Flight Simulator Model Integration for Supporting Pilot-in-the-Loop Testing in Model-Based Rotorcraft Design

Umut Durak[1], Torsten Gerlach[2]
German Aerospace Center (DLR), Institute of Flight Systems, Braunschweig, 38108, Germany

and

Anil Öztürk[3], Volkan Kargın[4], Hakan Aydemir[5] and Ugur Zengin[6]
TAI - Turkish Aerospace Industries, Inc., Ankara, 06980, Turkey

Model-Based Design (MBD) enables iterative design practices and boosts the agility of the air vehicle development programs. Flight simulators are extensively employed in these programs for evaluating the handling qualities of the designed platforms. In order to keep up with the agility provided by the MBD, integration of the air vehicle models in fairly complex flight simulators needs to be addressed. The AVES Software Development Kit (SDK), which is the simulation software suite of DLR Air Vehicle Simulator (AVES), enables tackling the model integration starting from the modeler's desktop. Additionally, 2Simulate, which is the enabling real-time simulation infrastructure of AVES SDK, provides automated model integration workflow for MATLAB/Simulink models using Simulink Coder code generation facilities. This paper presents the successful employment of AVES SDK and the 2Simulate model integration workflow for addressing integration challenges for Pilot-in-the-Loop Testing in AVES.

I. Introduction

FLIGHT simulators have been used by the aeronautics research community for many decades in developing and experimenting with advanced concepts and conducting human factor research. Some of the well-known early examples are ATTAS Ground Based Simulator from German Aerospace Center (DLR),[1,2] NASA Crew Vehicle Systems Research Facility in Ames Research Center[3] and Visual Motion Simulation and Cockpit Motion Facility from Langley Research Center[4]. On the other hand, Air Vehicle Simulator (AVES) of German Aerospace Center (DLR),[5] HELIFLIGHT from the University of Liverpool,[6] NASA Ames Vertical Motion Simulator[7] and SIMONA of Delft University of Technology[8] can be pronounced as the well-known ones which are currently in operation.

It is already a well-employed practice to conduct flight simulator experiments to evaluate the handling qualities of air vehicles.[9-13] Besides quantitative analysis, qualitative ratings can also be collected from the pilots by incorporating flight simulators in the air vehicle design process. Accordingly, Turkish Aerospace Industries, Inc. (TAI) and DLR are collaborating on conducting flight simulator experiments on the DLR Air Vehicle Simulator (AVES) which is a modern research simulator facility that has been operating at DLR Braunschweig.[5]

[1] Research Scientist, Flight Dynamics and Simulation, Lilienthalplatz 7, 38108 Braunschweig, Germany, umut.durak@dlr.de, AIAA Member.
[2] Team Leader, Flight Dynamics and Simulation, Lilienthalplatz 7, 38108 Braunschweig, Germany, torsten.gerlach@dlr.de.
[3] Design Specialist, Flight Mechanics and Autopilot Systems, Helicopter Group, Kazan, 06980 Ankara, Turkey, anil.ozturk@tai.com.tr.
[4] Technical Specialist, Flight Mechanics and Autopilot Systems, Helicopter Group, Kazan, 06980 Ankara, Turkey, vkargin@tai.com.tr.
[5] Technical Specialist, Avionics and Electrical Systems, Helicopter Group, Kazan, 06980 Ankara, Turkey, haydemir@tai.com.tr.
[6] Manager, Flight Mechanics and Autopilot Systems, Helicopter Group, Kazan, 06980 Ankara, Turkey, ugur.zengin@tai.com.tr.

Recent advances in Model-Based Design (MBD) have brought the aeronautics community the concept of agile model development workflows, where model development is integrated into product development by employing mature code generation practices.[14] MBD enables rapid iteration over aerodynamic configuration and flight control design in the air vehicle design process. Stability and control characteristics of the air vehicle can be investigated in the early stages. Constant quantitative evaluation of the current air vehicle design provides valuable opportunities for the designers for further optimization and tuning. The new iterative design practices have brought agility to the air vehicle development programs. In accordance with that, TAI has been developing their MBD environment, called TAI Indigenous Rotorcraft Simulation (TIRS), and utilizing it in their ongoing rotorcraft development programs.

