METHOD AND INTEGRATED TOOLS FOR EFFICIENT DESIGN OF AIRCRAFT CONTROL SYSTEMS

Henric Andersson* **, Bengt-Göran Sundkvist **
***Department of Mechanical Engineering, Linköping University, Sweden**
****Flight Control System, Saab Aerosystems, Sweden**

Keywords: *Design environment, Flight control, Model Based Development, MBD*

Abstract

This paper describes a method and an integrated environment for model based design, simulation and analysis of aircraft flight control systems. Design of flight control systems involves domain knowledge from several different disciplines such as mass & inertia, aerodynamics, hydraulics and electronics which requires a structured method as well as a powerful environment to succeed in the control system design. The core tool in this design environment is the model editor SystemBuild which is based on functional flow block diagrams. The presented method is illustrated using the development of the Gripen fighter aircraft flight control system as an example.

1 Introduction

Modeling and simulation are methods used to achieve high quality and cost-effectiveness in the design process for several of the subsystems that are part of a modern aircraft (a/c) [9], [11].

In the design of flight control systems for advanced fly-by-wire aircraft, several requirement sources set the boundaries of the design space, see Fig 1. Requirements are captured for different aspects as described by Pratt [11]. Examples are:

- Handling qualities setting the basic maneuvering requirements.
- Aircraft operation, structural loads and flight envelope.
- Structure and aeroservoelasticity limiting the actuator speed at certain frequencies.
- Equipment constraints. E.g. control stick
- Mass, inertia and center of gravity taking fuel, payload and crew into account.
- Hydraulic or electrical power limiting the total control authority.
- Flight dynamics and nonlinear aerodynamics with uncertain properties in some regions.
- System safety implying redundancy management with potential rapid mode changes.

These requirements have to be considered during the whole product lifecycle, from conceptual and architectural design steps through every development phase to further sustained engineering work during operation. In this paper we focus on the environment for flight control system (FCS) software design and verification.

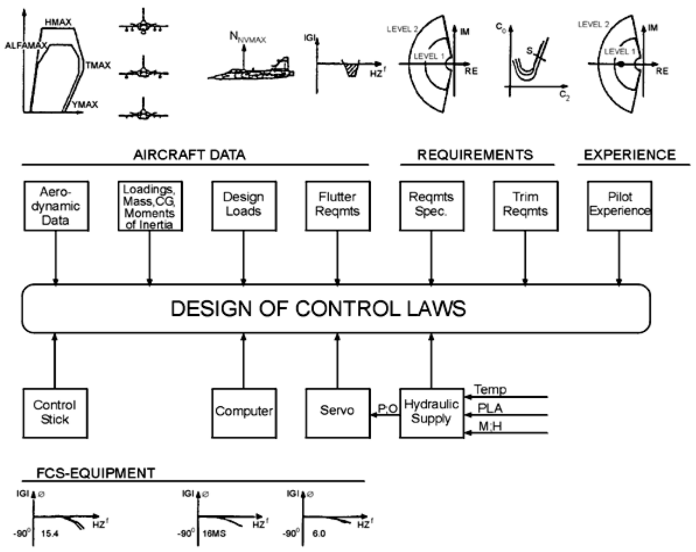

Fig 1. Requirement sources at control law design.

To be able to understand the different problem areas, how they interact and how changes in one area may affect other requirements, appropriate modeling and simulation techniques are necessary as described by Zipfel [17].

A basic air vehicle simulation model, representing an actual or a potential product, is typically composed of:

- Pilot response model
- Control laws and logic (software)
- Controls for pilot inputs
- HMI-system with information feedback through windows, panels or displays.
- Actuators transforming information to position/energy/power
- Sensors transforming position/energy/ power to information
- A/c and surrounding (physical things)

See Fig 2.

