UDC 621.16

P. AXMAN, T. KERLIN, D. SVAČINA, V. OPLUŠTIL, J. TOMAN

UNIS, a. s., Department of Mechatronic Systems, Czech Republic

MODERN METHODS OF FADEC DESIGN FOR AIRCRAFT ENGINES AND CERTIFICATION ASPECTS

An article describes the motivation for use of automatic code generators for the development of critical control applications in the aerospace industry. A V-cycle model based design is introduced and its advantages and development practices that leads from design of a MATLAB/Simulink [1] models to a real target application are depicted. Attention is also paid to FAA/EASA certification authorities requirements ([5], [6], [7]) with respect to a certification process of any newly designed aviation equipment. These practices are being used during the entire development cycle of an aircraft engine subsystems for small civil aircraft (category FAR 23 / CS-23).

Key words: *aircraft, engine, fadec, certification, modelling, V-cycle.*

Introduction

Software (SW) and hardware (HW) that are developed for critical application and have to meet a lot of standards that ensure their quality ([2], [3]). The certification level depends on the target application. Some experiences with certification for embedded systems in aviation are described in [4]. Generally, it can be said, that the software and hardware development and certification is time and cost consumable process. Main effort is to reduce development time and cost and ensure shorter time to market. On this account it is necessary to find new methods and approaches for SW and HW developing.

This paper describes certification process and way how to reduce cost connected with the equipment certification; modern approaches for development of control systems using V-cycle and example of a Full Authority Digital Engine Control (FADEC) design cycle.

1 Certification Aspects

The main reason for using the Rapid Control Prototyping (RCP) and development tools during the whole development cycle is to reduce time and costs of the development and certification process. We have analyzed COTS based SW and HW development tools which are fully or partially qualified for use in the aerospace industry, or they are in progress to be qualified in the near future. We have also performed an internal study which shown that COTS development tools should shorten the development cycle very effectively.

There are many qualified SW tools offered by many vendors. These SW tools cover airborne certification requirements and comply with technical recommendations and standards for safety-critical applications. Available methodologies introduce very effective processes during the SW and HW development cycle.

The best way to save resources available for the project is using integrated development tools. It is almost impossible to use only one tool for all activities during the project development, but it is likely to use such tools which can cooperate among each other, as is shown in fig. 1.

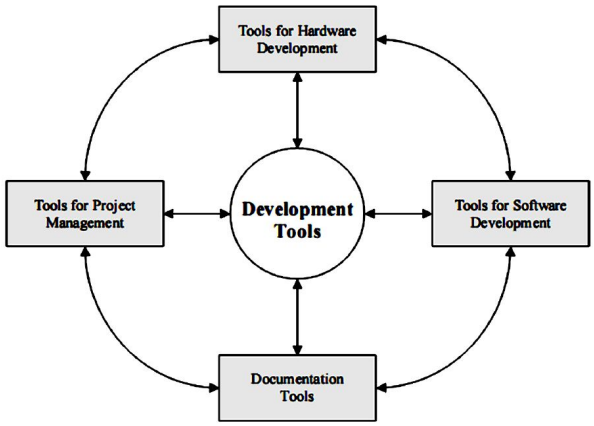


Fig. 1. Cooperation among the development tools during the certification process

An example of a commercial tool chain is shown in Figure 3. These development tools can be fully standardized for creation of higher quality and more reliable software, e.g. C, C++, Ada, Java compilators and automatic code generators, which are usually a part of Integrated Development Environment (IDE), such as SCADE, MATLAB/Simulink, NI LabView, etc. Ability to reuse certified SW parts and artifacts within the software modification process for use in another application (especially RTOS like LynxOS-178, PikeOS, Integrity-178B, etc., see more in [2], [3]) leads to cost reduction and time effective development because SW verification

© P. Axman, T. Kerlin, D. Svačina, V. Opluštil, J. Toman

and code tests are then less time consuming processes. Static and dynamic code testing tools are available commercially (CANTATA++, CodeTest, VectroCast, etc.).

