MODERN TURBOPROP CONTROL FOR THE TP400-D6 ENGINE POWERING THE A400M

H. Schirmer, H. Dostal
MTU Aero Engines GmbH
Dachauer Straße 665, 80995 Munich, Germany

1. ABSTRACT

As part of MTU Aero Engines' work share on the TP400-D6 engine project, a modern control concept was developed. It integrates propeller and engine control on a single full authority digital control system. Since it can rely on information on the entire powerplant, it allows for a high degree of coordination of these functions. As a result, the full potential of the engine may be exploited while ensuring easy handling and safe operation.

The complete development from concept to validated control system software is covered by MTU processes. These comprise simulation of the system behavior, automatic code generation, system and software integration, as well as validation on hardware-in-the-loop test rigs and engine test beds.

The approach taken permits very efficient development; problems are identified at an early stage, and can be solved and validated in an efficient and flexible manner due to time-saving automated processes. Model based design has enabled development to advanced maturity before the actual engine testing, providing a high level of confidence confirmed by a successful engine test campaign.

2. INTRODUCTION

2.1. Project Environment and Work Share

The TP400-D6 engine has been selected to power the future European military transport aircraft A400M. This modern turboprop engine is currently being developed by a consortium of four European engine manufacturers: ITP, MTU Aero Engines, Rolls-Royce and Snecma Moteurs. Ratier Figeac / Hamilton Sundstrand, a French-American company specialized in propeller technology, supplies the eight-blade propeller.

MTU's work share comprises the intermediate pressure compressor, shaft and turbine, as well as the shared responsibility for the control and monitoring system (CMS) in co-operation with Snecma. Within the CMS, MTU is in charge of the design and development of the engine control software, the complete engine protection and monitoring unit in hardware and software, as well as some accessories.

The turbopropeller control logic takes the key role within the multitude of functions performed by the application software. It is MTU's responsibility to provide the control concept and the logic architecture, to integrate and implement sub-functions supplied by partner companies, to perform control design and to ensure the logic as a whole controls the powerplant safely and effectively.

Part of the turbopropeller control algorithms is supplied by the propeller manufacturer: all logic directly actuating the propeller control module. Considerable effort has been put into harmonizing the control concepts and synchronizing the activities of engine and propeller control.

2.2. Plant Description

The TP400-D6 is a three-spool turboprop engine comprising a two-spool gas generator and a power turbine driving an eight-bladed propeller via the low pressure shaft and a power gearbox

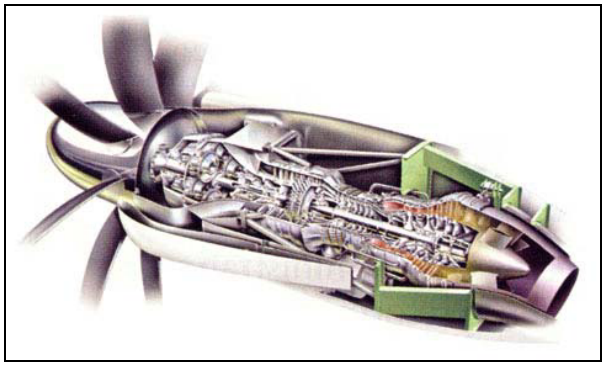

Figure 1 TP400-D6

Four engines, each rated at approximately 11,000 shp, will power the A400M at altitudes up to 40,000 ft. The propeller's high speed design will allow for a maximum flight Mach number of 0.72. Different propeller speed set points will permit adaptation to satisfy noise, performance and efficiency requirements varying with flight phase, or mission.

2.2.1. Main Control Parameters

As for any other type of aircraft engine, the quantity of most interest with respect to aircraft performance is thrust. Yet, it is difficult to measure on an installed engine, and therefore not used for control.

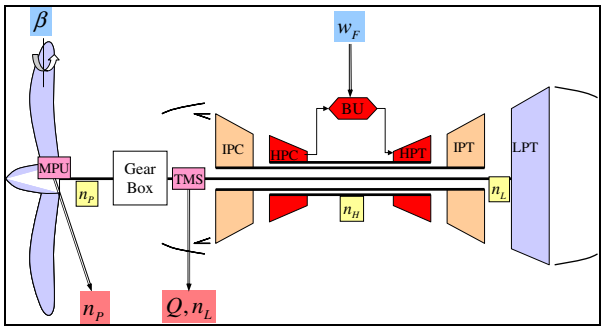

Figure 2 Schematic of the plant with main measured and control variables

Two parameters predominantly characterize the operating point of a turboprop at steady state in terms of thrust because, together, they determine for the largest part the efficiency of the propeller:

– Power delivered to the propeller and
– propeller speed NP, at which this power is consumed.

In addition, propeller speed is of interest for many mechanical aspects.
Hence, controlling these two quantities allows operating the engine at a defined point with regard to both thrust and mechanical integrity.
Besides the typical actively controlled engine limits, such as mechanical and aerodynamic spool speeds or acceleration and deceleration limits, a maximum torque limit must be maintained.

2.2.2. Main Control Variables

The two main degrees of freedom of the turboprop engine are
– fuel flow and
– propeller pitch angle.

More or less influential on the engine operating point, but subordinate to above controls are
– variable stator vane angle and
– handling air bleeds.

The following variables are not actual control variables but rather generate disturbances:
– air bleeds for various purposes (customer, anti-icing etc.)
– power off-takes

The latter will not be considered further in this paper.