There are some recommended practices from the aerospace industry for MBD. Estrada et al. introduce best practices for developing DO-178 compliant software using MBD.[15] Miller presents automatic flight code generation practices in Northrop Grumman and introduces a use case from desktop simulation to Hardware-in-the-Loop testing.[16] BAE Systems has a model-based flight control systems development process.[17] Fielding presents a process starting from aerodynamic dataset generation to flight clearance of the aircraft. In this process, the use of engineering simulators is mentioned for model-based flight control system design. Nixon states that in the F-35 project MBD forced them to re-interpret the traditional software development process for flight control systems, and introduces Lockheed Martin Aeronautics' practices of MBD.[18]

However, it is still a challenge to keep up with the agility provided by MBD during simulator experimentation. Flight simulators are complex systems with various heterogeneous data and computation intensive subsystems.[19] Considering that air vehicle models are updated with design iterations, their integration to fairly complex flight simulators becomes repetitive, labor intensive and error prone. Such integration efforts may take weeks to months. Therefore, it is necessary to address these problems by providing infrastructures and establishing a fast flight simulator integration workflows for the air vehicle models.

This paper presents a follow up study for the recently proposed 2Simulate model integration workflow[20] and AVES Software Development Kit (SDK). [21] 2Simulate model integration workflow proposes a MATLAB/Simulink based solution that will enable fast integration of air vehicle models with the AVES Software Development Kit (SDK). AVES SDK is the flexible and adaptable simulation software suite of DLR AVES. 2Simulate is the enabling real-time simulation infrastructure of AVES SDK.[22] The automated model integration capability is provided by 2Simulate for MATLAB/Simulink models using Simulink Coder code generation facilities. Further, AVES SDK provides deployment options starting from the modeler's desktop up until AVES. Hence, it also enables integration challenges to be addressed as early as the modeler's desktop without waiting until the final integration in AVES.

In this paper, we report how AVES SDK and the 2Simulate model integration workflow are employed to support TAI's model-based rotorcraft design process for Pilot-in-the-Loop testing in AVES. The next section will present TIRS, its structure, its utilization for model-based rotorcraft design and its code generation process. Then AVES and its software infrastructure, AVES SDK, will be introduced to the reader. Finally, the application of AVES SDK and the 2Simulate model integration workflow for TIRS will be disclosed.

II. TAI Indigenous Rotorcraft Simulation

TAI Indigenous Rotorcraft Simulation (TIRS) is an in-house tool being developed by TAI to support rotorcraft design activities including flight mechanics design and analysis, handling qualities analysis, automatic flight control system design and real-time flight simulation. The approach to developing TIRS is based on physical modeling of all the rotorcraft components individually in a modular structure. Then, the contribution of each component to the equations of motion is calculated based on the detailed rotorcraft characteristics. The complex interactions among these components are either simplified or omitted regarding the intended use of TIRS. The complexity of modeling in TIRS enables the designer to use it for detailed prediction in the whole flight envelope during the design phase. The model structure is shown in Figure 1 and the modules are explained in detail in the next section.

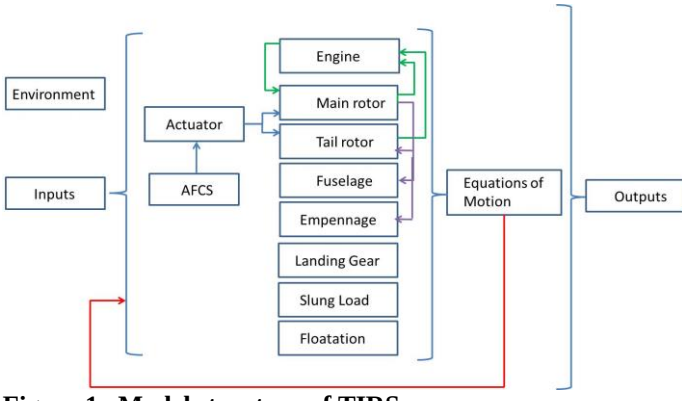

Figure 1. Model structure of TIRS.