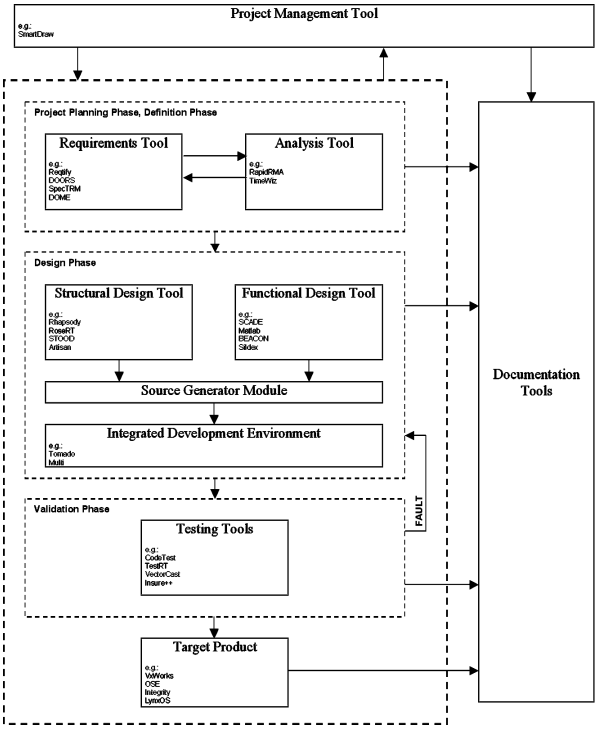


Fig. 2. Model of SW development process and its impact on SW development tools application

A certification by FAA/EASA agencies is required for both simple (based on CPU and simple electronic components) and complex hardware architectures (based on PLD, FPGA, ASIC micro-coded technology). Using simple hardware architecture with a standard core CPU (Freescale, ARM, TI, etc.) could simplify the certification process very dramatically because of proven architecture and reference projects that have passed the certification process formerly. The same situation is with COTS bus technologies - these are being used in the automotive industry for a long time and their safety, performance and reliability has been sufficiently proven [6], [7], [8].

Thus, wide enforcement of COTS components and data buses into the aerospace sector is only a matter of time.

1.1 RTCA/DO-178B and RTCA/DO-254 standards

The development of every equipment for use in aerospace has to be approved by FAA or EASA certification agencies. The two major standards in the aerospace industry that describe all the requirements and recommendations for successful certification are the RTCA/DO-178B – Software Considerations in Airborne Systems and Equipment Certification [4] and RTCA/DO-254 – Design Assurance Guidance for Airborne Electronic Hardware [5]. These standards expect from developers to describe the whole development cycle in order to assure reliability, data integrity, performance, proper development cycle, system configuration management and continuous airworthiness. As a resulting benefit there is identification of bottlenecks and prevention of fault states.

On the other hand, the drawback of these two standards is a necessity to elaborate a comprehensive amount of documentation, both for the SW and HW development.

List of the SW development documentation in accordance with DO-178B is given below:

- Plan for Software Aspects of Certification (PSAC)
- Software Development Plan (SDP)
- Software Verification Plan (SVP)
- Software Configuration Management Plan (SCMP)
- Software Quality Assurance Plan (SQAP)
- Software Requirements Standards (SRS)
- Software Design Standards (SDS)
- Software Code Standards (SCS)
- Software Requirements Data (SRD)
- Software Design Description (SDD)
- Software Verification Cases and Procedures (SVCP)
- Software Life Cycle Environment Configuration Index (SECI)
- Software Configuration Index (SCI)
- Software Accomplishment Summary (SAS)

The minimum software life cycle data to be submitted to a certification authority is:

- Plan for Software Aspects of Certification (PSAC)
- Software Configuration Index (SCI)
- Software Accomplishment Summary (SAS)

The regulation concerning retrieval and approval of SW life cycle data related to the type design applies to:

- Software Requirements Data
- Software Design Description
- Source Code
- Executable Object Code
- Software Configuration Index
- Software Accomplishment Summary

List of HW development documentation in accordance with DO-254:

- Plan for Hardware Aspects of Certification (PHAC)
- Hardware Development Plan (HDP)
- Hardware Verification Plan (HVP)
- Hardware Configuration Management Plan (HCMP)

- Hardware Process Assurance Plan (HPAP)
- Hardware Requirements Standards (HRS)
- Hardware Requirements (HR)
- Hardware Detailed Design Data: Top , Level Drawings (TLD)
- Hardware Detailed Design Data: Assembly Drawings (AD)
- Hardware Detailed Design Data: Installation Control Drawings (ICD)
- Hardware Traceability Data (HTD)
- Hardware Acceptance Test Criteria (HATC)
- Hardware Configuration Management Records (HCMR)
- Hardware Process Assurance Records (HPAR)
- Hardware Accomplishment Summary (HAS)

The minimum hardware life cycle data to be submitted to a certification authority is:

- Plan for Hardware Aspects of Certification (PHAC)
- Hardware Verification Plan (HVP)
- Hardware Detailed Design Data: Top , Level Drawings (TLD)
- Hardware Accomplishment Summary (HAS)

As is shown above, the list of required documentation is really comprehensive and it is – together with strict coding practices, test sets and system verification – the most disincentive issue during the development cycle of any new equipment/technology for the aerospace industry, because it requires strong experience, deep know-how and well managed certification practices.