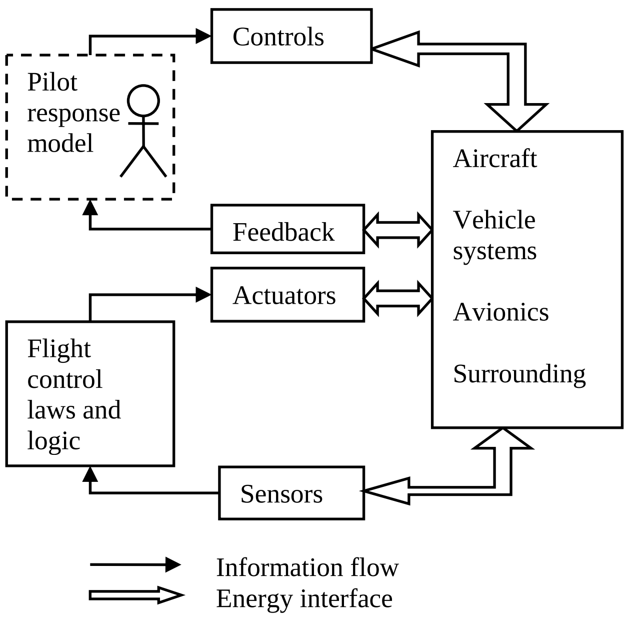

Fig 2. Basic model composition

To develop and maintain the model and to work with analysis, design and verification in an efficient way, the whole environment content and configuration (computers, simulation tools, tools integration etc.) is important. Several teams add and maintain parts of the total model for completeness. The aerodynamics group update aero-data, the hydraulics group maintain their sub-model and so on. The avionics model consist of interfaces and the functions necessary only to integrate the flight control functions, while many of the tactical systems are omitted. We here mainly discuss control law design and the transition to software specification documentation and software implementation in the operational flight program (OFP) as described in [11] by Pratt.

2 Development process

Several different types of models can be used to describe the process of product development. The V-model is a popular way to illustrate development of systems, see Fig 3. In Pratt [11] the V-model is used to describe the FCS development process.

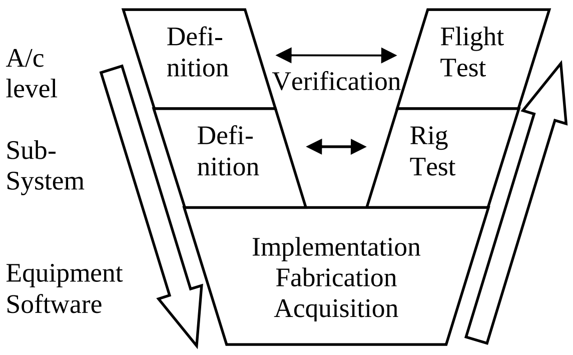

Fig 3. The V-model development process

On the left side, definition, specification and modeling activities are performed, mainly without any real parts of the product available. On the right side integration and test activities, with real parts/articles are included. The V-model tends to be rather top-down oriented, depending on the interpretation.

Another systems development model is the two dimensional model with system lifecycle phases versus process activities with visualization according to RUP [7], here the process activities are adopted from EIC/ISO 15288 [6] see Fig 4.

For flight control systems, the system safety requirements have great impact on the development process. For hardware, the safety requirements imply an architecture with redundant parallel signal channels and multiple processors and need for coordination of mode changes and integrators. Asynchronous inter-channel communication is not easily modeled, but is normally verified by test. This leads to thoroughly planned test facilities and test procedures. Process requirements on the software development work are elicited from RTCA/DO-178 [13]. This standard provides guidance for determining, in a consistent manner, that the software aspects of airborne systems and equipment comply with airworthiness requirements.

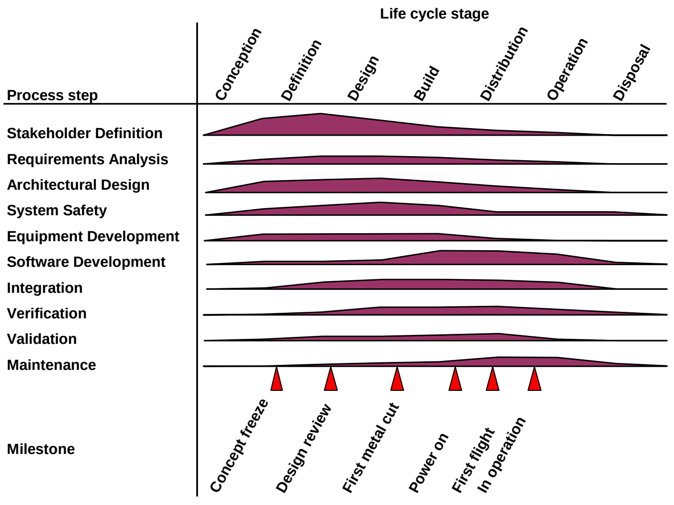

Fig 4. Process activities during product lifecycle

When the architecture is stable and the product is ready for test/usage, a relevant model to describe the work is as a change-driven process. For software implementation the different changes due to new features or known problems are best handled in iterations, by change tasks, which is further discussed in chapter 5.

For flight control law development, great flexibility is required to allow rapid design iterations, and the work may be accomplished in a less formal way as stated in Pratt [11].

3 Design Environment

A design environment usually consists of tools for calculation, simulation and plotting, and supporting tools for team communication, report generation, printing and change-/configuration control. Here we divide the environment in three main parts out of different focus in the engineering work.