A. TIRS Model Structure

TIRS model structure consists of the following modules:[23]

- **Equations of Motion module:** The main objective of a flight dynamics simulation is to describe the motion of the vehicle. This is governed by the well-known rigid body six degree of freedom equations of motion in TIRS.
- **Main Rotor module:** The standard approach used for main rotor modeling is the blade element formulation. Rigid flapping and lead-lag degrees of freedom are modeled and position limitations are also applied. Elastic torsional deflection at the blade tip can be added through a semi-empirical formulation. This deflection is distributed to the whole blade with first torsional mode shape.[24] Solution at Individual Blade Coordinates (IBC) or Multi Blade Coordinates (MBC) is a user selectable option to enhance the flexibility of the module. A finite state inflow model[25] with wake distortion and vortex ring state is implemented. The 0^{th} and 1^{st} inflow harmonics are used for real-time simulations. The 0^{th} harmonic creates the mean inflow and 1^{st} harmonics generate a uniform inflow change at the radial axis of blades.
- **Tail Rotor module:** The standard approach used for tail rotor modeling is also the blade element formulation. However, the Bailey[24] model is also implemented for real-time simulation purposes and thus, flap and lead-lag motions are ignored for improvement in computation time.
- **Fuselage module:** Aerodynamics of fuselage is modeled by using aerodynamic coefficient tables changing with respect to angle of attack, sideslip and position of landing gear. These tables can be either derived from Computational Fluid Dynamics (CFD) analysis or from wind tunnel measurements.
- **Empennage module:** Horizontal and vertical tail is modeled independently from the fuselage for flexibility in the design process. Thus, any other aerodynamic component can easily be added in the model structure. A similar aerodynamic modeling approach as in the fuselage is implemented for empennage, as well.
- **Engine module:** Since the helicopter response is highly coupled and influenced by the dynamic engine and drive drain torque, engine dynamics and the behavior of the rpm governor is modeled as second order engine dynamics with a Proportional Integral Derivative (PID) controller and a collective feed forward term. Integrator wind up is prevented. Multiple engines can be simulated independent of each other. Drive train dynamics are also modeled as a rigid body.
- **Actuators module:** There are three different types of servo mechanisms in helicopters. Primary servos, which are modeled as a first order linear system with rate and position saturation, provide power assisted flight control inputs to the main rotor swashplate mechanism. Flight control inputs are converted into piston deflections and then saturation limits are applied in main and tail rotor piston deflections and rates. Limited authority series actuators which provide short-term stabilization are modeled as second order system with user defined authority percentage. Parallel actuators are modeled as a first order system with dead zone which provides region of zero output. These types of actuators, in conjunction with the series actuators, provide automatic control of the rotorcraft in all axes.
- **Landing Gears module:** Nonlinear strut and linear tire dynamics are integrated. The height of each gear with respect to ground projection is calculated. If a tire is below ground, the landing gear force and moment calculation is activated. The vertical force of the tires is transmitted to the body if loads are below the preload of struts or the strut has reached its maximum limit. Otherwise, the strut forces are transmitted. The ground plane dynamics of tires are modeled as a spring damper system. These forces are limited by maximum available friction forces which are calculated simultaneously. The caster (free tire turn to the velocity vector direction) or steering, left brake, right brake, or no brake are selectable parameter options for each tire.
- **Slung Load module:** The 3DOF rigid body load model is available for preliminary analysis and low computation time. The 6DOF model with flexible cables is also integrated for higher fidelity simulations. The user can define spring damper systems at the vertical plane and friction properties at the ground plane on the load. So, the interaction of the load with ground is modeled. The load can be released by the pilot input.
- **Floatation System module:** Each floatation system is modeled similarly to the blade element approach, such that each float is divided into user selected small elements. The forces and moments are generated using the volume under the water and the sink rate of each compartment.
- **Weight and Balance module:** The effect of fuel change and extension/retraction of the landing gears on weight and balance is modeled, to be observed during simulation.

