HOSTED SIMULATION FOR HETEROGENEOUS AIRCRAFT SYSTEM DEVELOPMENT

Sören Steinkellner*, Henric Andersson*, Ingela Lind*, Petter Krus**
*** Saab Aerosystems, ** Linköping University**

Keywords: *hosted simulation, model detail levels, model structure, tool, fuel system*

Abstract

In aircraft development, it is crucial to understand and evaluate behavior, performance, safety and other aspects of the subsystems before and after they are physically available for test. Simulation models are used to gain knowledge in order to make decisions in all development stages.

This paper focuses on a special kind of simulation technique called Hosted Simulation, when a model created in one tool is generated to executable code and imported (hosted) in another tool to perform simulation.

In this paper we report on experience gained from evaluating two different approaches of hosted simulation.

Furthermore, the suitability of the two approaches can vary during different phases of aircraft development and utilization, from concept evaluation to end user support.

The model of an aircraft subsystem shows that hosted simulation is a powerful and efficient technique.

1 Introduction

Modeling and simulation in aircraft subsystem development, such as fuel, hydraulic and electrical power systems, is today an important part of the design process [1], [2]. Through modeling and simulation a defect in a function or system is found early on in the process.

An increasing part of the end system verification relies on results from simulation models rather than expensive testing in flight tests. The development of computer performance and modeling-and-simulation tools has enabled large-scale simulation. Consequently, the need for integrated models of complex systems, and validation of those, increases. Not only one model, but several interacting models with known accuracies and validity ranges, are required.

A growing challenge in modern product development is to use and to integrate simulation models from different domains. The model integration enables for example the system development, verification and validation. Difficulties lie in the fact that the models in most cases are based on different modeling techniques/tools. Submodels for aircraft system simulation can be organized into the following major categories:

- equipment models (e.g. resistors and capacitors in electricity, pipes and nozzles in hydraulics) for performance evaluation and dimensioning;
- models of the embedded software for control and monitoring of the hardware;
- models of the environment of the system.

Naturally, the models may be developed in different environments – each focused on supporting a specific engineering discipline. We use the term Hosted Simulation (HS) to denote that a model created in one tool is generated to executable code and imported (hosted) in another tool to perform simulation [3]. HS often combines several types of heterogeneous engineering systems such as mechanical, hydraulic, electrical with systems such as sensor, control and software. The resulting model structure is a good example of the complexity of aircraft system development.

2 Background

Heterogeneous aircraft subsystem development needs system modeling due to the complexity of the systems [4]. The extensive interactions between the different submodels, the equipment, the embedded software and the models of the environment, complicate simulation of one model without interacting models. One obvious example is the fuel system of the Swedish Gripen Fighter Aircraft. The specification for the fuel system control software consists of 270 pages of short text, tables and gate logic diagrams describing approximate 150 sub-modules [1]. There are hundreds of inputs and outputs and hundreds of internal variables and states that might be interesting to look at. Furthermore the fuel system equipment model has 226 state variables and more than 100 input and output sensor signals. There is no possibility to manage the simulation of just one of these two models in a correct way due to the amount of signals.

Simulation of systems can be performed in several ways, e.g. by modeling all domains in one tool, by HS or by using co-simulation where different software tools are interacting during simulation. More about the merits and drawbacks of co-simulation can be found in [6].

2.1 Modeling techniques

A challenging product development requires the most suitable tools in each engineering domain. Several tools and techniques [7] , [8] complicate the process to integrate models from different domains, for example a hybrid model containing the equipment of a fuel system and its software for control.

A hybrid model is defined as combined discrete- and continuous-time parts, in which the continuous parts are based on differential equations, and the discrete parts are updated at event times. Typically, a continuous model is used to describe physical phenomena of the system equipment, whereas the discrete model is used to define behavior of the software (see figure 1). HS is powerful when combining these models, to close the loop, which gives a hybrid system model.