3.1 Design Environment Overview

The three different environments are illustrated in Fig 5. In the first environment, graphical design, aimed for building functional models, a notation suitable for control engineers is used. The developed sub-models are transformed by automatic code generation and integrated into the simulation environment where time domain analyses are performed.

During transformation from the simulation model to the analysis and synthesis step, model reduction and linearization techniques are used. Each environment will here be described along with the methods developed to enable transition between the environments.

3.2 Graphical design environment

Function flow block diagrams (FFBD) and data flow diagrams (DFD) are modeled in a graphical editor based on the SystemBuild tool [15]. It contains signal flow elements, for example gains, multipliers and filters, but also different kinds of logical constructs such as switches and state machines.

Building or extending a model is done by selecting blocks from a predefined library of verified functions.

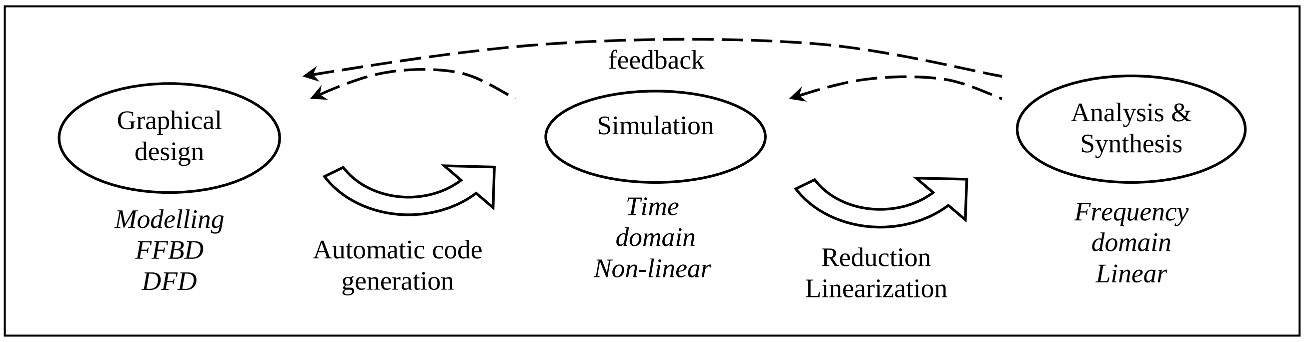

Fig 5. Illustration of the three environments including transitions for aircraft control system design

In practice it is usual drag and drop editing from a palette and connecting the blocks with arrows representing signals. The palette is structured by functionality group, for example all filters are grouped together. When using a specific block in the model, it is instantiated by choosing parameter values. Beside a signal name, every output signal may also be assigned a unit and a description. An example of connected blocks is shown in Fig 6.

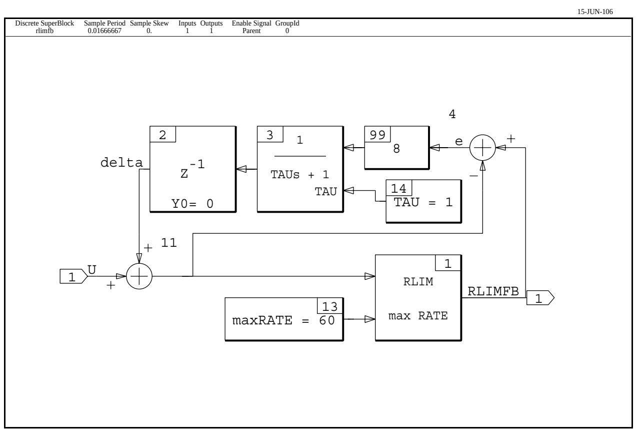

Fig 6. Example of blocks from modeling library

The layout of the blocks is made to be easily understood by control engineers, with a time-continuous representation, when appropriate. The implementation of underlying functions is however made with time-discrete techniques, for example with bilinear transformation (BLT) see Stevens and Lewis [16]. When a specific design is to be analyzed in continuous time in the environment, a transformation is performed called “inverse sampling”.

From the graphical design environment, the actual design is transferred both to simulators and to software implementation via automatic code generation and via automated documentation based on diagram printouts. This principle of “single electronic source” gives consistency of the information package containing requirements, graphical design, textual design and implementation.

3.3 Simulation environment

There are different types of simulation facilities:

- Desktop control simulation tools for offline (non real time) batch simulations.
- Handling qualities, software based, simulator with pilot in the loop.
- System simulator (rig) with a large extent of target hardware and other product-equivalent equipment present.

A picture of a handing quality type simulator is shown in Fig 7.