2.2.3. Dynamic Behavior and Couplings

Like on the intermediate and high pressure spools, the low pressure rotor system consists of two turbo machines, one generating and the other consuming it. However, on the low pressure system, the operating points of the two components can be influenced independently through the two major degrees of freedom.
The propeller blade angle β has direct, undelayed influence on propeller power. A change in fuel flow translates into a change of low pressure turbine (LPT) power part instantly, part delayed (lead-lag behavior). Neglecting power off-takes and parasitic losses, the difference between the two power levels determines the rotor acceleration at a given inertia.
Whereas the impact of NP on the LPT power is weak, the propeller power depends strongly on propeller speed. An equilibrium will be reached when the propeller speed at a given pitch angle results in a propeller power consumption matching the power provided by the LPT. In other words and simplifying: fuel flow determines power, and pitch angle defines the spool speed at which this power is consumed.

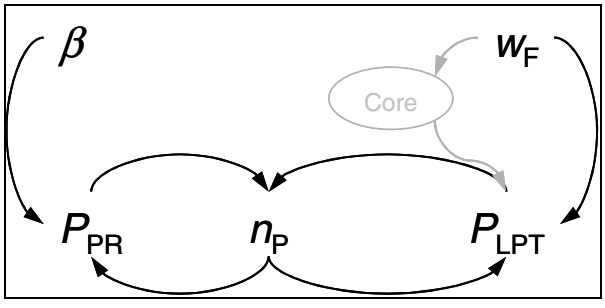

Figure 3 Couplings between main parameters

It becomes therefore obvious that fuel flow is the appropriate variable to control engine power, and that propeller speed can be controlled most effectively via propeller blade angle. Yet, it is also apparent that there is a more or less strong coupling between the controls, pitch angle and fuel flow, and the resulting power and rotational speed. These dependencies are illustrated in Figure 3.
Either of the main control variables primarily impacts its associated power. Since that power change incurs a power unbalance, it is followed by a change of rotational speed. This, in turn, has an indirect effect on the respective other power term. If, for instance, the fuel flow is increased while the propeller pitch remains fixed, the propeller speed will increase to compensate the additional power delivered by the LPT. Conversely, increasing the blade angle immediately raises propeller power, followed by a decrease in propeller speed, with a rather mild influence on turbine power.
Effectively, neither of the mentioned power quantities can be measured dynamically and therefore none of them play a practical role as controlled parameter (see Section 3.1). Figure 2 shows the location of the torque measurement system (TMS). Its measurement is used to compute the engine shaft power.
The mechanical model in Figure 4, together with Equation (1), illustrates how the measured torque – and thereby measured power – is a function of both propeller and LPT torque. The inertias of propeller plus gearbox on one end, and LPT on the other, are approximately equal when taking into account the gear ratio. Thus, roughly half of the immediate propeller torque impact caused by a blade angle change is sensed in the measured torque. This implies a strong coupling between the control variable β and the measured power.

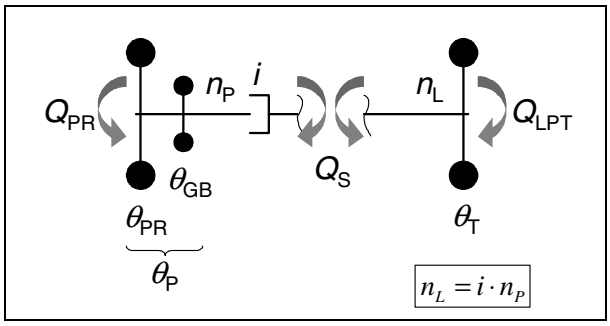

Figure 4 Simplified mechanical model of the LP rotor system

Summarizing, the main cross couplings are
1) the influence of fuel flow on propeller speed and
2) the impact of propeller pitch on measured power.

2.3. Control System Architecture

The TP400-D6 is equipped with a full authority digital engine control (FADEC) system consisting of two electronic devices, an Engine Control Unit (ECU) and an Engine Protection and Monitoring Unit (EPMU).
All CMS tasks can be grouped into control, protection and monitoring functions. As far as control and protection are concerned, the entire system is fully duplex and multi-fault tolerant. Single faults are accommodated without impact on the system performance.
The primary control functions are performed by the ECU. It realizes all control algorithms digitally. Both the propeller and gas generator control are fully integrated within the ECU application software. This allows for a high degree of integration and interaction characterizing a modern turbopropeller control system.
Protection and monitoring are performed independent of the control function by a separate electronic device, the EPMU. It is also a fully digital system providing backup protection of

the powerplant's spool speed limits as well as prevention of erroneous operating conditions which cause a hazard to the aircraft. In particular, the EPMU prevents propeller blade angles incurring excessive drag.

3. CONTROL CONCEPT

3.1. Control Problem and Solution

The turboprop engine has two major degrees of freedom, and two properties shall be controlled. Moreover, the variables and controlled parameters exhibit significant cross coupling, which poses a multivariable problem.
The approach taken here is to implement two single-input single-output controllers, in conjunction with appropriate decoupling measures to minimize their influence on each other. As concluded in Section 2.2.3, propeller speed is controlled through propeller blade angle, and fuel flow is utilized to control power.
Ideally, one would want to control propeller power; it correlates – independent of dynamic effects – directly with thrust. One of the drawbacks is certainly its strong coupling with propeller pitch. The other is that it practically cannot be measured as it denotes the power exerted by the working fluid on the blades.
The input shaft power may be measured, and it is equal to the propeller power for constant propeller speed. In fact, the power measured at steady rotational speed differs from the propeller power only by the power offtakes and losses, which can be accounted for in the rating. For steady state conditions the measured power is thus the appropriate control parameter, but decoupling mechanisms are applied to address the dynamic issues.