2 Development of Control Systems

The development procedures and practices in the aerospace industry have originated on those used in the industry and automotive sector. A process that describes steps and linkages between individual development stages of the project has been established over the time. This process is often called a ”V-cycle”.

2.1 V–cycle

The V-Cycle is a graphical construct used to communicate a model-based software development methodology. The advantages of V–cycle lies in its inherently intuitive nature, easy reuse of model and portability across multiple platforms. Model-based control design is a time saving and cost-effective approach, allowing engineers to work with a single model in an integrated software environment. The graphical representation of the V-cycle is shown in Fig. 3.

The complete design consists of particular steps, such as: control design, rapid control prototyping, target implementation, hardware-in-the-loop testing and calibration.

Within the function design stage, the modelling and computer simulations of closed-loop system have been done. A mathematical model of both the controlled system (so called plant model) and a controller (ECU) are necessary at this point. The important thing is that the control algorithms are developed as symbolic models, not in a C-code.

When the synthesis of the ECU is finished and the results of simulations are well, the engineers start with verification of ECU’s algorithms in ”real-time” on a real-time hardware. This stage is called a rapid control prototyping (RCP).

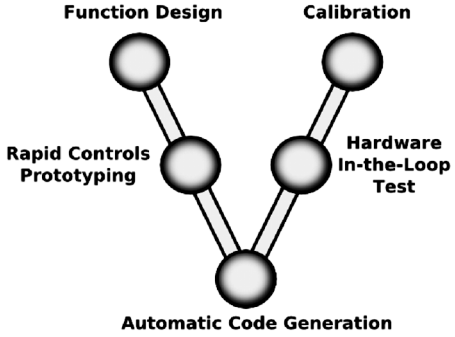


Fig. 3. The V-cycle development process [1]

RCP is a process that lets the engineers quickly test and iterate control strategies on a real-time computer with either real or modelled system-under-control. The computer model is used in case where inadequate action of ECU could cause a damage of equipment or endanger lives. The biggest advantage of using the integrated software environment for modelling, simulation and also function prototyping is that the control engineer does not have to be a C-code expert nor have enough skills to port the C-code to a real-time target. By virtue of an automated build process the RCP systems do this work for them.

In the next stage, the target code for the ECU is automatically generated by a special software, which reads math model files and generates a compile-able code that replicates the behaviour of the model. This dramatically reduces the implementation time and, in addition, the engineers have systematic consistency between a specification and production stage. Moreover, improvements to the ECU could be easily added, even after implementation of an initial code. The time for hardware-in-the-loop testing is coming on once the ECU is programmed.

Hardware-in-the-loop is a form of a real-time simulation that differs from a pure real-time simulation by addition of a ”real” component into the loop. This component might be the ECU or the real system-under-control. The current industry definition of the hardware-

in-the-loop system is that the plant is simulated and the ECU is real. The model of the plant (and the simulation HW also) is the same like in the stage of RCP.

The next step is a calibration, which is a process of optimizing or tuning real control algorithms to get the desired response from the system. A calibration tool is a combination of a hardware interface and a software application that enables the engineer to access and change the ”calibration variables” in the ECU. Typical components of control algorithms which need calibration are look-up tables, gains and constants. The structure of the control algorithm is not changed during the calibration process.

2.2 Implementation of the V-cycle

The typical V-cycle development process is based on a software development tools such as MATLAB and dSPACE. These tools provide a seamless transition from a block diagram to a real-time and target hardware.

Particularly, for function design is mostly used MATLAB, Simulink, Stateflow and other toolboxes for MATLAB. These tools together comprise a complex software package that forms the core environment for mathematical computation, analysis, visualization, algorithm development, etc. MATLAB is a high-level technical computing language and interactive environment that enables performing computationally intensive tasks such as algorithm development, data visualization, data analysis, and numeric computation. Simulink provides an interactive graphical environment and a customizable set of block libraries that let the user to design, simulate, implement and test a variety of time-varying systems. With Stateflow, you can integrate state diagrams into Simulink models.