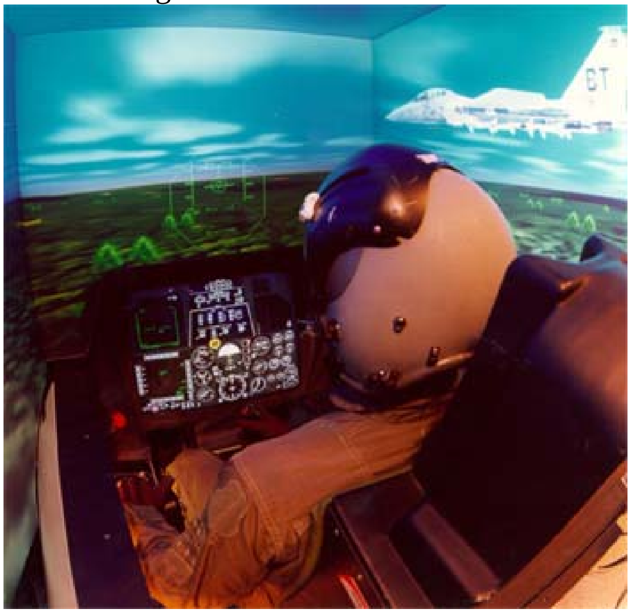

Fig 7. Handling quality type simulator “Styrsim”

3.3.1 Model development

The main modeling technique for simulation models within the control engineering domain is the input/output oriented explicit continuous time state-space representation (1) based on Ordinary Differential Equation (ODE) with time as the independent variable, where x is the state of the system model, u is input and y is output.

$$\begin{aligned} \dot{x} &= f(u,x) \\ y &= g(u,x) \end{aligned} \tag{1}$$

$$\begin{aligned} \dot{x}_0 &= f(u_0,x_0) \\ y_0 &= g(u_0,x_0) \end{aligned} \tag{2}$$

Initial condition (2) depends all on u_0 and x_0. The simulator environments are built up from other information beside the software based functionality produced in the graphical design environment. Important sources for the sub-models are the mass & inertia, aerodynamic data and characteristics of a/c sub-systems such as fuel-system and hydraulic-system. Those sub-systems are built from a mix of physical components (pumps, pipes, valves and so on) and logical components (control unit). For development of models representing those physical sub-systems, desktop vehicle-system simulation tools based on the power-port technique are used for analysis of system behavior and verification. Basic equations for this type of modeling are of the Differential Algebraic Equation (DAE) type (3), a generalized ODE that is relation oriented.

$$f(u, x, \dot{x}, y) = 0 \tag{3}$$

This method and how it is used in development of the Gripen a/c is further described in Gavel et al [3] and Lantto et al [9].

3.3.2 Execution of simulations

Batch analyses are performed by running the simulation model in a specified envelop of operating points. This is done by giving initial operating points and a number of inputs for the dynamic simulation at every operating point. A multidimensional matrix is created with selected values of speed, altitude and fuel content as well as a range of a/c subtypes, payload configurations and pilot inputs. It is also possible to introduce h/w failures, in the steady state solution, or at arbitrary time during the dynamic simulation.

In batch simulations, the pilot stick commands in pitch, roll and yaw are varied to find the most severe pilot command combination in each flight condition and for different payloads. An example of pilot command combination is shown in Fig 8.

The efficiency of batch simulations highly depends on consumed time for getting the stationary operating point (trimming the initial condition) in advance of each dynamic simulation. One way of increasing the efficiency during this part of the work is to calculate and save a library with steady state solutions in advance. States (x_{0p}) and inputs (u_{0p}) are needed for all operating points, *p*, and these are used to initiate the simulation model from the library.

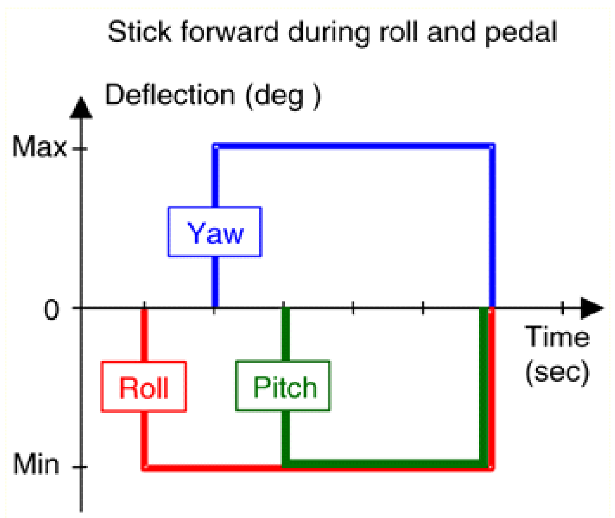

Fig 8. Pilot command combination example

Each payload combination is normally simulated with over 10000 variations in maneuvers and flight conditions. A graphical user interface is used to plot all the extreme values from the batch simulations. The extreme values are connected to the maneuver, see Fig 9.