3.2. Decoupling

The goals for the decoupling mechanisms are

1) to reduce the disturbance of the fuel control loop by pitch movements,
2) to moderate the effect of propeller acceleration on propeller power and
3) to diminish the impact of gas generator acceleration on propeller speed

3.2.1. Dynamic torque compensation

Technically, there is no direct impact from propeller pitch on LPT power, but rather only an indirect, weak influence via changes in NP. Therefore, LPT Power would be much more suitable as a controlled parameter from a decoupling point of view. However, for the same reasons as stated in the previous section for the propeller power, it cannot be measured directly.
To compensate the pitch influence, a virtual LPT power is computed and used for control (Equation (2)). At steady state, this virtual power is equal to the measured. During speed variations, the acceleration term for the LPT is added. This is independent of the cause for the acceleration, i.e. it works also for accelerations brought about by pitch changes. For instance, a pitch increase will augment the measured power, but at the same time, the resulting acceleration will compensate for this in the virtual power.

3.2.2. Propeller Acceleration Power Feedforward

A defined rate of change in propeller speed corresponds to a certain power required for this acceleration. This power term is the difference between propeller and turbine power. If turbine power remains constant, the acceleration can only be effected by lowering propeller power. Yet, this means an undesired reduction in thrust.
During propeller speed transitions, the acceleration power is therefore added to the power demand, and the propeller speed demand change is delayed to reflect the finite response time of the gas generator to the power demand. In this manner, the entire acceleration power is provided by the engine and, ideally, propeller power is not affected by the transition.

3.2.3. Torque Anticipation

A change in turbine power causes NP to vary as well. Without any interaction between engine and propeller control, the propeller speed must exhibit a control deviation before the controller can begin to counteract the development by moving blade angle.
The resulting speed excursion can be moderated by presteering a pitch change corresponding to the power change. For this purpose, the engine power is anticipated from the commanded fuel flow by a simple engine model. Its rate of change is subsequently translated into a pitch rate, which is fed forward into the propeller speed governor.

3.3. Turbopropeller Control Modes

The control mode described so far is commonly referred to as *Speed Mode*. It works as long as the propeller blade angle has significant impact on propeller power and is therefore an effective means to control propeller speed.

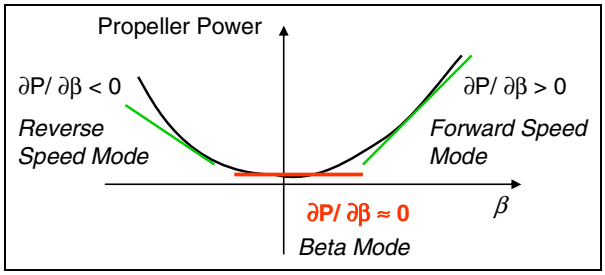

Figure 5 Propeller power as a function of blade angle (qualitative)

There exist propeller regimes where this is not the case. Figure 5 depicts how the gradient $\partial P/\partial\beta$ virtually vanishes in a certain region of the blade pitch range. This region is relevant at moderate air speed. At higher forward speed, the propeller exhibits this characteristic at negative power consumption only, which is a regime to be avoided anyway.
In this area the so called *Beta Mode* is applied. Propeller pitch is scheduled open loop as a function of throttle lever position, while speed control is performed by the fuel control. This mode is used on ground at low power settings.
In the reverse thrust regime, traditionally, Beta Mode was applied. However, this control mode makes it difficult to align propeller blade movement and engine power development; pitch can be actuated fast whereas the engine has restrictions in terms of acceleration. On the other hand, the propeller characteristic features the aforementioned gradient required for Speed Mode control. Consequently, the TP400-D6 is controlled in Speed Mode in the reverse thrust regime.

3.4. Control Algorithms Architecture

Figure 6 shows how the sub-functions of the turbopropeller control are structured. The signal flow shows only a few selected parameters – in fact, much more data are actually exchanged.

3.4.1. Integrated Power Management

The power management function computes a power rating along with the matching propeller target speed. It performs integrated power management, which means that the only input required from the pilot for default operation is the throttle lever position. This determines the level of power, or thrust, demanded by the pilot, and the function automatically selects the appropriate combination of rating and propeller speed. For special flight phases, the pilot has the possibility to override these default settings.

3.4.2. Engine and Propeller Control Coordination

The engine and propeller control coordination is the heart of the TP400-D6 turbopropeller control, and therefore a separate section (see Section 3.5) is dedicated to this logic. In essence, it manages the demands for both, fuel and propeller control, as well as the orientation of the propeller controller. Depending on the propeller speed setpoint changes and control modes, it biases the power demand for decoupling as described in Section 3.2.2.

3.4.3. Propeller Control

There are three main tasks performed by the propeller control: Pitch control, propeller speed control and synchrophasing. The pitch controller is used to attain a scheduled pitch angle, and to maintain certain blade angle limits such as the minimum in-flight pitch (primary low pitch stop).
The speed governor controls the propeller speed through pitch angle. In addition, it is utilized to achieve defined attitudes of the propeller blades with respect to the other propellers. For reverse operation, the speed governor uses gains of inversed sign to account for the changed propeller characteristic.

3.4.4. Fuel Control

To accommodate both Beta and Speed Mode, the fuel control accepts two demand inputs: power and engine LP speed. The respective control loops compete in a lowest-wins logic. Thereby, the controller, whose command should normally not be selected in a certain control mode, provides a backup limitation for its controlled variable.
As on other turbine engines, the fuel control function actively protects engine limits through the appropriate control loops.