- **Environment module:** A Von Karman turbulence[26] model with different intensities is available. A Sharp edge and "1-cos" gust models can be applied.
- **Control System module:** The modules described above are all utilized to model the open loop dynamics of a rotorcraft. To be able to conduct closed loop simulations for a rotorcraft, a basic control system module is also included in TIRS. Unlike the other modules, this module is modeled only by using Simulink blocks. The control system module consists of the basic stabilization and attitude control loops provided in pitch, roll and yaw axes. Any necessary upper control loops such as velocity hold, altitude hold etc. can be easily added to the module. Further, this basic control system module can always be replaced with an advanced Automatic Flight Control System (AFCS) module including all the control loops and logic.

All TIRS blocks are being developed in MATLAB script files (*.m) and integrated into Simulink by employing an automated process.[27]

B. Model-Based Rotorcraft Design and Analysis using TIRS

TIRS has been utilized in the whole flight envelope during the design phase. Design problems can be investigated through trim, linearization and simulation:

- **Trim:** The trim condition of a helicopter is the combination of states and control inputs that brings the helicopter into an equilibrium point by means of forces and moments at a given flight condition. Thus, any instant of a stationary flight can be frozen by trim analysis. The trim procedure used in TIRS is based on the Newton-Raphson optimization algorithm[28] with proper constraints and unknowns defined. Controllability, static stability and performance requirements of a model have been investigated by trim analysis. Further, simulations that require integration of multiple components, such as slope landing and autorotation, can be simplified and solved in a very time efficient way via trim analysis. In addition to these analysis, cross-coupling effects of the model can also be investigated.
- **Linearization:** The helicopter model can be linearized around the trim conditions and a nonlinear mathematical model can be represented in a state-space form. The relation between inputs and outputs to small disturbances can be investigated. For small perturbations around the equilibrium, a linearized model can adequately represent nonlinear dynamics. However, for large deviations from the equilibrium point, linear model prediction fails. Both Multiple Input Multiple Output (MIMO) and Single Input and Single Output (SISO) analysis can be performed. Preliminary evaluation of dynamic stability requirements of a model can be done using eigenvalues and eigenvectors of the state matrix. Efficiency of controls and cross-coupling terms can be identified through the control matrix and transfer functions. In addition, linear control designs are based on the linearized models.
- **Dynamic Simulation:** Both trim and linearization are not valid when the system drifts away from the initial condition. Also, discontinuities such as backlash, position saturation, rate saturation cannot be observed without simulation. Highly nonlinear cases such as engine failure, high control inputs, and autorotation flare are analyzed with simulation. In addition, simulation is regarded as the only way to integrate pilots into the design loop. The Pilot-in-the-Loop testing with TIRS is carried out in AVES.

III. AVES and its Software Infrastructure

A. AVES

Since 2013, AVES has been the primary tool of DLR Institute of Flight Systems for flight test preparation and research in flight system, pilot training and simulation technologies. It has two interchangeable cockpits: one for rotorcraft (EC135 ACT/FHS[29]) and the other for airplanes (A320 ATRA[30]). Both can be operated on motion and fixed-base platforms according to the particular needs.[5] The motion simulator consists of a high fidelity 6DOF motion platform and a large 15 channel front projection system with 240 degrees horizontal and up to 95 degrees vertical field of view. To provide the same environment, the fixed-based installation also uses a 15 channel projection system with the same specifications. Both simulation cockpits are equipped with replicas of the real cockpit devices. Controls are simulated using active control loading systems, which can be tuned according to any aircraft specification or research requirements. The real cockpit environment is supported by large operator cabins to control the simulation, observe the simulator trial or develop software right in the place.