In this paper a model of an aircraft subsystem, the fuel system, is modeled. The model includes both hardware equipment, modeled in a power-port tool, and software, modeled in a signal-flow tool, in the form of control system code. This is a common situation in the aircraft industry.

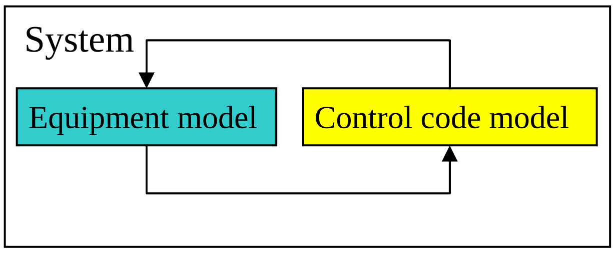

Fig. 1. The equipment and control code model in relation to the system

2.1.1 Tools for signal flow technique

Tools with good support for data flow and control system modeling are often based on the signal-flow technique. The signal flow between components is causal with predefined inputs and outputs. Model equations have to be stated as ordinary differential equations (ODE) or discrete time series.

2.1.2 Tools for power-port technique

Tool with good support for physical modeling are often based on the power-port technique [21]. The concept of power port was introduced by Paynter [20]. The power-port is the means by which physical interaction can take place between parts of a system. Mathematically a power-port is a set of acausal entities which will express connection between parts. These are often expressed as (but not restricted to) effort and flow variables, such as voltage and current, pressure and flow etc, where the product is power. The model network topological structure will remind of the real system layout. The technique supports modeling with differential-algebraic equations (DAE).

2.1.3 ODE versus DAE

In many applications ODEs are used and there are several powerful ODE solvers and analysis techniques/tools available.

For other problems, such as mechanics and electrical where physical laws are drivers, we end up with sets of DAEs [9], which are more general and more powerful for simulation models than ODEs.

Reformulation of DAEs to ODEs is time-consuming, error-prone, and sometimes impossible, while ODEs simply are a subset of DAEs.

3 Model category definitions and descriptions

The following sections describe the equipment model the control code model, and the model set-up for a total system.

Obviously, for HS, the total system model can be set-up for execution in two ways, using two different approaches, as shown in figure 2. The approaches are exemplified by a fuel system model of an unmanned aerial vehicle (UAV).

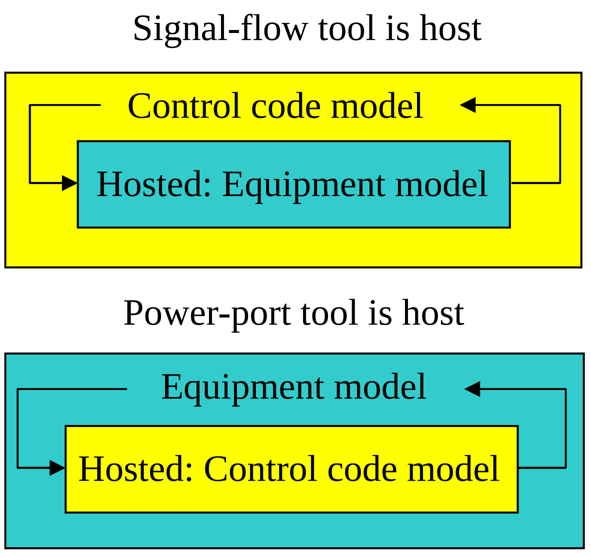

Fig. 2. Different approaches for execution.

3.1 Equipment model detail levels

A model can be modeled in different levels of detail depending on the purpose of the model and the amount of available knowledge about the real system. The accuracy of the simulation result increases with more complex model component equations but on the other hand the model execution time will increase. The needed knowledge of the real system behavior, for a model with complex component equations, can be tremendously expensive and time-consuming or almost impossible to capture.

One model grouping [15] with the respect to complexity of the model components equations is architectural, functional and behavioral level. The architectural level often uses constants or algebraic equations without dynamic response. The functional level has dynamic response but every transient physical phenomenon is not modeled. A complete flight mission is in the functional level manageable to simulate. The behavioral model tries to cover all of the most important physical phenomena. Typically, only some seconds and only parts of a complete system are simulated in a behavioral model.