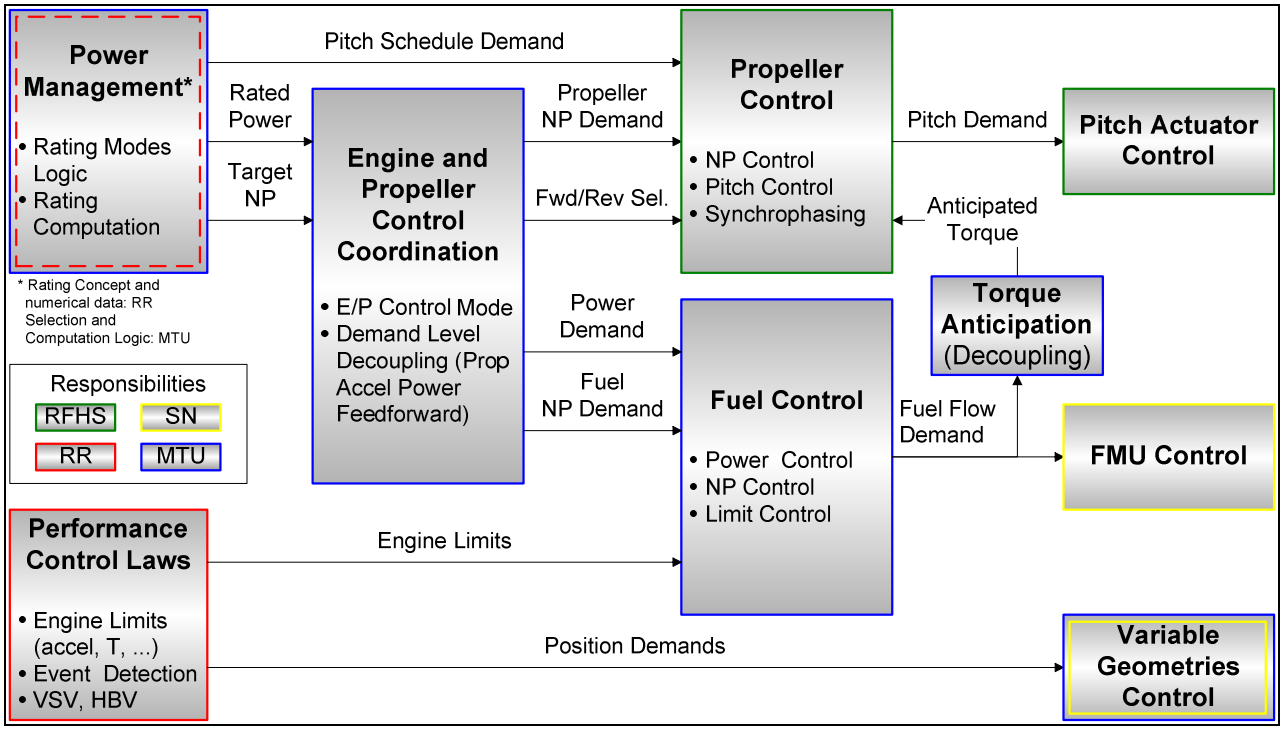

Figure 6 Control algorithms architecture

3.5. Engine and Propeller Coordination and Control Mode Transitions

During all phases of normal operation, the control loops in both propeller and engine control remain active. There is no switching between controllers, or deactivating them. The different control modes and transitions between them are achieved entirely by coordinating the demands of the four main control loops: power controller, engine LP speed governor, propeller speed and pitch governors.
With increasing air speed, a growing propeller blade angle range is associated with negative power consumption, the so called windmilling regime, depicted in Figure 7. Continued operation in this regime can lead to propeller overspeed within a short period of time. As the engine cannot produce less than approximately zero power, the resulting positive power balance leads to acceleration of the rotor.
In Beta Mode, the propeller speed governor may therefore control pitch to prevent NP from exceeding a limit in case the scheduled blade angle is in the windmilling regime. This protective logic allows the pilot to make use of the full powerplant capability while not having to worry about exceeding a limit.

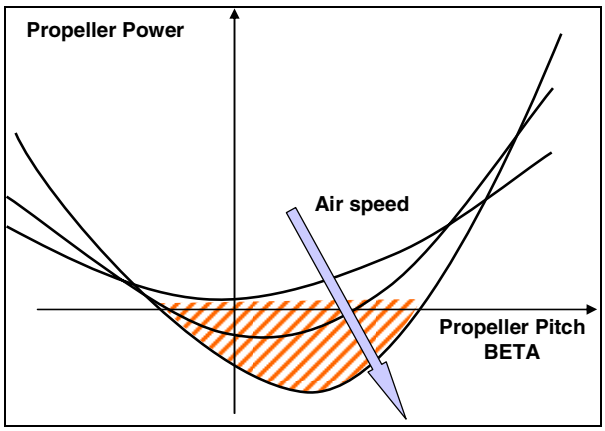

Figure 7 Windmilling regime

For transitions between the three basic modes, the coordination logic temporarily schedules the demands to drive pitch quickly through the critical areas in order to keep speed excursions small. It also ensures that this procedure is not performed while the engine is producing power, not to aggravate the acceleration.
Due to the integrated approach, the logic has knowledge about the operating states of both engine and propeller. It can thus manage the transitions efficiently and in a smooth and safe way.

4. DEVELOPMENT PROCESS

The key to success in modern embedded software projects, which are typically characterized by a highly demanding time schedule combined with high system complexity, is nowadays believed to be a completely model based design approach.
At MTU Aero Engines a fully integrated model based design process including automatic generation of the control system application software was accomplished for the TP400-D6 control and protection system.
There exists already a multitude of papers pointing out the various advantages of model based design in general. The aim of the following section is to point out the specific possibilities utilized in the TP400-D6 project at MTU.