During RCP stage the Real-Time Workshop (RTW) and Stateflow Coder (SC) automatically generate a C code from Simulink block diagrams and Stateflow systems. And here comes into play a dSPACE Real-Time Interface (RTI) which is a link between a dSPACE hardware and the development software from Mathworks. RTW/SC generates the model code, while RTI provides blocks that implement the I/O capabilities of dSPACE systems in Simulink models. Then the real-time model is compiled, downloaded and started automatically on the real-time hardware. The program can now be controlled and instrumented by the GUI application – ControlDesk. This is referred to as an experiment control.

The system-under-control could be also simulated on a real time hardware, especially in case of very complicated systems, such as e.g. engines. Many different types of HW simulators that cover all the different requirements (such as computational power, I/O interface, data bus systems, etc.) are provided with the simulations tools. The generated code could be optimized for fixed- or floating-point operations, in accordance to a certain processors. Versatile code configuration options ensure that the produced code copes with all the processor constraints.

Hardware-in-the-loop stage is closely connected with the next stage – calibration.

3 FADEC Development Cycle

FADEC is the most important control authority on the aircraft. The new and modern approach for designing of the engine control unit brings:

- Shorter developing time,
- Reducing time for code testing,
- Reducing cost for prototypes manufacturing,
- Higher quality of the application code,
- Effective support of certification, etc.

The design cycle of the FADEC will be described on the Complex Power-plant Control System (CP-CS) for small aircraft that is designed in a frame of the project CESAR. Power control system configuration for small aircraft is shown in Fig. 4, its block diagram is shown in Fig. 6. The power of the jet turbine is control by the dual channel FADEC that cooperates with Fuel Control Electrical Interface Device (FCEID) and Propeller Control Electrical Interface Device (PCEID). The FADEC can be back-upped by the manual control system.

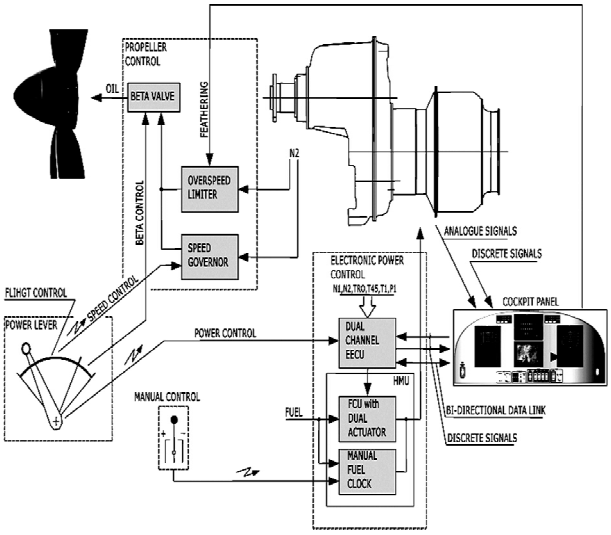


Fig. 4. CP-CS system configuration for small aircraft

Generally the FADEC model based development consists of the following steps:

- Engine and control system requirements and their decomposition
- Mathematical modelling
- Model integration and simulation (Model in the loop - MIL)
- Automatic code generation and verification

[8]

- Software in the loop (SIL) testing
- Hardware in the loop (HIL) testing
- Target platform implementation (Processor in the loop - PIL)

The graphical relation among particular steps of model based design is shown in Fig. 5.