3.2 Unmanned aerial vehicle model

The fuel system of an UAV has been modeled, as an aircraft subsystem example [10]. The used UAV has a fuselage length and the wingspan of 10 m and an empty weight of 5000 kg. UAVs can be remote controlled or fly autonomously based on pre-programmed flight plans. The equipment model of the fuel system is modeled in a power-port tool (Dymola [11], using the modeling language Modelica [12]) and the control code the fuel system in a signal flow tool (Simulink [13]).

3.3 Equipment model

The fuel system of an aerial vehicle is a flight critical system and has therefore redundant functions, and often robust equipment and functional design. Except from the primary task to deliver fuel to the engine there are some secondary tasks such as to be a heat sink for avionics and equipment that needs to be cooled and to provide a stable center of gravity.

The equipment model in figure 3 is modeled so that it resembles the geometrical layout of the real system and subsystems are modeled in submodels. The advantage of this approach is that non-frequent users will find the model easier to use. Flight direction is to the left in figure 3 and upper part shows right side and lower part left side of the UAV. The fuel system is symmetrical with right forward, centre and aft tank (RF, RC and RA) and analogous for left side. Forward and aft tanks feed the center tank with fuel with assistance of electrical pumps. The two boost pumps that provide the UAV engine with pressurized fuel suck fuel from

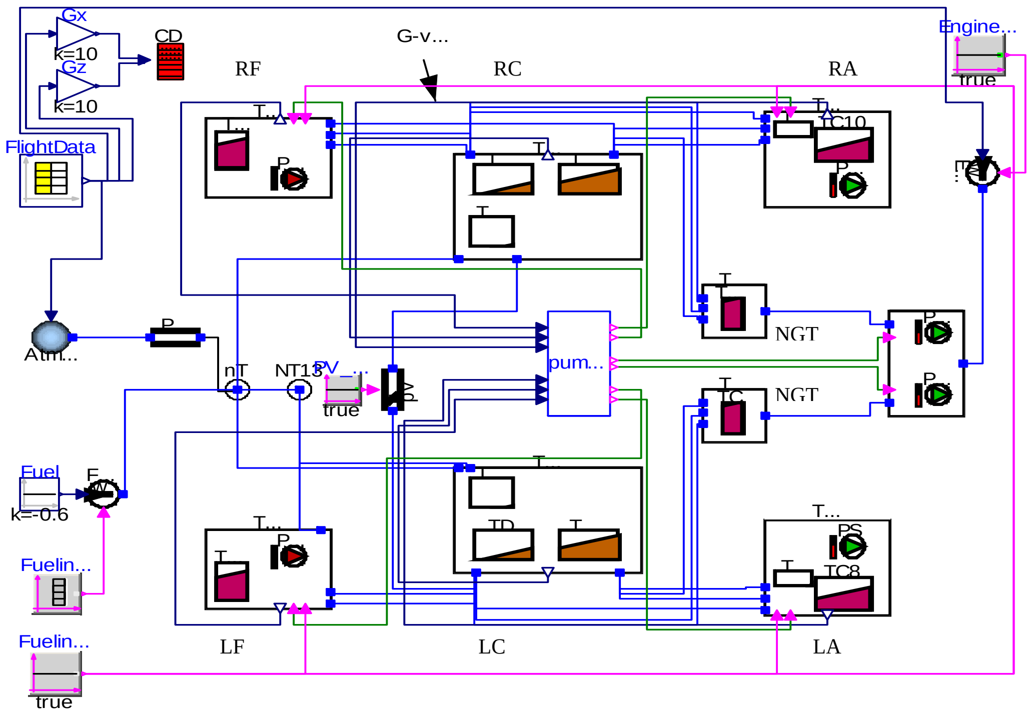

Fig. 3. Plant model. The fuel system of an UAV.

negative g-tanks (NGT) that are designed to allow inverted flying for the UAV. The NGT is connected to the centre tank. Figure 3 also shows the fuel levels in the tanks that are a function of the g-vector. Notice the symbolic arrow in the upper part in the figure 3 that shows the magnitude and direction of the acceleration in 2 dimensions.