4.1. Model Based Design in a Project-Wide Context

The A400M project as a whole follows highly integrated model based design processes, with the TP400-D6 engine being part of this progressive approach. From the very beginning of the system design phase models have been exchanged between the customer and the supplying consortium.
Under the authority of Rolls-Royce Germany a simulation model of the entire set of A400M powerplants, composed of the four turbopropeller engines, is delivered to the customer in short time intervals. It reflects the current status of system definition.
Airbus integrates this powerplant simulation software in various simulation platforms including dynamic simulations of the complete aircraft and hardware-in-the-loop (HIL) test systems. Therefore, the model interfaces with simulated and real aircraft systems; it is utilized for both system validation and the continuous development and refinement of customer requirements.
Within the powerplant simulation model, one single engine system is designed as a re-usable module. Each module consists of engine, propeller, actuator and sensor models, as well as the FADEC module. Together with some accessories' models, this module is under MTU responsibility, and represents the most complex part of the engine model.
A key feature of the TP400-D6 development is the fact that all FADEC simulation models provided to the customer are directly derived from the ECU and EPMU models, which are also used to automatically generate the respective application software code. The models delivered to the aircraft manufacturer are essentially identical to the autocoding sources, except for some minor adaptations made to realize custom simulation features requested by Airbus.
The shared models are the most effective component of technical information exchange within the project and could justifiably be called a main project language, used to describe the system definition. They enable all involved parties, including the customer, to perform validation and verification to a considerable degree already in early project phases, resulting in advanced system maturity well ahead of hardware components being available.

4.2. Simulation Environment

In the TP400-D6 project it was achieved to install one closed loop powerplant simulation model shared by all involved parties as the single source for all applications that use simulation of the entire system model, or parts of it:

- Control design, development and pre-validation of control logic in an offline closed loop simulation environment
- Automatic generation of application software for both control and protection units (ECU, EPMU) directly from the system model
- Real time powerplant (i.e. engine, propeller and accessories as required) simulations on various HIL test rigs, directly derived from the engine system model via autocoding processes employing The Math Works' Real Time Workshop (RTW)
- Simulation of the powerplant behavior as part of the aircraft system simulation at customer level

A common top level structure of the system model which interfaces with the testing environments of all kinds of simulation platforms has been agreed between the involved parties. As a result, test vectors and simulation results can be exchanged in a very efficient way.
A high level of modularity of the system model provides the flexibility necessary to make it the single source model also satisfying the needs of the employed autocoding processes. An integrated model structure fulfilling the various requirements has been achieved in cooperation with Rolls-Royce Germany engineers. Especially the common usage of interface models in the multiple simulation configurations was a complex task difficult to accomplish.
All simulation tools required for the MTU software development environment are part of The Mathworks' product portfolio. Beginning with MATLAB Release 14, these tools have become powerful enough to simulate and autocode large systems like the TP400-D6 FADEC utilizing a single model for all purposes.

4.3. Control System Software Design and Offline Simulation

In coherence with the A400M project approach, development and design of the TP400-D6 control and protection application software at MTU Aero Engines is performed for the most part within the closed loop simulation of a single-engine system. The model is a one-engine version of the four-

engine powerplant simulation model delivered to the customer, and updated according to the status of system definition.
Both dual-lane systems, ECU and EPMU, are simulated as single lane units with the possibility to simulate inter-unit and inter-channel communication aspects as well. The communication is modeled at a level of detail required to enable the developers of signal validation and selection logic to both implement and validate their functions. Consequently, all logic development and control design tasks can be performed within the closed loop simulation model – all developers take responsibility for the proper integration of their functionality into the system.
The tradeoff between simulation performance and representativeness does not warrant complete simulation of both lanes of the duplex system. Also, it does not yield a real benefit; in principle, the FADEC software is designed to accommodate a single fault (including electrical failure of a complete lane) without effect on the system performance. Certain hardware effects may temporarily have a small impact, e.g. due to hardware switching. However, these cannot be modeled accurately with reasonable effort, and it is therefore much more efficient to rely on the HIL test environment.
Any modification to the logic, whether identified and developed in the offline simulation, at a HIL test rig or an engine test bed, is realized in the offline simulation environment, as the FADEC model is the basis for the application software codes.
While the core of the application software is produced automatically, the software parts interfacing the operating system and hardware are coded manually.
Since this manual code represents the external interface for the autocoded software, it is simulated in appropriate detail in dedicated interface models. Furthermore, these models reflect significant characteristics of the controller hardware and/or operating system such as delays and filters on sensor signals.

4.4. Autocoding

4.4.1. Autocoding Software Development Environment

For the TP400-D6 project, an in-house autocoding software development environment has been established at MTU. It covers all steps from system requirements to software, including software requirements specification and design. The final product, certifiable software, is generated automatically from a system model interfacing an open or closed loop simulation environment.
After thorough evaluation of the integrated simulation and autocoding concepts, the concept proposed by The Mathworks had yielded the best results, and thus the RTW Embedded Coder was selected as the core of the autocoding software development environment at MTU.
Using simulation and autocoding tools from a single suite offers the significant advantage of easy maintainability and future upgrades. Also, by using the Embedded Coder, a much better integration level of software and simulation needs has been achieved in comparison to the formerly used tool set.
The target to keep the installed process straightforward has been met. Three components are in the developer's toolbox to design the model:

- a library of basic functions,
- a data base and
- model design guidelines.

In the simulation model, basic functions are represented by imported legacy code. These code functions are qualified prior to qualification of the integrated autocode.
Model design guidelines provide rules on how the basic functions are to be applied. All data for automatically as well as manually generated code is maintained in a single data base.
While ensuring successful application of the process, the modeling guidelines are kept sparse enough not to unnecessarily restrain the designer's work.
After completion of the modeling phase, the automatic code generation routine is executed. The resulting code files need only marginal post-processing, which is automated as well.
The integrated build process including compilation, linking and final post-processing of the compiled program is automated to a high degree.
For changes that affect only the modeled functionality, the software is running on the target and ready for testing approximately one hour after finishing the autocoding model.