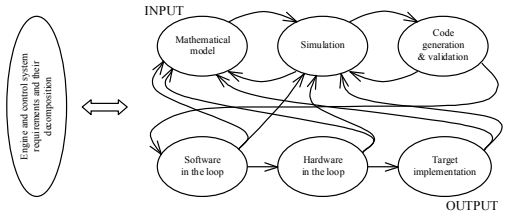
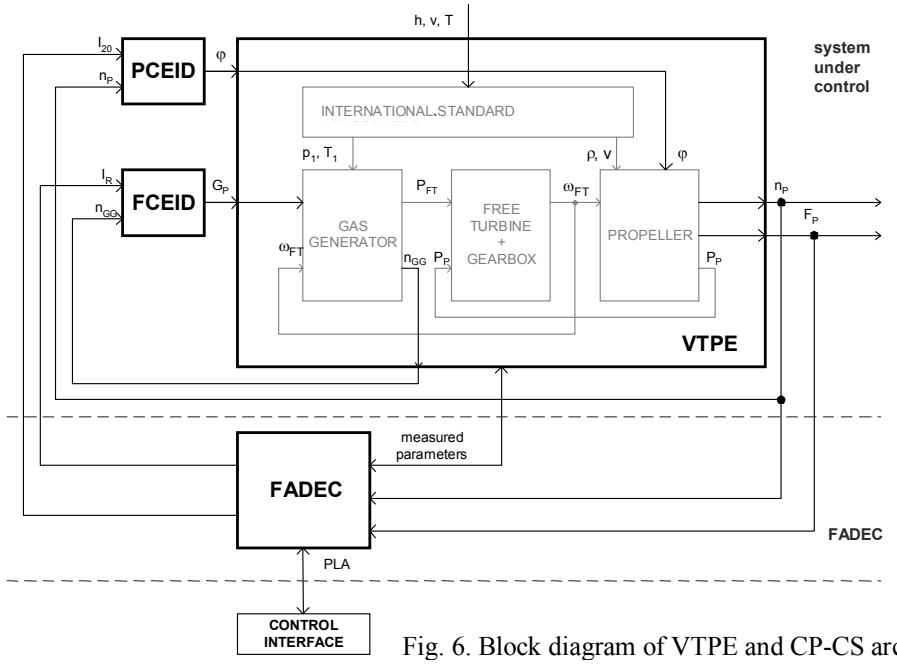


Fig. 5. FADEC development cycle



FCEID – Fuel Control Electrical Interface Device
F_P – Propeller thrust
G_P – Fuel flow
h – Altitude
I_{20} – Control current of FCEID
I_R – Control current of PCEID
n_{GG} – Gas generator speed
n_P – Propeller speed
P_{FT} – Power transmitted from gas generator to free turbine
PCEID – Propeller Control Electrical Interface Device
PLA – Power Lever Angle
P_P – Power of propeller
p_1 – Air total inlet pressure
T – Ambient temperature
T_1 – Air total inlet temperature
VTPE – Virtual Turbo Prop Engine
v – Air speed
φ – Propeller pitch
ρ – Density
ω_{FT} – Free turbine Angular speed

Fig. 6. Block diagram of VTPE and CP-CS architecture

3.1 Mathematical modelling

A model formulation of controlled system is an essential part during the stage of its control algorithm design. The model is used for examination and prediction of the behaviour of the real system. Real system could be very expensive.

A model of the system is based on the mathematical description. In engineering disciplines the mathematical model is usually described by a set of algebraic, differential equations, the transfer functions or the state matrixes.

These relations are mostly derived either by a mathematic-physical analysis of the system's phenomenon or by an experimental examination of the real system. Within the modelling of very complicated systems both approaches are combined. The aim is to get as precise model as possible, but also as simple as possible. These two requests go unfortunately against each other – the more precise model is more complicated.

The CP-CS model is very complex and highly non-linear system. The physical phenomenon involved to cover domains such as solid and fluid mechanics, thermodynamics and electromagnetism.

All the model parts are based on the mathematical description of the each part, provided by their designers.

3.2 Model integration and simulation

The simulation is a way, how to verify the behaviour of a control system that includes its environment without real hardware.

The models were created in MATLAB / Simulink that is a comprehensive software package that form the core environment for mathematical computation, analysis, visualization, algorithm development, model-based design, etc. The schematic drawing implementation of the real hardware and its mathematical models to the Simulink is shown in Fig. 7.

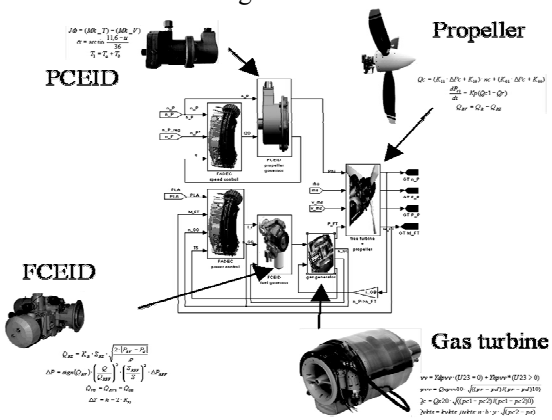


Fig. 7. Mathematical model implementation to the simulation software (Simulink)

The structure of the VTPE model is based on splitting of the whole engine into two basic parts, which can be solved separately. These main parts are: gas-generator (inlet, compressor, combustor and turbine) and power turbine with gearbox and propeller together. There is only thermodynamic power linkage between these two parts, the only hand over variable is the power transmitted from gas-generator to free turbine P_{FT}. Due to this fact the complexity of model is markedly decreased. The MATLAB / Simulink representation of the VTPE that is depicted in picture Fig. 6 is shown in Fig. 8. The VTPE model is precise enough to for examination of its behaviour during flight, for different aircraft speeds, heights, power extractions and outside conditions. The start of the engine, reverse mode, taxiing and feathering are not possible to simulate.