The model has approximately 20 interesting state variables, mass and pressures in tanks, and it is a stiff model. A model that includes both very fast and very slow dynamics is called stiff. Stiff models need specialized solvers since ordinary solvers take a lot of computer time or fail completely.

The model can be classified as a functional model.

3.4 Control code model

To be able to develop and simulate the equipment model a temporary and simplified control code model is necessary to implement in the equipment model if the equipment model is developed before the control code model.

3.4.1 Simplified control code model

Usually, it is beneficial to have a simple model of the control functionality (logic and control loops) in the same language/tool as the equipment simulation model. In our UAV model, the control code, written in Modelica, looks typical as

```
 …
   /* Pump on right side*/
     if massRfw > 80 then
       if massRaft > 20 then
         if massRcentral < 334 then
           PS3_on :=true;
         end if;
…
```

and so on.

The purpose with this simplified control code model is to enable and facilitate the simulation, verifying and validating the equipment model, but also to analyze control concepts.

3.4.2 Complete control code model

An unambiguous specification of the control code is worth striving for. One way of achieving this is to let the specification be a model. The UAVs control code model is modeled in the tools Simulink and State Flow. One of the state charts is shown in figure 4.

3.4.3 Autocoding of complete control code model

Some control code specification tools has the autocoding possibility. This feature enables a minimization of the introduction of misinterpretation and faults between specification and production code. It also enables the possibility of HS when the power-port tool is the host. In the UAV control code model the tools Matlab, Real-Time Workshop, Real-Time Workshop Embedded Coder, Stateflow and Stateflow Coder has been used to produce C-code.

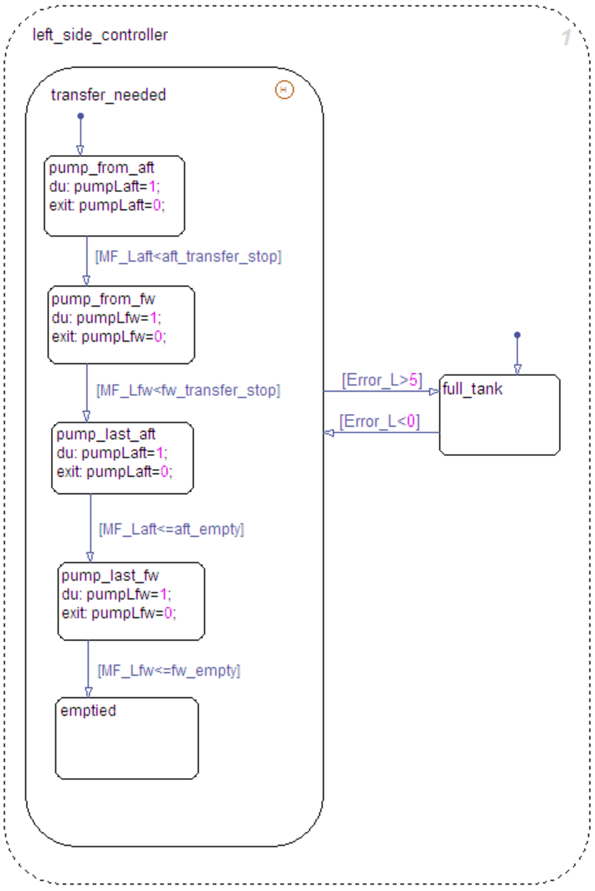

Fig. 4. A state chart from the control code model.

4.5 Model implementation

Following sections describe the implementation structure of the equipment and the control code models.