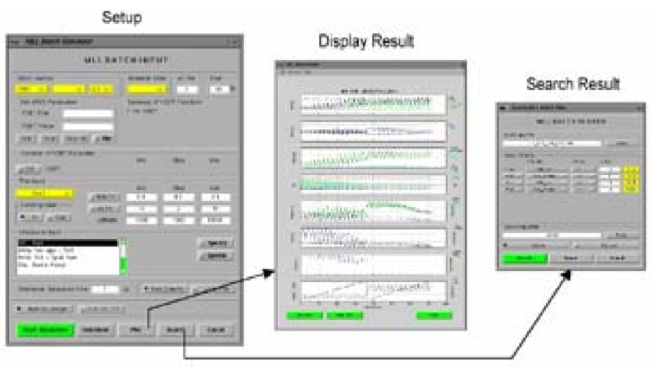

Fig 9. The desktop batch procedure.

3.3.3 Simulation analysis and visualization

The analysis of simulation results require a great deal of post processing, for example plotting, comparison between baseline and the current design (two iterations in the process) and summarizing of results.

Each batch simulated maneuver is evaluated with Matlab based scripts. An evaluation script finds maximum angle-of-attack, angle-of-sideslip, load factor, structural loads and maximum control surface deflections.

When the analysis has been completed, the simulation starts with the next maneuver. An example plot of a single simulation at one operating point in the envelop is shown in Fig 10.

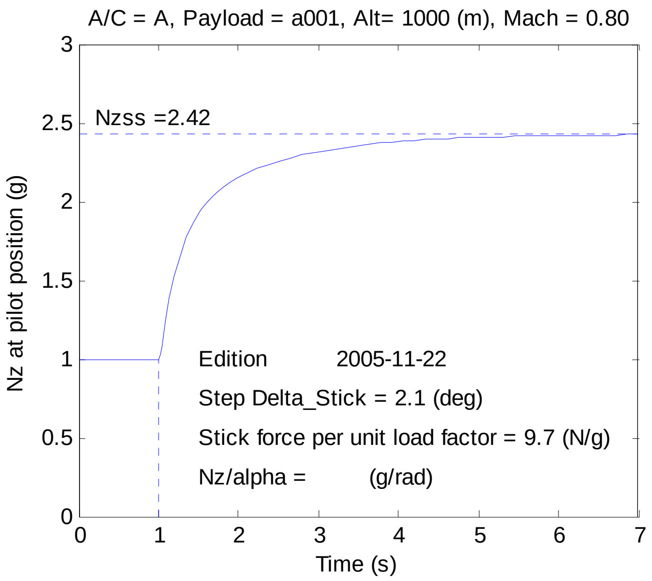

Fig 10. Example of performance verification plot

Visualization of batches of simulated data is done by summary plots for good overview. Results from all operating points (height, speed) for one a/c type and one payload combination is summarized together with information of the limits, for example Handling Quality limits, see Fig 11.

3.3.4 Model verification

When developing systems in a model based approach, quality of the models is crucial. Confidence that the models are reliable and knowledge what the weaknesses are must be kept high within the product team.

Supporting tools are needed for data handling when comparing data between models, real systems, rigs and simpler simulation tools. A part of the model verification is performed by comparing simulation results with measured data from an instrumented a/c.