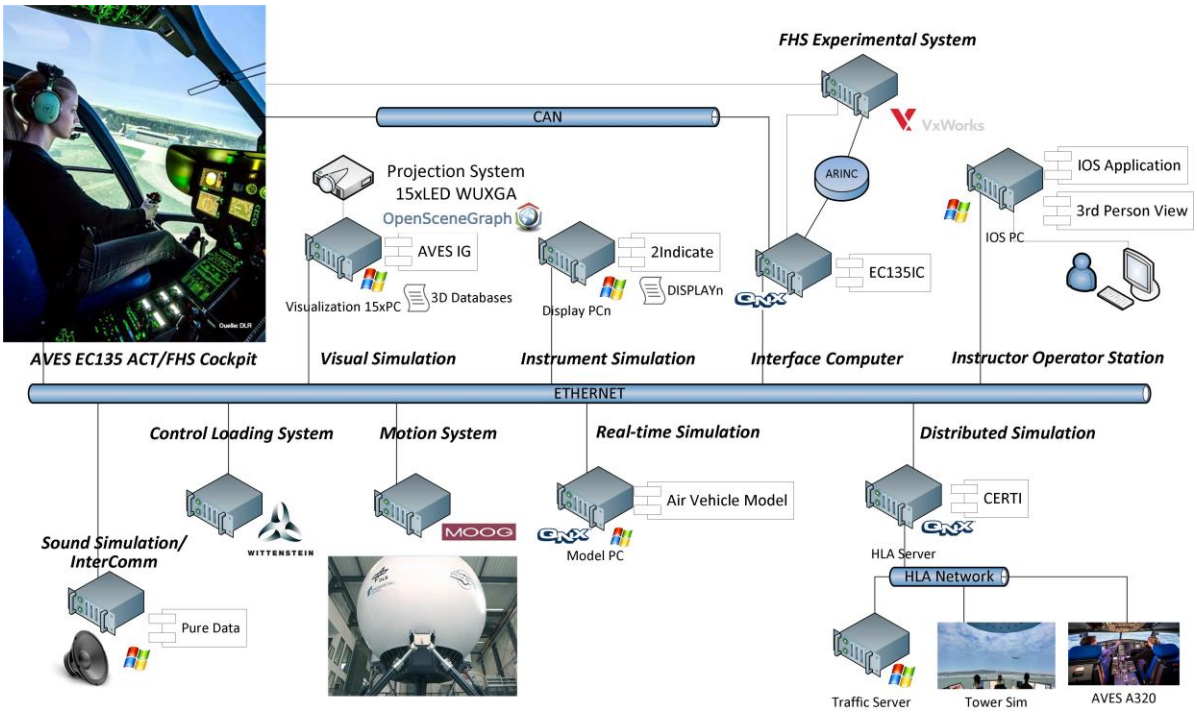

Figure 2. DLR Air Vehicle Simulator (AVES) architecture for rotorcraft simulation.

Figure 2 presents the top level architecture of AVES for rotorcraft simulation. It is composed of the AVES EC135 ACT/FHS Cockpit which possesses a control loading system from Wittenstein AG.[31] It can be located on a motion platform from Moog.[32] The OpenSceneGraph based AVESImageGenerator (IG) drives a 15 channel projection system.[5] High Level Arhitecture (HLA) based distributed simulation capabilities allow the rotorcraft simulation to interact with external systems such as other flight simulators, a tower simulator or the Traffic Server.[33] The FHS Experimental System provides hardware-in-the-loop capabilities for flight test preparation for the EC135 ACT/FHS.[34] The Instructor Operator Station (IOS) provides the IOS Application to manage the simulator experiment and a Third Person View. The Interface Computer (IC) is the data management node on the QNX Real-time Operating System (RTOS) that enables the real-time simulation with various software and hardware components. Real-time simulation refers to the Air Vehicle Model that runs either hard real-time on QNX RTOS or soft real-time on Microsoft Windows.

