e.g. IPR)

3.3 Binding time

Binding time describes when a variable model part or function is to be bound, i.e. selected to become a mandatory part of a simulation model instance. Possible binding times include model time (also referred to as "design time"), translation time, compile time, link time, load time, and run time.

Choosing different binding time for elements affects their properties. For example, deciding to bind two model components during translation time will yield different system properties than deciding to bind these two components at run time [14].

Example of a setting is whether the simulation is to represent a single seater or a dual seater aircraft. Another example was given in Section 2.2. It is however not sure that all models with this feature as a possible choice use the same binding time as mechanism for this setting. One model may have a translation time alternatives whiles another uses run time switch for the same variation.

Run time binding provides in general shorter turnaround time when shifting feature. There are situations when run time binding not is sufficient, for example when propriety models are integrated in a training simulator and delivered to a customer. In this case only functionality relevant for that customer is allowed to be present in that model variant. Specific customer oriented model variants are maintained and binding is done in e.g. model time.

3.4 Integration methods

There are different kinds of integration. One is integration of software- and hardware models to form closed loop simulation. This can be performed using the hosted simulation method with Dymola or Simulink as hosting environment as mentioned above.

To integrate models into a complete aircraft simulation require models from different domains often including legacy models developed with older methods and with different software generations. A large portion of the software used for simulation in the aircraft industry is for example still implemented in different versions of FORTRAN. Some suppliers of equipment also provide

their existing models to the aircraft integrator for functional- and integration verification.

To enable integration of a wider variety of model formats, an integration framework is needed. At Saab Aeronautics there are different simulation integration frameworks based on the ADA and FORTRAN languages respectively. To be able to connect models implemented in different languages one used method is to use adaptors (or wrappers). Figure 3 shows an example of architecture for model integration.

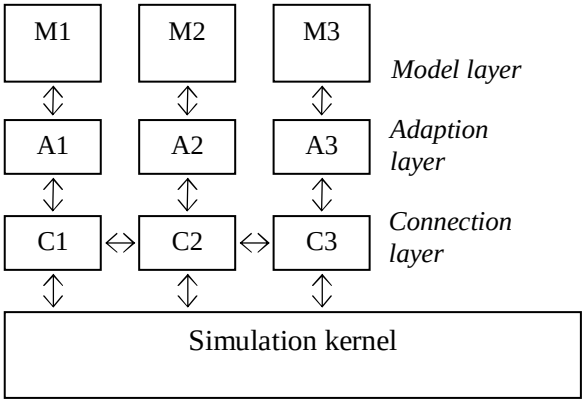

Figure 3. Example of an architecture for model integration with defined layers. The model layer is the actual simulation code in some for the framework accepted language (e.g. FORTRAN, C and/or ADA). Purpose of the connector layer is to connect models with each other and with the simulation kernel, and the adaption layer is responsible for the software language adaption (e.g. to connect C with ADA).

The emerging FMI standard [6] for model integration has the C language as a basis for software integration, which fits well with simulation code generated from Dymola. The standard also provides interface specifications in XML, which gives a powerful tool to support integration in other frameworks as it is fairly easy to map one XML scheme to another. A Modelica model created with Dymola is thereby easily integrated in a simulator with respect to the signal interface definition.

4 Future needs

Apart from the needs expressed in Section 2, the following issues are things we would like to do but have not yet found solutions to or the time to learn existing solutions.

There is a need to simulate several large system models together in a desktop environment. Each of the system models is close in size to what Dymola today handles with ease and result files get close to Windows limitations on file sizes if important signals are chosen to be stored. We do have possibility to use clusters for computation, but we lack tool support for distributing computations and results.

A related issue is support for effective code generation for multi-thread and multi-processor platforms with real-time operating systems to make complex models run able in real time.

Simple set up of batch simulations as complicated parameter sweeps or running the same simulation with several different models with good control of result files and connected with report generation is missing. Parts of it is possible to script in Modelica but especially support for detailed plot layouts and generation to reports has not yet been solved to our knowledge.

We are not yet satisfied with the auto generation of model descriptions, but this is partly due to too small amount of invested time. We would like to include more information than seem possible to do at the moment, but that might depend on that we have not yet understood how to do it.