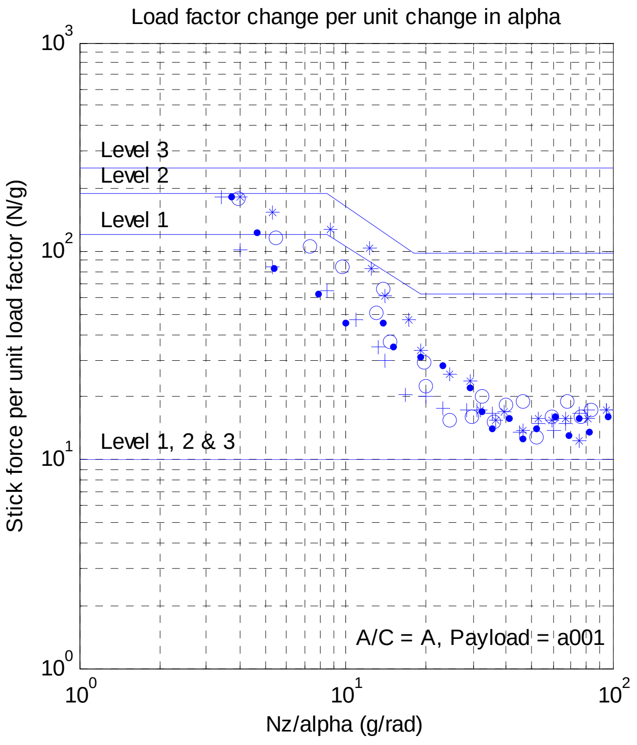

Fig 11. Example of verification summary plot

From the simulation environment, the actual control design, as well as relevant parts of the physical model, is transferred to the linear synthesis and analysis environment via model reduction and linearization.

3.4 Analysis and synthesis environment

Within the control design, a lot of the work is carried out in the frequency domain. The main input for many design methods is a linear model of the system. A standard representation is to use the linear continuous-time state-space model with the four real constant matrices *A*, *B*, *C* and *D* building a linear dynamic mathematical model (4).

$$\begin{bmatrix} \dot{x} \\ y \end{bmatrix} = \begin{bmatrix} AB \\ CD \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \qquad (4)$$

To get a linear model representing the essential dynamics for the control problem, sufficient level of fidelity, and not more, is desirable. For reduction of complexity, division into uncoupled longitudinal axes and lateral axes models is a standard method that can be found in for example Stevens and Lewis [16].

Other inputs needed for closed loop analysis are the values for gains and time constants in the controller. These are captured at the same time as the dynamics of the physical system. Every operating point of the system has its own set of controller and linear model values. Parametric variation for analysis and optimization is performed with data from the simulation environment.

4 Transition between environments

For stringent and safe transition of information, supporting tools are developed, so the transition is partly automated. Unique aspects have to be considered during transition from graphical to simulation environment as well as from simulation to analysis environment as described below.

4.1 From graphic modeling environment to simulation environment

Transition from the graphic modeling environment to the simulation environment is done by code generation, compilation, linking and integration of the actual module into the simulation tool. As this is repetitive routine work, industries have developed several automatic code generators. For the control engineering domain the following code generating tools may be mentioned;

- BEACON by General Electric [12]
- HOSTESS by Daimler Aerospace [8]
- Matlab/Simulink
- MATRIXx/SytemBuild [15]
- SAO by Aerospatiale [1]
- SCADE by Esterel [4]

The same code generator may be used to produce simulation code, as well as embedded (target) code for the flight control computer. For target code, a qualified code generator will dramatically reduce the work of software verification, code review and test.

There are certain requirements on the code for efficient use during simulations:

- Unit and description of signals and states shall be transferred, from the model, in order to be available during simulation and analysis
- The code shall be possible to initiate to arbitrary values from the simulation environment, that is, setting of the states and inputs (2).

4.2 From simulation environment to analysis & synthesis environment

Transition from the simulation environment to the analysis and synthesis environment is achieved in an automated way. The model is reduced by finding the linear dependencies between states, input- and output signals in order to extract a linear model (*ABCD*-matrices), (4) as well as the needed controller values.

Model reduction is usually needed because the simulation model has higher order dynamics and several states not relevant for analysis made with the linear model. Every operating point of the nonlinear system has its own set of *ABCD*-matrices (4), and reduction of matrices is performed by a tool. The automation enables a rapid and reliable transition to the analysis and synthesis environment for further work in the frequency domain.

5 Change control

In addition to the actual configuration items, the change control mechanism is built on a number of information objects (classes). For every new need/feature or known problem a Problem Report (PR) is defined. A list of pending PRs is used for prioritizing issues and planning of new editions as well as for stating remaining problems at release (status reporting).

A Change Request (CR) defines the change in a was/is context. Integration of modeling and configuration tools is made so that new, changed or deleted model elements are highlighted to simplify the review of each CR. All affected configuration items are listed, hardware, software and/or documents.

Every new edition of the operational flight program is defined by an Edition Definition (ED). It contains purpose of the edition and the included changes listed by CRs together with their associated PRs. Relations between these classes are shown in Fig 12.