4.4.2. Autocoding Model Design Guidelines

A modern development environment must address permanently increasing system complexity and ongoing system requirement evolution during the development phase of a project. The time schedules in the TP400-D6 project presume such flexibility requirements can be met by application of high efficiency development processes; the advantages of model based design were utilized to cope with these expectations.
Specialists in simulation based control software design and those experienced in the classical approach, based on documents and manual code, tend to have contradicting philosophies. Considerable effort has been put in to exploit the benefits of progressive simulation techniques while preserving the standards traditionally applied to produce safety-critical software.
As a result, a limited but very effective set of modeling rules has been developed which can be easily followed by system/software developers. It addresses both the simulation based system development and the coding-related aspects, and guarantees RTCA/DO 178B certifiable autocode as the outcome of the integrated development process.
A good example to point out the difficulties one is faced with when integrating the different development approaches is the calculation order of controller algorithms.
The numerous model based design iterations are associated with modifications which affect the calculation order within the modeled logic
For a developer familiar with this simulation model characteristic it is self-evident that the calculation order should be determined automatically by the simulation tool. The classical software engineer, however, used to document based software design, requires first of all the possibility to impose a calculation order on the simulation model. He tends to doubt the advantage of automatic determination of the optimal calculation order by a tool which has the authority to restructure the generated software.
More generally speaking, one of the main challenges in setting up a model based software development environment was to bring together a highly iterative bottom-up approach, involving a modern simulation environment, and the proven top-down approach of classical embedded S/W projects. The latter relied on the fact that the first system tests are carried out with considerable delay to the software design "on paper".
Regarding the software computation order, initialization and timing behavior, an integrated solution was developed which combines the advantages offered by the simulation environment while addressing the clear need to be able to impose a certain calculation order on the autocoded software.

This is a prerequisite for incremental development and testing. Tool-supported control over this aspect within the system model, including the possibility to simulate the expected software behavior to the associated level of detail, minimizes the risk of surprises during system and software integration.
Recapitulating, it was a challenging but worthwhile project to define rules reflecting a common understanding, in general and detail, of how a simulation model should be set up, structured and configured in a change management system, when used for both: system design in a closed loop simulation environment and automatic generation of control system application software.

4.5. Hardware-in-the-Loop Testing

Prior to each control and protection software release, the FADEC equipment is tested in HIL simulations before operating the control system on a real engine. To ensure the independence of system development and validation, HIL testing is performed by a separate department within MTU.
All rigs are part of a highly automated test environment. They can be operated from each authorized desktop PC within the company network. Both open loop and closed loop system testing is performed at MTU facilities with the ECU and the EPMU connected to a test rig in application hardware configuration.

Figure 8 HIL Test Rig

4.5.1. Open Loop Testing

Software acceptance testing (SAT) verifies the application software against the software requirements on system level. For this purpose, the application software is stimulated with a monitoring and setup unit (DMSU) accessing the program directly at the application software interface. The DMSU is an in-house development, and interfaces the automated test environment as well.
The signal paths between application software output and electronic engine model input, i.e. including control unit operating system and hardware as well as test rig hardware, are validated in a similar manner. To ensure that only control unit faults are found during these tests, the test rig interfaces are validated separately before testing. The same principle is applied to the chain between engine model output and application software input.
Open loop testing already provides a good level of confidence in the respective unit's application software, as well as in the target platform. The next step is the system integration validation in closed loop HIL simulations.

4.5.2. Closed Loop Testing

System integration tests are performed for verification of the control system against the system requirements.
In this configuration, the test system performs a real-time simulation of the turbopropeller engine, which is directly derived from the offline-system simulation model by automatic code generation using RTW. For simulation with the real control units in the loop, the engine simulation has to run at a higher sampling rate than in the offline environment, due to the asynchronous sampling at the interfaces of test system and control units.
To relax the computation performance requirements for the real-time platforms, a multirate simulation of the engine has been set up. In particular, the actuators – characterized by fast dynamics – are simulated at a higher refresh rate than engine subsystems featuring a less dynamic time response.
There are two hardware configurations for closed loop HIL testing at MTU:

a) Dry Rigs
b) Wet Rig

On the former type a), the engine simulation comprises all engine subsystems, whereas on the so-called "wet rig" the real fuel system hardware replaces the models. The fuel system actuators are the fuel metering valve and the variable stator vane actuator.

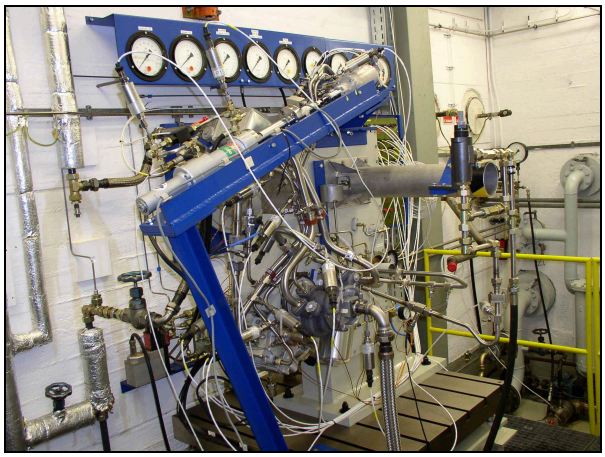

Figure 9 Fuel System Installation at MTU Wet Rig

After passing the closed loop rig testing, the control units have proven to operate properly under almost real conditions. The uncertainty remaining is mainly reduced to the characteristics of the simulated components differing from the actual equipment. Generally, the dominant component is the model of the engine. In the particular case of the TP400, this applies to the propeller including its actuation system. Nowadays modeling techniques are advanced enough to produce a high-performance, accurate model for turbine engines. Consequently, the confidence in the product validated in a HIL test rig environment is very high.