4.5.1 Signal-flow tool is host

Dymola has a prepared interface between Simulink and Dymola. The DymolaBlock in Simulink is a shield around an S-function (system-function) that is automatically created on request. An S-function is a computer language description of a Simulink block written in Matlab, C, C++, Ada, or FORTRAN and is used e.g. at creating new customer blocks or incorporating existing C code into a simulation. Initial states and parameters for the imported model can be set in the DymolaBlock. The input and output signals have to be provided manually or via script redirected to the DymolaBlock.

4.5.2 Power-port tool is host

The simplified control code model is replaced by a call to the autogenerated C-code from the control code model with 1 Hz via a wrapper function. The autogenerated C-code contains at least 3 functions, see figure 5, the wrapper

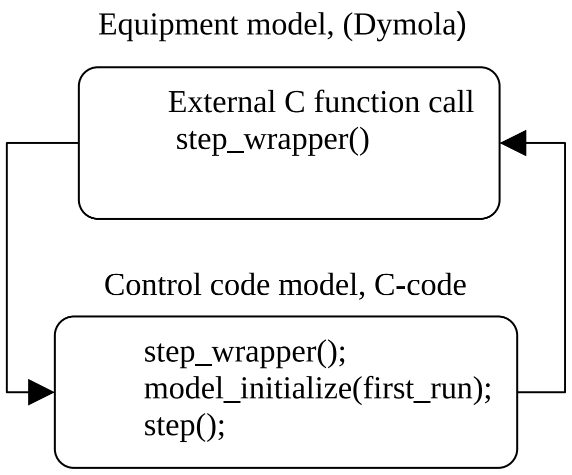
Equipment model, (Dymola)
External C function call
step_wrapper()
Control code model, C-code
step_wrapper();
model_initialize(first_run);
step();

Fig. 5. Function structure.

function, the step function, that is executed at every time step and which the wrapper function calls at every time step and the initialization function that only will be called once. The code generation tool needs, in the UAV model, a fixed-step solver. The length of the time step is not included in the control code call and therefore the frequency of control code calls and the internal time step in the auto generated control code has to be set manually. A suggestion of integration of non-trivial non-Modelica submodels to a complete Modelica model with an external model interface for Modelica can be found in [18].

5 Verification and Validation

The general definition of verification and validation are:
Verification; did I build the thing right?
Validation; did I build the right thing?
Figure 6, adopted from [1], describe verification and validation in the modeling process. The *problem entity* is the system to be modeled, in this paper the UAV fuel system. The *conceptual model* is the mathematical/logical representation of the UAV fuel system, written in a model specification. The *computerized model* is the conceptual model implemented in a computer. The definition of activities in figure 6 is according to [1]:

"*Conceptual model validation is defined as determining that the theories and assumptions underlying the conceptual model are correct and that the model representation of the problem entity is "reasonable" for the intended purpose of the model.*

Computerized model verification is defined as assuring that the computer programming and implementation of the conceptual model is correct.

Operational validation is defined as determining that the model's output behaviour has sufficient accuracy for the model's intended purpose over the domain of the model's intended applicability."

With modeling and simulation tools, such as Dymola or Simulink, the number of programming and implementation errors may be reduced. The activity *computerized model verification* is then primarily ensuring that an error free simulation tool has been used, that the simulation language has been properly implemented on the computer, and the model has been programmed correctly in the simulation language [1].

The activity *operational validation* can not be fully performed in early development phases such as the concept phase due to the need of system experiment data. However, sensitivity analysis can still be done in order to point out model component parameters that have strong influence on the overall simulation results. Partial validation of reused equipment may be available.