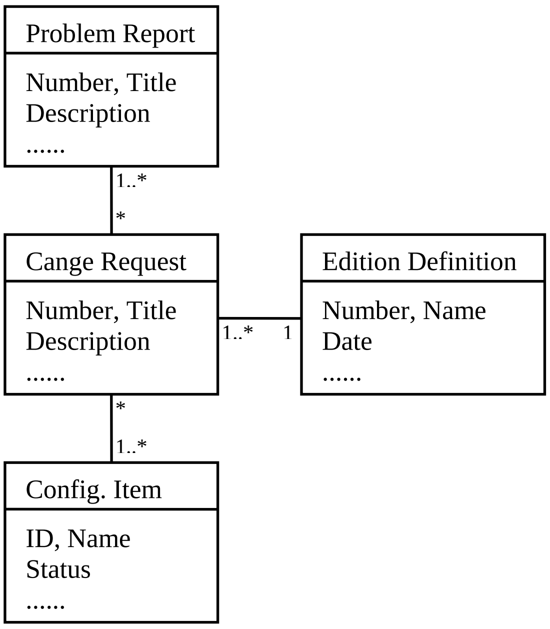

Fig 12. Information model of change control in UML

Problems and changes in the development environment are handled in the same principle manner, but in another system. The environment itself is defined as an Enabling Product according to ANSI/EIA-632 [2].

In addition to change control, the subsystem is under configuration control, including identification, lifecycle definition, version handling, baselining and document control. These activities are not described here, but further reading in the application of Gripen development is available in Hangvar and Hübinette [5].

6 Verification

Verification activities are performed at different levels according to the V-model:

- Functional verification
- Software test/verification
- System test/verification
- Flight test

Out of the over 10000 batch simulated maneuvers described in chapter 3.3 there is a selection. For each payload combination approximately 40 of the most severe and critical maneuvers are selected for the pilot in the loop simulation and flight test.

6.1 Verification of control law functions

The verification activities are defined in a verification program. If the results from the verification test with pilot deviates from the specified results, a Verification Discrepancy Report (VAR) is written. The VARs are processed in a VAR meeting with designers, test conductors and pilots. If a reported problem requires further changes to be solved, this will be agreed upon, performed, and then retested. When the verification is finished, a verification report is issued, which is used as a platform for the decision of the final software freeze at the configuration board.

6.2 Software verification

Formal software test and verification is made based on configuration freeze. Different levels of code review are made depending on actual process requirements and amount of code automatically generated. One efficient method of software verification is to compare target code with generated simulator code.

6.3 System verification

This activity is mainly performed in a system simulator (rig) with product-equivalent computers and other equipment in the loop. When verifying multi-channel systems, a drawback with this model-based method is that all inter-channel behavior/aspects and bus communication has to be separately tested, in a rig with target software, in parallel to the model-based functional verification.

7 Examples from the Gripen a/c

Two examples from development of the Gripen flight control system are described here, one control law oriented function and one in the field of flight data and navigation, but implemented in the flight control system software.

7.1 Rate limiter with phase compensation

The rate limiters in the Gripen FCS are phase compensated resulting in small time delay effects compared to conventional rate limiters, according to Saab patent [14]. Briefly, the limiter works as follows, see Fig 13.

When the rate of change of the input signal *u* is greater than the rate limit, the output *y* increases at a smaller rate than *u* and then the feedback signal *e* becomes negative. The output of the lag filter *x* will then also become negative and thus reduce the input signal to the rate limiter. If the input *u* reverses direction the output *y* will almost immediately reverse direction too, i.e. less phase shift is obtained.

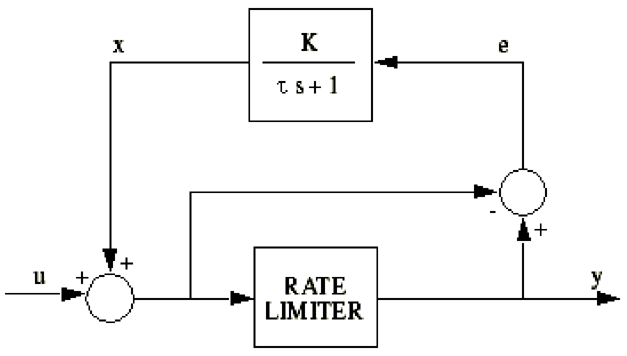

Fig 13. Phase compensated rate limiter

Implementation of the phase compensated rate limiter in the graphical design environment is viewed in Fig 6.

As presented in this paper, the Gripen design is transferred via automatic code generation to simulation environments where analysis and verification by simulation is performed. Analysis in the frequency domain is also made when appropriate. These tools and methods are part of the process and are used in a suitable extent for every increment depending on the type and scope of changes included.