4.6. Process Overview

Figure 10 illustrates the process in which the described development environment is employed. The left column depicts the various simulation platforms, and the steps from one to another with the portion of real hardware increasing.
The right column summarizes the steps in a process-oriented chart, and shows the various levels at which results flow back into the development process to allow iterations in the design. Any modification step in response to a result is realized as

early as possible in the process, but at the adequate level of simulation representativeness.
For instance, the overall system behavior is largely developed to satisfy the customer expectation already at the offline simulation level. The modeling depth is accurate enough for this purpose – which, of course, does not exclude the possibility of optimization during flight tests much later.
Where detail hardware effects play a role, the problem may become visible on a HIL rig only. In this case, the remedy must be developed, then implemented into the software model and coded. Within a short period of time, the modified software is available on the target platform to verify that the problem has been eliminated.
In this manner, quick iteration cycles are achieved and the real engine test bed is equipped with a very mature product. Moreover, this saves development time in general, and in particular test rig time and costly engine test running hours. As a consequence, the number of required test rigs and test beds is also kept low.

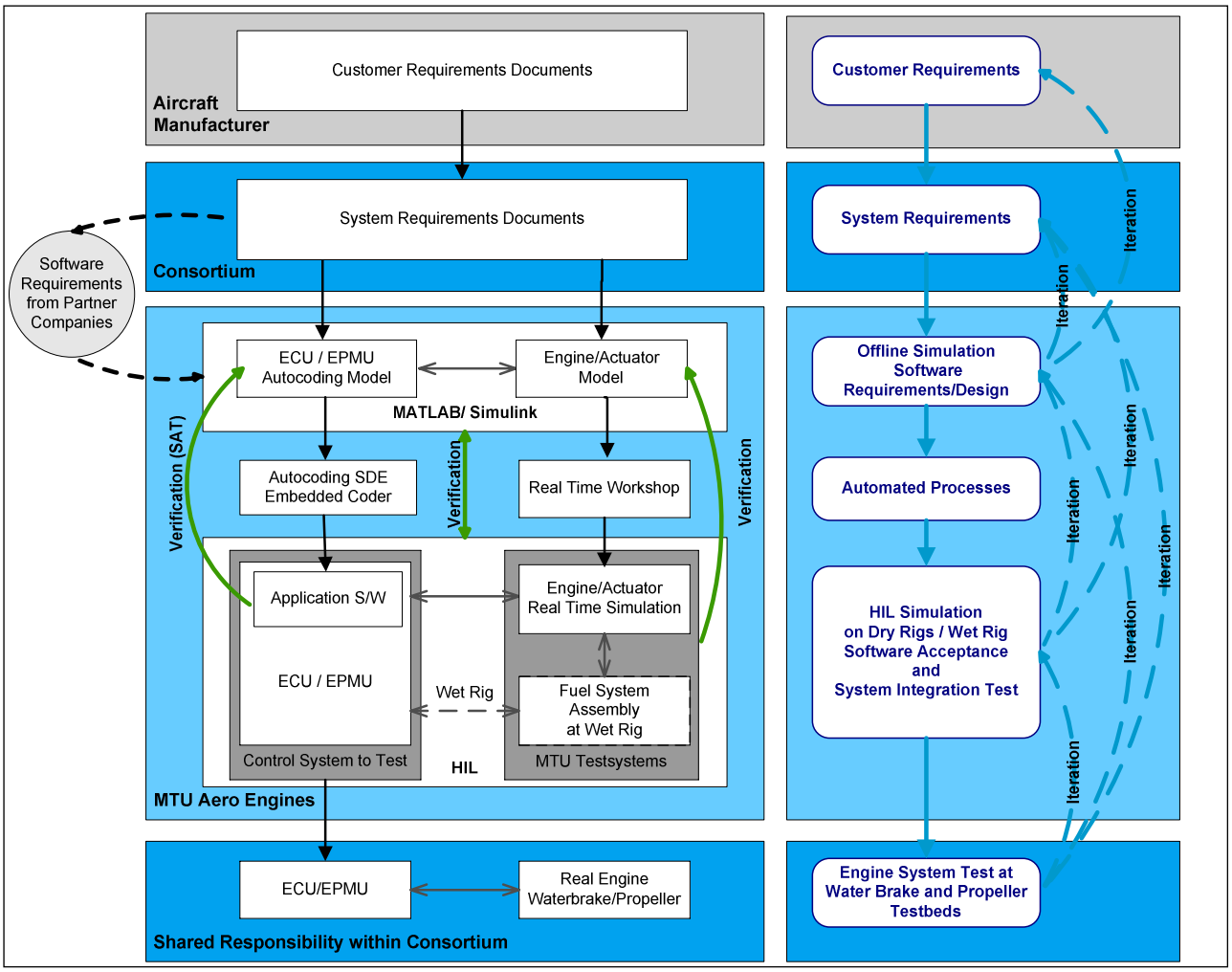

Figure 10 Development Process Utilizing Simulation Platforms and Automated Procedures