6 Results and experiences

When choosing a workflow and toolset for Hosted Simulation in a project, there is a choice to make regarding simulation tool(s). One should compare power-port tools (for instance Dymola), signal-flow tools (for instance Simulink) and other possibilities (for instance

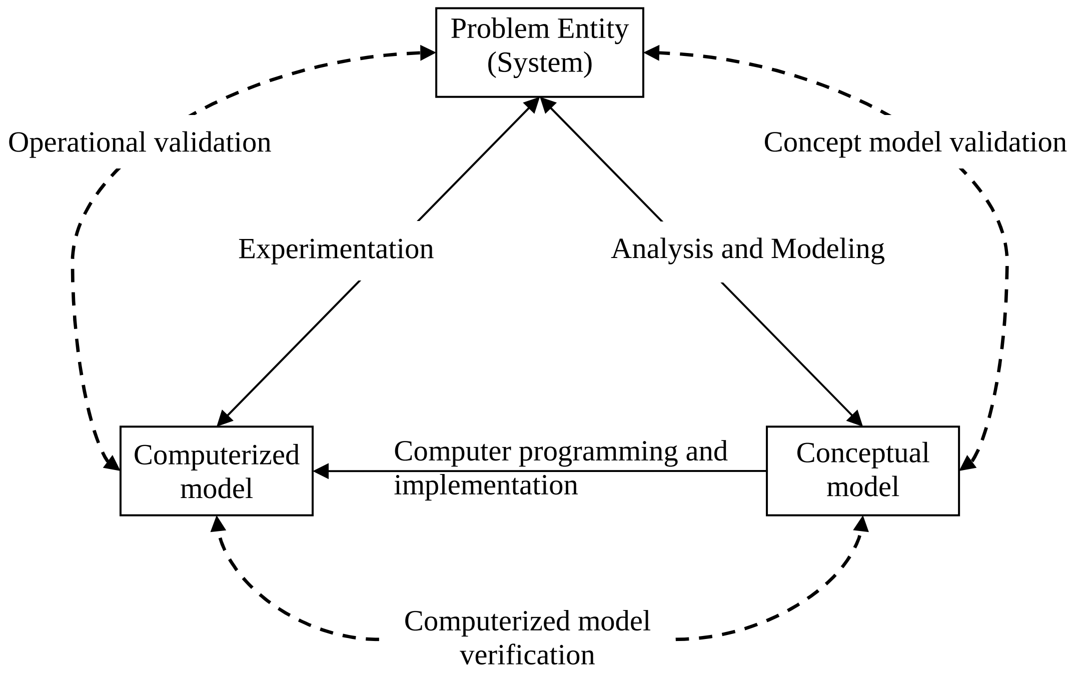

Fig. 6. Verification and validation in the modeling process.

in-house developed tools) as HS tool. Methods and tools based on SysML with simulation capabilities are also emerging, as shown in [16].

Here are some of the aspects to evaluate in such a comparison work:

- model merge during model development
- execution time performance;
- initialization of the model;
- plotting/visualizing of variables with units;
- verification and validation;
- integration to version/configuration tool;
- pre- and post-processing of data, e.g. comparing simulation runs with measured data;
- documentation

The following sections describe which one of the two techniques in figure 2 that is suitable for each of the above aspects, for a team of engineers, based on the UAV fuel system model modeling and simulation experience.

For a typical model, such as the UAV model, the system development may take several years and the product could be used by the customer for decades. This implies that even the different parts of the model have to be maintained and supported during all this time within the responsible teams, and a combination of the two techniques is therefore most suitable.

6.1 Model merge during model development

During the intense model developing phase both techniques in figure 2 is useful to achieve an efficient development modeling and simulation process.

According to the systems engineering standard ISO/IEC 15288, System life cycle processes, a product life cycle can be divided in to the stages: concept, development, production, utilization, support, and retirement. In this paper the product is the complete UAV where our model is a representation of an UAV subsystem.

The development of the equipment model starts somewhere late in the concept stage and before the control code model development, that often starts late in the development stage, [17]. The merge of the two models, a rather complete equipment model and a control code model under development, is therefore a fact first at the stage late development, but the preparation of the model merge must be included in the equipment model architecture.

One example is the wish of elimination or minimization of the manual connection of the signal interface between the control code and equipment model. Manual connection of signals, at model implementation, between the two models should be smooth in theory but is a possible source of mistakes. The simplified control code in the equipment model should be much alike in number of signals and their names to make the change to real control code model easier.

