APPLICATION OF THE V-CYCLE DEVELOPMENT IN THE AEROSPACE INDUSTRY

Jiří Toman*, Tomáš Kerlín**, Vladislav Singule*

The aim of this article is to inform reader about the development steps which are used during the development of a critical control algorithm in aerospace industry. An article describes the motivation for use of automatic code generators for the development of critical control applications. A V-cycle model based design is introduced and its advantages and development practices that lead from design of a MATLAB/Simulink models to a real target application are depicted.

Keywords: *aerospace, aircraft, V-cycle, model based design, dSPACE, MIL, HIL, RCP, Matlab/Simulink, FADEC, CP-CS, FCEID, PCEID, modeling and simulation, ISA*

1. Introduction

Development procedures and practices in the aerospace industry have originated on those used in 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 'V-cycle'.

The V-cycle is a graphical construct used to communicate a model-based software development methodology. The advantages of Vůcycle 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. 1.

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

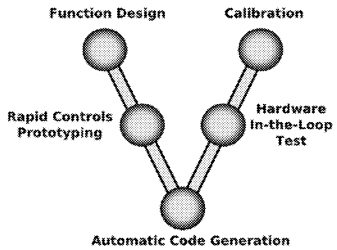

Fig.1: The V-cycle development process

* Ing. J. Toman, doc. Ing. V. Singule, CSc., Institute of Production Machines, Systems and Robotics, Faculty of Mechanical Engineering, Brno University of Technology
** Ing. T. Kerlín, Ph.D., UNIS, a.s.

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).

The 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 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 'real' component into the loop. This component might be the ECU or the real system-under-control. Current industry definition of the hardware-in-the-loop system is that a plant is simulated and the ECU is real. The model of the plant (and the simulation HW also) is the same like in stage of RCP.

Next step is a calibration, which is a process of optimizing or tuning real control algorithms to get 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 'calibration variables' in the ECU. Typical components of control algorithms which need calibration are look-up tables, gains and constants. The structure of control algorithm is not changed during the calibration process.

2. Implementation of the V-cycle

The typical V-cycle development process is based on a software development tools such as SW MATLAB & dSPACE system or NI LabVIEW. These tools provide a seamless transition from a block diagram to a real-time and target hardware. The following text shows application of the V-cycle development in aerospace industry demonstrated on SW MATLAB and dSPACE only. These SW & HW environments have been chosen based on the experience of the author of this article.

Particularly, for function design is mostly used MATLAB, Simulink, Stateflow and other MATLAB toolboxes. These tools together comprise a complex software package that forms 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 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 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. 2, its block diagram is shown in Fig. 3. The power of the jet turbine is control by 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 manual control system.