4.7. Test Automation and Verification Aspects Concerning the Model Based Approach

By consistently following a single source model approach, in which all information pertinent to a software function is managed through the respective part of the system model, the advantages of model based design can be transferred into the formal validation and verification (V&V) processes granting significant improvement in efficiency.
With all necessary information being gathered within the system model, the model defines the requirements and the design for a large proportion of the application software.
For the functionality fully under MTU's responsibility, the software requirements specification (SRS) and the software design document (SDD) are supplied with information automatically generated from the system model. Consequently, all software V&V activities which have to be carried out against the SRS and the SDD can be carried out against the system model.
The advantage is obvious: the possibility to actually simulate the software requirements and design from system level down to unit level may legally be utilized for all V&V activities including certification.
First step in the process is to validate the software requirements against the system requirements by employing the software model in a closed loop simulation.
Then, for testing the code on the target platform against the SRS (Software Acceptance Test, SAT), the system model of the respective control unit's application software is used to support the definition of test cases and generation of reference results. The SAT results, generated by stimulating the actual software with these test vectors, are automatically

compared to the reference values. This approach is valid from a certification point of view, because the system model is defining the software requirements.
The concept of the model being identical to software requirements and design aids the software verification tasks further by automatically providing traceability and avoiding errors in translating requirements into code. Automated document generation reduces the required effort and source of errors.

As the project is now approaching the certification phase, the priorities in optimizing the highly integrated, model based software development shift from maximizing the development efficiency towards exploitation of test automation possibilities with respect to certification aspects. For instance, the possibility of a model based, and potentially automatic, generation of test vectors for unit testing is currently under assessment.

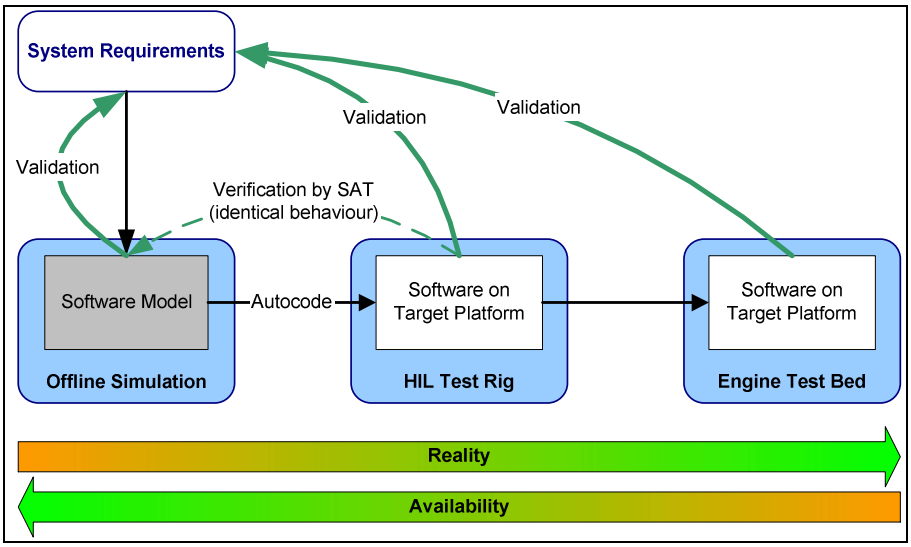

Figure 11 Test Environments for V&V Activities

4.8. Engine and Propeller Test Beds Within the Consortium

Before testing the engine on an aircraft, tests are performed on three types of test beds:

1) Engine test beds with water brake
   a) Static conditions at approximately sea level
   b) Altitude test facility
2) Powerplant test bed with engine and propeller, at approximately sea level, static conditions

On the former group, the engine is tested with a water brake replacing the propeller. On the one hand this takes away the possibility to validate the behavior of the powerplant as a whole, especially with respect to interactions of engine and propeller. On the other hand, it offers the possibility to test indoors. Moreover, it is feasible to test in a facility simulating conditions throughout the flight envelope, which is not an option with a huge propeller.
This restraint strengthens the role of simulation, because the first test with the complete powerplant at high Mach number and altitude cannot be performed until it is installed and flying on an airplane.
Testing with propeller requires an outdoor test bed, which constrains validation to ground level static conditions but is a fundamental and indispensable step in engine and propeller integration. Especially from a controls point of view this phase is important. It is the first opportunity during development to verify control system performance on the real system.

5. RESULTS

5.1. Engine and Propeller Tests Accomplished

So far, engines have been tested on water brake test beds at both sea level and altitude facilities, as well as on an outdoor powerplant bed with propeller.
The CMS-related tests on all of the facilities showed that the turbopropeller control function fulfills the expectations in all significant aspects. Engine limits were maintained and control of all parameters was found to be stable. All control modes, engine and propeller coordination and mode transitions exhibited satisfactory behavior.
Up to date, only potential for detail optimization in the control software has been identified. This is certainly owed to the iterative development process; with its several stages of validation, based on simulation with a growing part of real hardware, it allows to identify most problems even before the first engine is turning.

5.2. Future Milestones

Since the powerplant including the propeller has been tested only at static ground conditions so far, the next major challenge from a controls point of view will be the Flying Test Bed. It will be the first opportunity to assess the behavior of the real system at high airspeed and altitude. This test bed will consist of a C130 turboprop-powered aircraft where one of the original engines is going to be replaced with a TP400-D6 engine.
Testing of the full functionality throughout the A400M flight envelope will be possible during the new aircraft's flight test campaign.

6. EQUATIONS

(1) $$Q_S = \frac{\Theta_P}{\Theta_P + \Theta_T \cdot i^2} Q_{LPT} + \frac{\Theta_T \cdot i}{\Theta_P + \Theta_T \cdot i^2} \cdot Q_{PR}$$

(2) $$P_{LPT} = (Q_S + \Theta_{LPT} \dot{\omega}_{LPT}) \omega_{LPT}$$