As references for the scripting and generating capabilities related to batch simulation, plot- and document generation we have the tools MATRIXx and Matlab. These tools are according to our experience more mature in the respect of e.g. pre- and post-processing of data. There are also usable alternatives such as the python programming language [15] for data processing and spreadsheet applications for plotting.

5 Conclusions

In this paper the driver for and experiences made when shifting to Modelica as a modeling language for vehicle systems simulation at Saab Aeronautics are presented. The benefits of model based engineering are e.g. deeper insight of the systems behavior and performance as well as earlier detection of errors as compared to document based systems engineering.

For vehicle systems simulation the introduction of Modelica has largely been positive. This said there are several areas where method and tool support must be improved before MBSE and Modelica/Dymola will be the natural method to apply in all projects developing complex vehicle systems.

Areas of needed improvements are to our experience:

- better support for large size models
- support for model uncertainty and quality tracking
- support for validation of complex models using measurement data
- scripting language features for set-up, execution and post-processing of batch simulations
- better performance of code generated for real-time simulation
- code generation support for multi-thread and multi-processor targets
- generation of model documentation adapted to industry/aerospace standard

As with all methods/tools/processes there are potential for improvement, but for one of the most important ambitions we believe we have come closer to the goal; to gain happier engineers with insight and a feeling of control of their problem solving efforts.

Acknowledgments

The development methods reported in this paper has been partly inspired by and based on research funded by the research projects The Swedish Governments Agency VINNOVA's NFFP 2006-02705, 2009-01359 and 2010-01262; VINNOVA's 2007-010019 and The European Community's Seventh Framework Programmes (FP7/2007-2013) CRESCENDO (grant agreement n°234344) and JTI CleanSky.

References

[1] Modelon AB website: www.modelon.se

[2] Ljung, L.: System Identification, Theory for the User, 2nd edition, Prentice Hall, 1999

[3] Crescendo website to appear at: http://www.crescendo-fp7.eu

[4] CleanSky website: http://www.cleansky.eu/index.php?arbo_id=83&set_language=en

[5] SAE Aerospace: Aerospace Recommended Practice ARP4754a, Guidance for Development, Validation and Verification of Aircraft Systems, 2008

[6] FMI website http://functional-mockup-interface.org

[7] NASA-STD-7009**:** Standard for Models and Simulations, National Aeronautics and Space Administration, Washington, DC 20546-0001, 2008. http://www.everyspec.com/NASA/NASA+-+NASA-STD/NASA-STD-7009_16145/

[8] Steinkellner, S., Andersson, H., Gavel, H., Krus, P.: Modeling and Simulation of Saab Gripen's vehicle systems, AIAA Modeling and Simulation Technologies Conference, Chicago, Illinois, 2009

[9] Lantto, B., Jareland, M.: Model-Based Diagnosis Studies of Complex Fluid Mechanical Aircraft Systems, In proceedings of the 25th International Congress of the Aeronautical Sciences, Hamburg, 2006

[10] Steinkellner, S., Andersson, H., Krus, P., Lind, I.: Hosted Simulation for Heterogeneous Aircraft System Development, In proceedings of the 26th International Congress of the Aeronautical Sciences, Anchorage, Alaska, 2008

[11] Andersson, H.: Aircraft systems modeling - model based systems engineering in avionics design and aircraft simulation. Linköping Studies in Science and Technology, Licentiate Thesis No. 1394, ISBN 978-91-7393-692-7, Liu-Tryck Linköping 2009.

[12] Andersson, H., Weitman, A., Ölvander, J.: Simulink as a Core Tool in Development of Next Generation Gripen, In proceedings of Nordic Matlab User Conference, Stockholm, Sweden, 2008.

[13] National Aviation Engineering Programme at Vinnova website: http://www.vinnova.se/en/Activities/National-Aviation-Engineering-Research-Programme

[14] Vranić, V., Šípka, M,: Binding Time Based Concept Instantiation in Feature Modeling. In proceedings of the 9th Inter-

national Conference on Software Reuse (ICSR 2006), LNCS 4039, Turin, Italy, June 2006.

[15] Python official website http://wiki.python.org