Today companies try to minimize the number of models of a system due to the large efforts of administration work that is needed and the large risk of handling faults that follow development work of a system with several models which has to be manually updated to be at equal development status. A method to avoid several different models is to use the Multi Level Approach [15]. The Multi Level Approach can easily switch between model levels in a complex system model, from an e.g. simple and super fast model for energy consumption design, a detailed model for fast network stability analysis and to a very detailed model for network quality assessment by increase of the equation complexity in the model components. The Multi Level Approach can also be used to switch between different complexities levels of the control code models, simplified and different development version of the control code model.

Another problem at model developing that must be carefully planned is where and how all sensors in the equipment model are modeled. In the UAV model, sensors may deliver an analogue voltage signal to the control code unit that has a transformation table and convert it to a useful digital signal. Sensors may not be thoroughly modeled in the equipment model, so sensor signals are simply represented by existing "physical" values in the equipment model, and therefore their values are incompatible with what the control code model needs. When combining an equipment model with its corresponding control code, as for example during HS, there are at least 3 solutions of the sensor problem:

1. The sensors are modeled with a physical approach. This solution is less common if the sensor is complex or the inner physics/logics is more or less unknown.
2. The inverted transformation table from the control code model is inserted for each sensor in the equipment model. This solution conserves the control code model structure and is therefore recommended.
3. The transformation tables in the control code model are deleted. A quick and dirty solution that sometimes is useful but can be a source of mistakes.
4. The sensors are modeled based on measurement data from sensor tests. By varying the signal to drive the measurement “noise”, this facilitates many “what if”-studies.

6.2 Execution time performance

Some simulation environments demand real time simulation. Equipment models tend to be stiff and execution time consuming. Equipment model modification efforts, such as component simplification and linearization, have to be taken or/and usage of an effective solver to achieve real time performance. It is likely that a power-port tool handles a stiff system better than a signal flow tool but not necessarily.

6.3 Initialization of the models

One way of increasing the overall simulation efficiency is to calculate in advance and save some steady-state solutions to a library, and use these to initiate the simulation model. The state variables are in the equipment model and therefore it is sometime more convenient to use the power-port tool, especially at the creation of the steady-state solution library.

6.4 Plotting/visualizing of variables and pre- and post-processing of data

This aspect is tool dependent. One drawback with HS is that variables in the hosted model not easily can be shown. To access variables in the hosted model the variables has to be connected to the model interface or be saved e.g. in a file and post processed together with data from the hosting model.

6.5 Verification and validation

Verification and validation can be regarded as a part of the development phase and therefore are both techniques in figure 2 useful. Most of the initial equipment model verification is done with the simplified control code model but end verification must be done with the control code model.

6.6 Integration to configuration management

There should not be any difference between the two techniques. If the interface definition is handled “outside” any of the two models (interface model) it should be under configuration control in relation to the other models.

6.7 Documentation

To document a model in detail the modeling tool where the model was modeled is needed.

7 Discussion and conclusion

One trend in the modeling and simulation domain is the consolidation of vendors of modeling and simulation tools. The tools have been developed and also integrated with other tools. But nothing points to the fact that modeling and simulation for different engineering disciplines will be integrated in one tool in the foreseeable future for advanced products such as aircraft systems. With this in mind and with increase of complexity and interaction between aircraft systems, this paper has presented various aspects of Hosted

Simulation and has shown that HS is and will be a useful and powerful technique.
As mentioned, HS has both merits and drawbacks but it is obvious that if both approaches defined in fig 2 are used for large systems the drawbacks can be minimized. For smaller systems or if one part of the system is small/simple it is enough to use only one of the approaches of fig 2. If the control code model is complex, approach 1 is preferable, otherwise approach 2 should be chosen.

There are more aspects related to modeling & simulation and to HS that this paper does not cover, such as tool licensing, tool management and workflow efficiency.

Acknowledgement

The research reported in this paper has been carried out within the Swedish National Aviation Engineering Research Programme (NFFP) and in close relation to the EC project SPEEDS (Speculative and Exploratory Design in Systems Engineering).