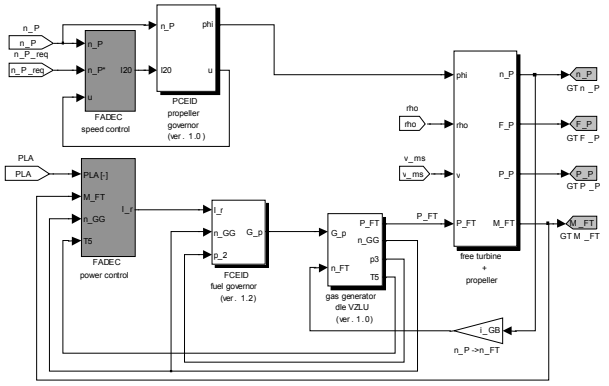


Fig. 8. Simulink diagram of VTPE and CP-CS architecture

All necessary climatic variables are computed on the basis of International Standard Atmosphere model (ISA). The input blocs of the ISA model could be set to a constant or to any time-dependent curve (e.g. typical flight profile could be simulated).

Propeller Speed Control

This part of FADEC must keep propeller speed at constant speed throughout the whole range of inputs (e.g. for changing value of power transmitted from gas-generator to free turbine and for all possible climatic conditions).

System under control is ‘propeller governor’ + ‘free turbine + propeller’ and control signal is control current of ‘propeller governor’ I_{20}. The control structure is cascade, with inner φ feedback loop and outer n_P feedback loop. Instead of measuring φ directly, the value of u is measured, which is directly proportional to φ.

The overall architecture of this structure is shown in Fig. 9.

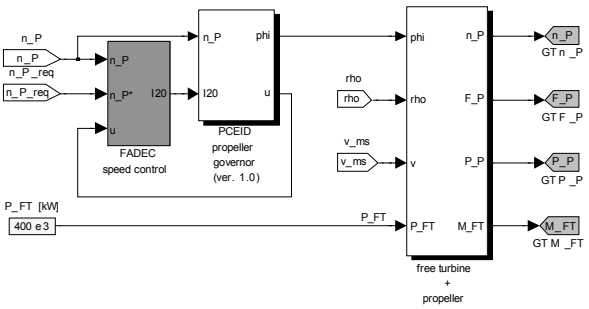


Fig. 9. Simulink scheme of the propeller speed control

Requested propeller speed is set with respect to Propeller Speed Switch, with which pilot can set one of the three different propeller states (speeds). This value is restricted by a rate limiter, which doesn’t allow too steep changes of this value. The difference between requested and actual value of n_P is processed by and PI+AW (PI + antiwind-up) controller, whose output is requested value of φ. The error signal of φ is processed with another (PI2) controller, which produces control current I_{20}. It supplies an electromagnetic actuator which drives a pilot valve of propeller governor that changes an amount of oil flowing to or from propeller head resulting in change of φ and change of n_P too. The PI+AW and PI2 were set in respect of achieving as good response to control signal as possible.

Power Control

Power control should ensure proper power of the engine, particularly proper power on the free-turbine shaft P_{FT}, with respect to Power Lever and throughout all possible climatic conditions. It also checks important parameters and doesn’t allow them to overcome secure values. System under control consists of ‘fuel governor’ with ‘gas-generator’, control signal is a control current of ‘fuel governor’ I_r. The control structure comprises n_{GG} feedback loop with PI controller and some blocs providing limitations.

But because measurement of the power (or torque) is not precise enough for using it in the feedback control, the speed of gas-generator is used instead (power of free-turbine is basically proportional to the speed of gas-generator).