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

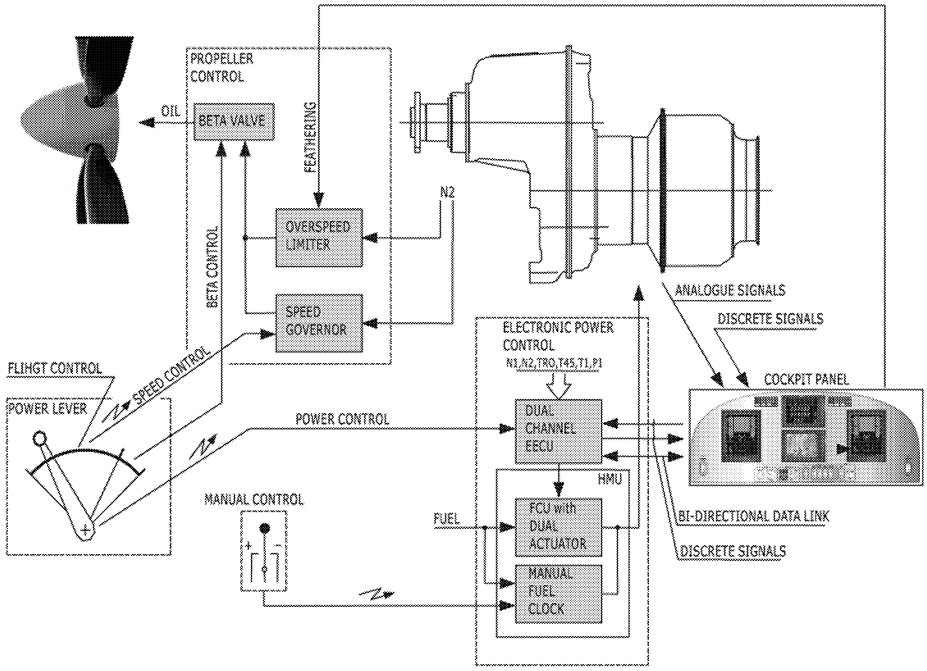

Fig.2: CP-CS system configuration for small aircraft

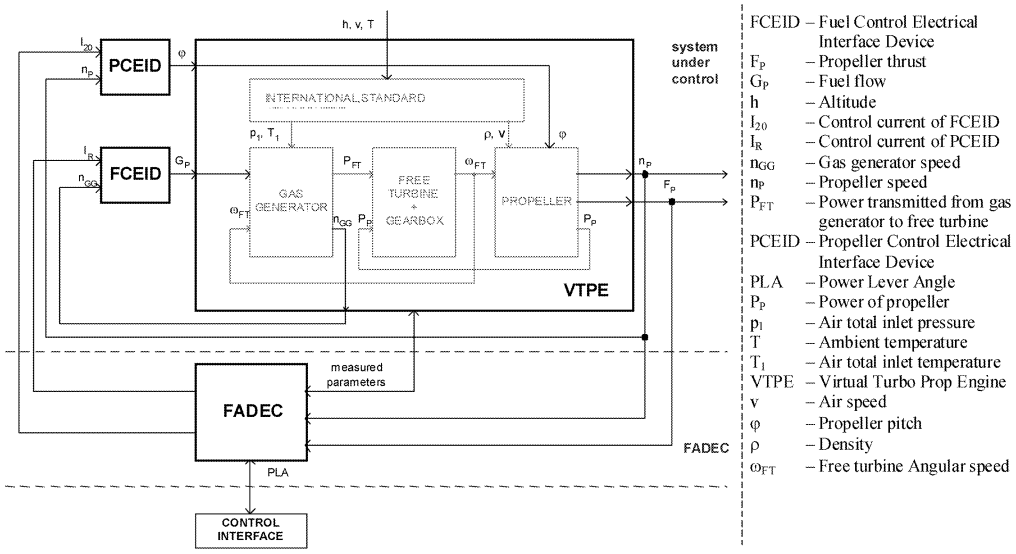

Fig.3: Block diagram of VTPE and CP-CS architecture

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 behaviour of real system. The 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 set of algebraic, differential equa-

tions, transfer functions or 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 core environment for mathematical computation, analysis, visualization, algorithm development, model-based design, etc. Schematic drawing implementation of the real hardware and its mathematical models to the Simulink is shown in Fig. 4.

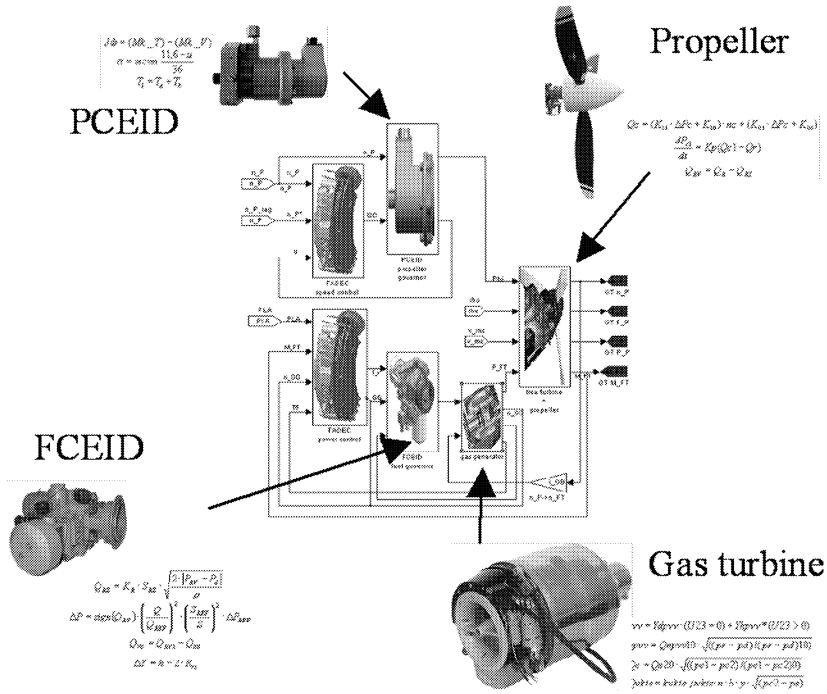

Fig.4: Mathematical model implementation to simulation software (Simulink)

The structure of the VTPE model is based on splitting of 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 PFT. Due to this fact the complexity of model is markedly decreased. The MATLAB/Simulink representation of the VTPE that is depicted in picture Fig. 3. is shown in Fig. 5. The VTPE model is precise enough to for examination of its behaviour during flight, for different aircraft speeds, heights, powers extractions and outside conditions. Start of the engine, reverse mode, taxiing and feathering are not possible to simulate.