B. AVES Software Development Kit

AVES SDK is the software infrastructure of AVES. It consists of a set of source code, project files, libraries, executables and scripts. It proposes an organization structure and development rules. The main motivation of AVES SDK is to reduce integration time to AVES by involving the AVES users in the development and integration activities. AVES SDK can be used in all Software-in-the-Loop, Simulator-in-the-Loop and Simulator modes. The Software-in-the-Loop (SIL) mode is used for integrating and testing these air vehicle models or flight systems in a desktop environment and in a real-time Software Test Device (SWTD) environment in a systems integration laboratory setting that also provides device I/O capabilities. The Simulator-in-the-Loop (SimIL) model enables integration and testing in AVES utilizing the developer station in the simulator control room rather than the final target environment, thus it provides local debugging opportunities. These modes provide AVES users with the capabilities to start mitigating the integration risks of their air vehicle models and flight systems early, from the desktop simulation step. The simulator mode is the normal operation mode. The unified set of simulator software assets enables integration and testing to be started from desktop (Figure 3Figure 3) and, using other modes as required, the real integration in AVES is facilitated as fast as possible.

The AVES SDK infrastructure is constructed such that it distinguishes between runtime and development. The Runtime Environment (RTE) is a complete set of ready to use simulator software products that are managed by a suite of deployment scripts. The Software Development Environment (SDE) is tightly connected to the RTE and supports air vehicle model or flight systems development with ready to use development projects managed by a set of source control scripts.

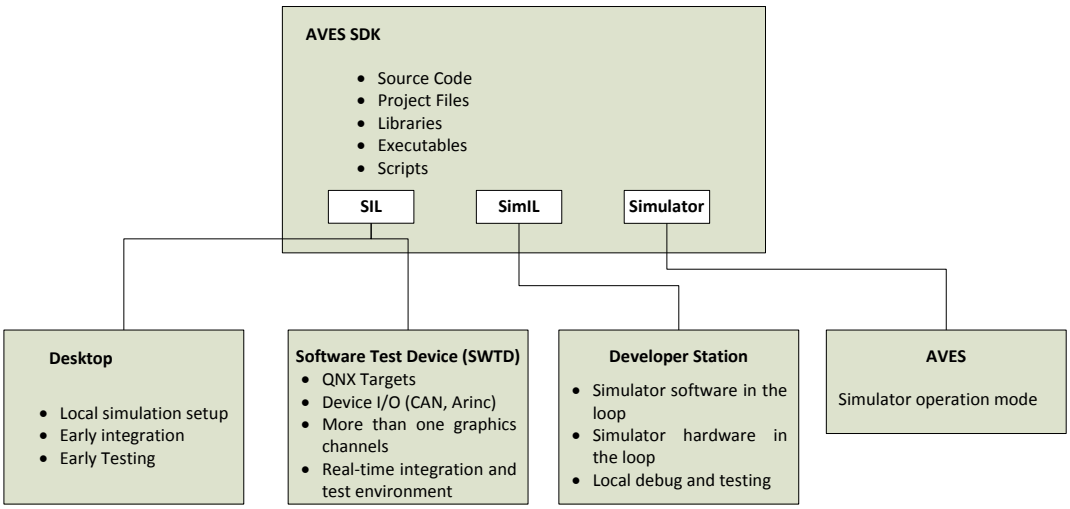

Figure 3. AVES SDK integration and test facilities.

C. 2Simulate and Model Integration

2Simulate is an overall simulation framework to facilitate the integration of models and simulation components such as external devices, data recorders or image generators.[22] It is a C++ real-time distributed simulation framework which is composed of three components, namely 2Simulate Real-Time Framework (2SimRT), 2Simulate Model Control (2SimMC) and 2Simulate Control Center (2SimCC) (Figure 4 (a)).

The core simulation framework of 2Simulate is 2SimRT which provides deterministic scheduling and controlling of real-time tasks. Some example top level tasks are generic SimpleTask, UDPTask for Unified Datagram Protocol, TCPTask for Transport Control Protocol or ModelTask for system models. 2Simulate offers various tasks that extend the top level tasks (Figure 4 (b)). For instance, SimulinkTask extends ModelTask for the real-time simulation of Simulink models. Tasks can be programmed using their pre- and post-initialization and pre- and post-process callbacks. 2SimRT also provides a common database to manage the data.