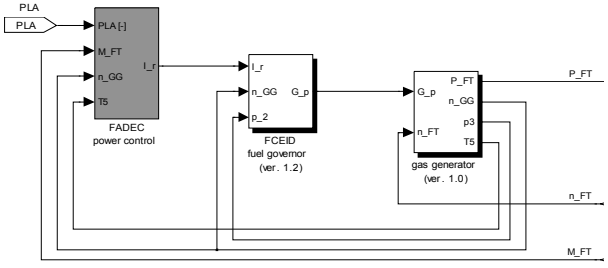
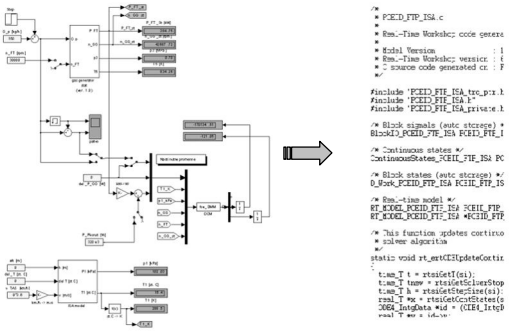


Fig. 10. Simulink scheme of the power control

3.3 Automatic code generation and verification

Automatic code generation software is an extension tool that can create executable code from a model. Real-Time Workshop (RTW) is a tool that can be used for automatic code generation in MATLAB / Simulink. The RTW generates stand-alone C code for proposed algorithms modelled in Simulink. The resulting code can be used for accelerated simulation which mostly contains software in the loop and hardware in the loop simulations. The code can be tuned and monitored by these simulations.

After automatic generation the build-in verification tool can locate and test dangerous parts of the generated code and prevents the possible accidents. The code can be tested by external verification tools like Cantata C/C++/Ada as well as.



Model in Simulink Generated code

Fig. 11. Automatic code generation

Note: The model part that contains algebraic constraint blocks has to be replaced by numerical iterative calculation (showed in 12).

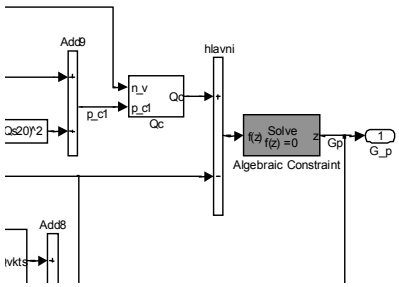


Fig. 12. Example of model with algebraic constraint block

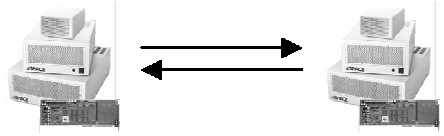
3.4 Software in the loop testing (SIL)

SIL test is proceed by the MATLAB / Simulink tool. SIL tool is control systems simulator for temporal and functional simulations. The behaviour of a control system depends on the proposed architecture and on the target hardware where the FADEC software is implemented. On this account the results given from the SIL tests can analyse only model behaviour, nevertheless the SIL tests are important step for the finding of model faults before model implementation into the hardware.

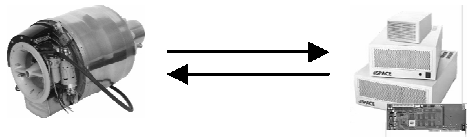
3.5 Hardware in the loop testing (HIL)

HIL is a kind of testing to validate the interactions between the designed software and the test or real hardware. The HIL tests can reduce the number of expensive prototypes. The HIL test is performed by the dSPACE environment. The HIL testing offers to use following test combination:

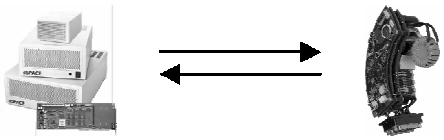
- Turbine model × ECU model



- Real turbine × ECU model



- Turbine model × Real ECU



All test combination can be realized on the created dSPACE workplace.

The dSPACE is used as a development environment that ensures implementation created models and generated C code to the real (evaluation) hardware. Controlling of the test procedures is provided by the Real-Time Interface (RTI) tool. RTI provides tool for controlling panel creation.

3.6 Target platform implementation

The proposed and generated FADEC application code will be finally loaded to the target platform. The target platform can run either as a standalone application (without OS) or as a program module in OS or RTOS. For both types of output code representation it is necessary create link interface that allows running the control algorithms.

4 Conclusion

Model based design for engine control system was approved. Main advantage of presented approach consists in development time and cost reduction. This approach supports very effectively certification process as well as. Models were created and simulated for a virtual system and will be verified on a real CP-CS.

Acknowledgement

Development of Complex Power-plant Control System (CP-CS) for small aircraft was supported by the European Union, FP6 IP research project No.: 30 888, "CESAR – Cost Effective Small Aircraft". Requirements, mathematical descriptions of particular sub-systems were provided as know-how by CESAR partners (Ivchenko-Progress, VZLU, Jihostroj, PBS and UNIS).