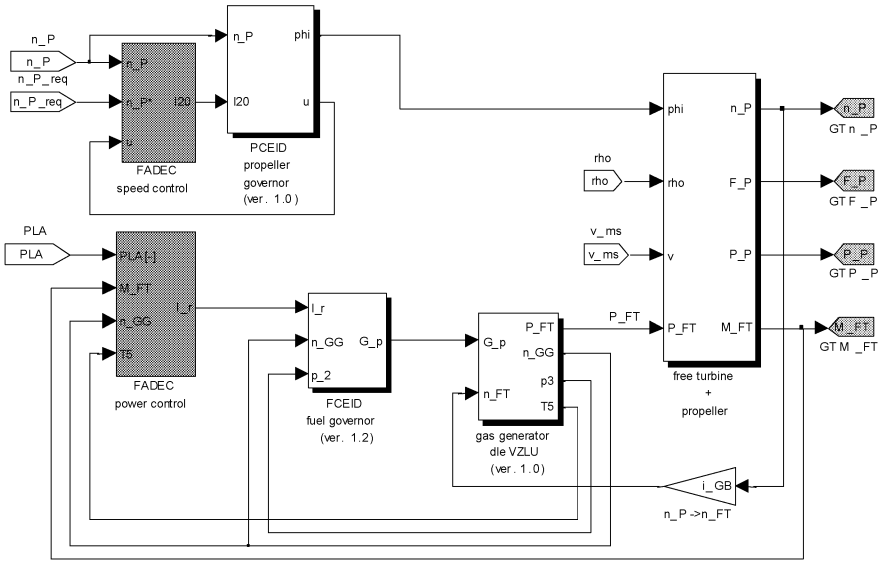

Fig.5: Simulink diagram of VTPE and CP-CS architecture

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) 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).

3.2.1. 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. 6.

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)

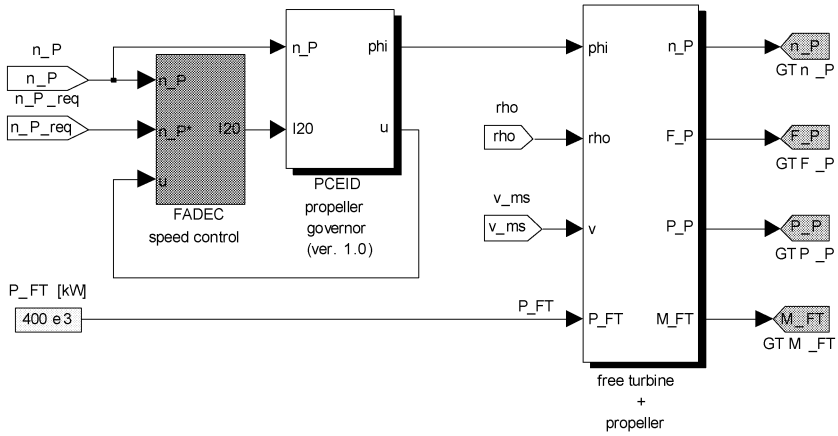

Fig.6: Simulink scheme of the propeller speed control

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.

3.2.2. Power control

Power control should ensure proper power of the engine, particularly proper power on the free-turbine shaft PFT, 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).

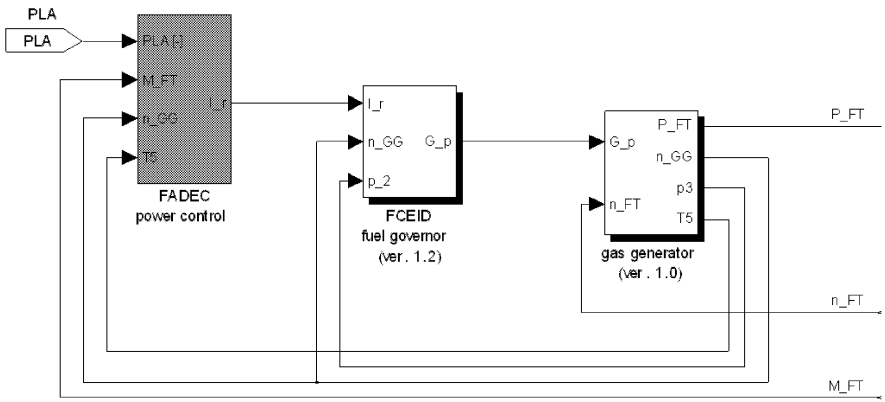

Fig.7: Simulink scheme of the power control

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.