7.2 Extended Kalman filter

In later versions of Gripen, the mechanical artificial horizon is replaced by a computer calculated attitude and heading, independent of the inertial navigation system (INS). The system is called Synthetic Attitude and Heading Reference System (SAHRS) and uses data from sensors already existing in the a/c. The sensor information used is a three-axis magnetic detector, true airspeed, angle of attack, barometric altitude, flight control rate gyros and load factor. The sensor data is fused together in an Extended Kalman Filter. Each sensor by itself is of relatively poor quality. For instance, the accuracy of the rate gyros is in the order of degrees per second, rather than degrees per hour as is the case in gyros dedicated for navigation use. However, when all data are combined, they provide an attitude and heading estimate with sufficient quality for its purpose; to cross-monitor the INS, and to serve as a backup in case the INS fails or data can not be displayed.

Examples of useful graphical modeling elements used within SAHRS are matrix multiplications, matrix inverse transformations and matrix transpose functions. SAHRS is published by Lundberg et. al. in [10] and is a Saab patent.

8 Discussion and Conclusions

The presented method and integrated tools form an efficient environment for rapid design and analysis of aircraft control systems. Focus is on the requirements for each separate part of the engineering environment together with the automated transitions.

Advantages with the described environment:

- Development in an analysis tool which supports design and instant analysis.
- Standardized appearance with use of a palette and underlying library functions.
- Code generation at the time of development: conformity between design and functionality, more rapid prototyping, possibility for early verification in desktop simulator.
- Automatic generation of documentation.

- Review of code and documentation is reduced.
- Automatic model reduction, simplification and “inverse-sampling”.
- Integrated version management and configuration control.

Disadvantages identified with the described process and environments:

- The development is more frequently done in a single channel system. Effects which are results of inter channel communication might be discovered late in the development.
- The natural communication between developer and implementer of the code is reduced. An important resource may thus be bi-passed during the early development stage.
- The documentation is considered to be hard to read for people without prior experience of the tool.

Further research and development is proposed in the area of asynchronous simulation to handle not only single channel analysis.

Another area is integrated component based hardware modeling and object oriented software modeling with more stringent reuse of classes between the two domains, preferably with use of SysML notation.

9 Acknowledgement

Thanks to supervisors Petter Krus and Björn Johansson at IKP, Linköping University and Stefan Andersson at Saab Aerosystems for their support. Founding of this work was provided by the Swedish Foundation for Strategic Research through the ProViking program in the area of product realization.

References

[1] Briére D, Ribot D, Pilaud D and Cahus J.L. Specification tools for Airbus onboard systems. ERA Technology, *Proceedings of the 1994 Avionics Conference and Exhibition*, Heathrow, 1994.

[2] ANSI/EIA-632. Processes for Engineering a System. American National Standards Institute.

[3] 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.

[4] Guennec A.L, Dion B. Bridging UML and Safety-Critical Software Development Environments. *Embedded Real Time Software*, Toulouse, 2006.

[5] Hangvar J and Hübinette L. *Improving the software development for JAS 39 Gripen.* Master Thesis LiTH-IKP-Ex-1571. Institute of technology, Linköping University, January, 1999.

[6] ISO/IEC 15288 Systems engineering – System life cycle processes. International Organization for Standardization, 2003

[7] Kruchten P. *The Rational Unified Process: An Introduction*. Addison-Wesley, cop. Boston, 2004

[8] Kröger A. Data flow oriented control law design with the graphical language HOSTESS. ERA Technology, *Proceedings of the 1994 Avionics Conference and Exhibition*, Heathrow, 1994.

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

[10] Lundberg M, Nordlund, P-J, Ståhl-Gunnarsson K. Synthetic attitude and heading reference for Saab Gripen. *Proceedings of the IEEE Conference on Decision and Control 3 pp. 2040-2043*. 2000.

[11] Pratt R.W. *Flight control systems, practical issues in design and implementation.* The Institution of Electrical Engineers and American Institute of Aeronautics and Astronautics, 2000.

[12] Rimval C.M. et al. Automatic generation of real-tima code using the BEACON CAE Environment. *Proceeding of the 12th IFAC World Congress*, Sidney, Australia, July 1995.

[13] RTCA/DO-178B Software Considerations in Airborne Systems and Equipment Certification. Radio Technical Commission for Aeronautics, RTCA. 1992.

[14] Rundqwist L, Elgcrona P-O, Hillgren R, Sjöberg B, Ståhl-Gunnarsson K. Method and apparatus for phase compensation in a vehicle control system. Patent number: WO95/33229. 1995

[15] www.ni.com/matrixx, National Instruments Corp.

[16] Stevens B.L and Lewis F.L *Aircraft control and simulation*, 2:nd edition, Wiley, cop., 2003

[17] Zipfel P.H. *Modeling and simulation of aerospace vehicle dynamics.* American Institute of Aeronautics and Astronautics , 2000.