6 References

[1] Lantto B, Ellström H, Gavel H, Jareland M, Steinkellner S, Järlestål A and Landberg M. Modeling and simulation of Gripen's fluid power systems. *Recent Advances in Aerospace Actuation Systems and Components*, Toulouse, France, 2004.

[2] Gavel H, Lantto B, Ellström H, Jareland M and Steinkellner S. Strategy for modeling of large a/c fluid systems. *World Aviation Congress and Display*, Paper no. SAE-2004-01-3093. Reno, USA, 2004.

[3] Graf S. OMEGA: correct development of real time and embedded systems. *Software and Systems Modeling*, Vol. 7, No 2, 2008.

[4] Eds. Klein I, Krus P, Törne, A. *The cohsy project – complex heterogeneous systems.* Report LiTH-ISY-R-1920, Dept of Electrical Engineering, Linköping University, 1996.

[5] Ellström H and Steinkellner S. Modelling and simulation of fuel systems in (military) aircrafts. *Simulation of On-Board Systems. Royal Aeronautical society*, London, No. 447, 2004.

[6] Larsson J and Krus P. Stability analysis of coupled simulation. *ASME International Mechanical Engineering Congress*, Washington D.C., Vol. 1, pp 861-868, 2003.

[7] Sinha R, Liang V, Paredis C and Khosla P. A survey of modeling and simulation methods for design of engineering systems. *Journal of Computing and Information Science in Engineering*, Vol. 1, pp 84-91, 2001.

[8] Carloni, L. P, Passerone R, Pinto A and Sangiovanni-Vincentelli A. L. Languages and Tools for Hybrid Systems Design. *Foundations and Trends. in Electronic Design Automation*, Vol. 1, No. 1-2, pp 1-193, 2006.

[9] Najafi M, Nikoukhah R and Campbell S. L. The role of model formulation in DAE integration: Experience gained in developing Scicos. *17th IMACS World Congress Scientific Computation, Applied Mathematics and Simulation*, CDROM ISBN 2-915913-02-1, Paris, 2005.

[10] Larsson E. *Modelling of a fuel system in Modelica – applied to an unmanned aircraft.* Master thesis, ISRN LITH-ISY-EX--07/4128—SE, Dept of Electrical Engineering, Linköping University, 2007. In Swedish.

[11] http://www.dynasim.com/ Dynasim AB, 2008-05-14.

[12] http://www.modelica.org/ Modelica Association, 2008-05-14.

[13] http://www.mathworks.com/products/simulink/ The MathWorks, Inc, 2008-05-14.

[14] http://www.omgsysml.org/ OMG Systems Modeling Language, 2008-05-14.

[15] Kuhn R. A multi level approach for aircraft electrical systems design. *6th International Modelica Conference*, Bielefeld, Germany, Vol. 1, pp 95-101, 2008.

[16] Johnson T, Paredis C. J. J and Burkhart R. Integrating models and simualtions of continuous dynamics into SysML. *6th International Modelica Conference*, Bielefeld, Germany, Vol. 1, pp 135-145, 2008.

[17] Andersson H and Sundkvist B. Method and integrated tools for efficient design of aircraft control systems. *25th International Congress of the Aeronautical Sciences*, Hamburg, Germany, No. 667, 2006.

[18] Blochwitz T, Kurzbach G and Neidhold T. An external model interface for modelica. *6th International Modelica Conference*, Bielefeld, Germany. Vol. 2, pp 579-584, 2008.

[19] Sargent R. G. Verification and validation of simulation models. *2007 Winter Simulation Conference*, pp 124-37, Washington D.C., 2007.

[20] Paynter H. M. *Analysis and Design of Engineering Systems.* MIT Press, 1961.

[21] Breedveld P.C. Port-based modeling of mechatronic systems. *Mathematics and Computers in Simulation*, Vol 66, No. 2-3, pp 99-127, 2004.

Copyright Statement