Note: The model part that contains algebraic constraint blocks has to be replaced by numerical iterative calculation (showed in Fig. 9).

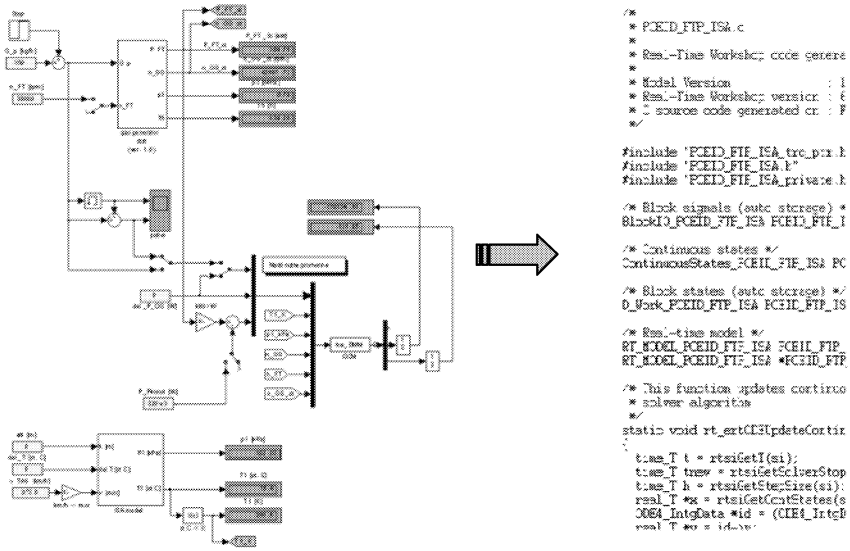

Fig.8: Automatic code generation

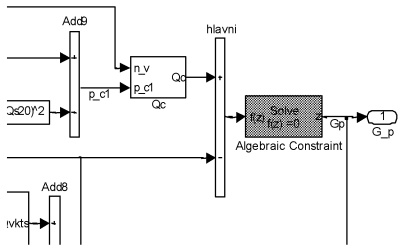

Fig.9: Example of model with algebraic constraint block

3.4. Software in the loop testing (SIL)

SIL test is proceeds by the MATLAB/Simulink tool. SIL tool is control systems simulator for temporal and functional simulations. The behaviour of the control system depends on

proposed architecture and on target hardware where the FADEC software is implemented. On this account 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 the model is implemented 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 number of expensive prototypes. The HIL test is performed by the dSPACE environment. The HIL testing offers to use following test combination (Fig. 10).

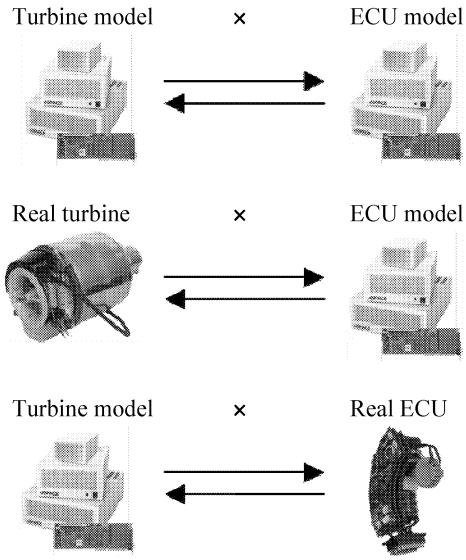

Fig.10: HIL test combination

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 to 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 (UNIS, Ivchenko-Progress, VZLU, Jihostroj and PBS) and project No.: MSM 0021630518 'Simulation modeling of mechatronic systems'.

References

[1] Kerlin T., Hubík V., Toman J.: Turboprop control electronic modeling, Technical Computing Prague 2009, pp. 54–62, ISBN 978-80-7080-733-0
[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., Opluštil, V. Experience with integration and certification of COTS based embedded system into advanced avionics system, 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

Received in editor's office: May 5, 2011
Approved for publishing: November 9, 2011