References

1. MATLAB SW producer: http://www.mathworks.com [cit. 2009-05-20].

2. RTCA/DO-178B: Software Considerations in Airborne Systems and Equipment Certification. RTCA, Inc.: USA, 1992.

3. RTCA/DO-254: Design Assurance Guidance for Airborne Electronic Hardware. FAA Advisory. Circulars, AC No: 20-152, June 30, 2005.

4. Švéda M. Experience with integration and certification of COTS based embedded system into advanced avionics system / M. Švéda, V. Opluštil // In 2007 Symposium on Industrial Embedded Systems Proceedings. Lisbon, Portugal: UNINOVA, Lisbon, Portugal, 2007. ISBN 1–4244–0840–7.

5. Certification of Aircraft Propulsion Systems Equipped with Electronic Control Systems, AMC 20-1 Effective: 26/12/2007. Annex II to ED Decision 2007/019/R of 19/12/2007, EASA.

6. Compliance Criteria for 14 CFR §33.28, Aircraft Engines, Electrical and Electronic Engine Control Systems. Advisory Circular 6/29/01, AC No: 33.28-1, FAA.

7. Electronic Engine Control Specifications and Standards. AIR4250, rev A, March 2004, SAE.

8. Automatic Code Generation Tools Development Assurance, Position Paper CAST-13, June 2002 Certification Authorities Software Team, FAA.

Поступила в редакцию 28.05.2009

Рецензент: д-р техн. наук, проф. С.В. Епифанов, Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Харьков, Украина.

СУЧАСНІ МЕТОДИ РОЗРОБКИ СИСТЕМ АВТОМАТИЧНОГО УПРАВЛІННЯ ДЛЯ АВІАЦІЙНИХ ДВИГУНІВ І ЇХНЯ СЕРТИФІКАЦІЯ

P. Axman, T. Kerlin, D. Svačina, V. Opluštil, J. Toman

Стаття описує необхідність використання автоматичних генераторів об'єктного коду для розробки систем критичного управління в авіакосмічній промисловості. Представлено базову модель V-циклу, її переваги й зображені методи розробки, які отримані з моделей MATLAB/Simulink [1] для реального цільового застосування. Увага також звернена до сертифікаційних вимог FAA/EASA ([5], [6], [7]) і конкретно процесу сертифікації нового авіаційного обладнання. Ці методи використовуються протягом повного циклу розробки підсистем авіаційного двигуна для малого цивільного літака (категорія FAR 23 / CS-23).

Ключові слова: літак, двигун, fadec, сертифікація, моделювання, V-цикл.

СОВРЕМЕННЫЕ МЕТОДЫ РАЗРАБОТКИ СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ ДЛЯ АВИАЦИОННЫХ ДВИГАТЕЛЕЙ И ИХ СЕРТИФИКАЦИЯ

P. Axman, T. Kerlin, D. Svačina, V. Opluštil, J. Toman

Статья описывает необходимость использования автоматических генераторов объектного кода для разработки систем критического управления в авиакосмической промышленности. Представлена базовая модель V-цикла, ее преимущества и изображены методы разработки, которые получены из моделей MATLAB/Simulink [1] для реального целевого применения. Внимание также обращено к сертификационным требованиям FAA/EASA ([5], [6], [7]) и конкретно процессу сертификации нового авиационного оборудования. Эти методы используются в течение полного цикла разработки подсистем авиационного двигателя для малого гражданского самолета (категория FAR 23 / CS-23).

Ключевые слова: самолет, двигатель, fadec, сертификация, моделирование, V-цикл.

Аксман Петр – инженер-программист отдела исследований и разработок мехатронических систем в компании «UNIS», Брно, Чехия, e-mail: paxman@unis.cz.

Керлин Томас – канд. техн. наук, инженер-конструктор отдела исследований и разработок мехатронических систем в компании «UNIS», Брно, Чехия, e-mail: tkerlin@unis.cz.

Свачина Давид – руководитель группы программистов отдела исследований и разработок мехатронических систем в компании «UNIS», Брно, Чехия, e-mail: dsvacina@unis.cz.

Оплуштил Владимир – д-р техн. наук, руководитель отдела исследований и разработок мехатронических систем в компании «UNIS», Брно, Чехия, e-mail: oplustil@unis.cz.

Томан Юрий – инженер-конструктор отдела исследований и разработок мехатронических систем в компании «UNIS», Брно, Чехия, e-mail: jtoman@unis.cz.