2SimCC is the graphical user interface that is configured to a control center for specific needs. It is an Microsoft Windows executable which can be customized via configuration files, named the 2SimCC project files. The control center can run, pause or stop various targets. In addition, it accesses the target data dictionaries which can be defined

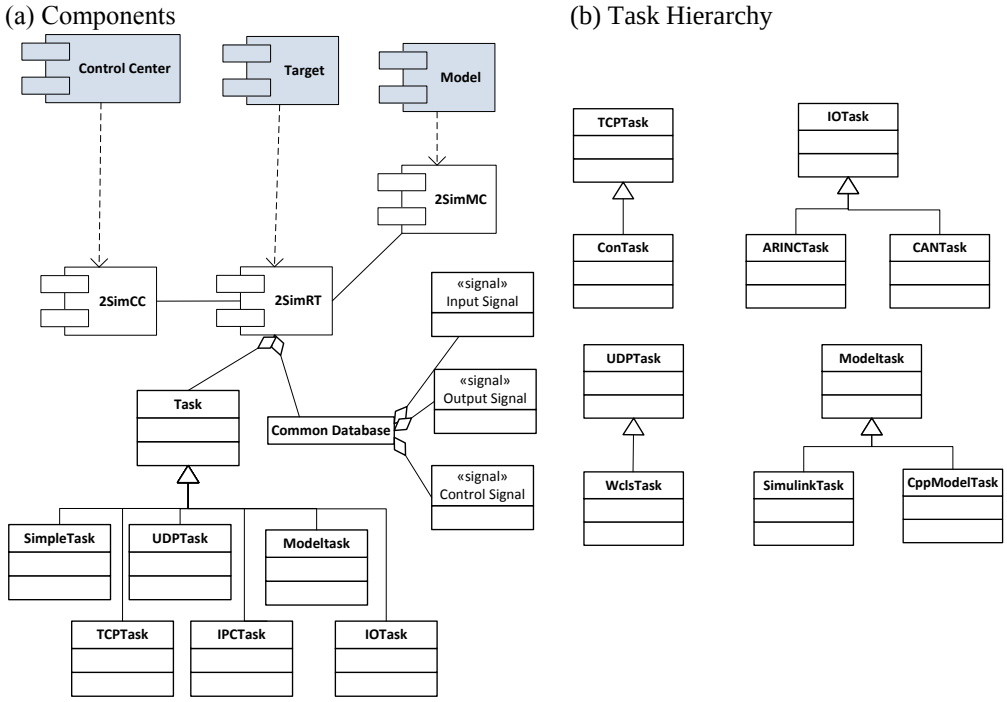

Figure 4. 2Simulate components and task hierarchy.

as the data access mechanisms and enables presenting or editing target data at runtime. It can also enable user management to define and enforce user access rights.

2SimMC is the enabler of model integration in AVES. It is composed of 2Simulate Model Control Source (2SimMC-Source), which abstracts model interfaces for 2SimRT, and 2Simulate Model Control Scripts (2SimMC-Scripts), which include Simulink Coder Target Language Compiler files (TLC files) to specify the 2Simulate target, and m-files to conduct the code generation and build process. There are two types of scripts in 2SimMC-Scripts:

1) The MATLAB build automation scripts that make use of MATLAB command line utilities to control Model Advisor, Simulink Coder and to call some external executables for source control and cross-compilation.
2) TLC files that are used to specify the 2Simulate target.

The model-based methodologies, while providing graphical means to ease the model development, they regard the models as core artifacts and propose an implementation strategy with transforming models to other models using Model-to-Model (M2M) transformations and eventually to code using Model-to-Text (M2T) transfromations.[35, 36] The Simulink Coder is the M2T transformation toolset provided by Mathworks for MBD using MATLAB/Simulink.[37] The code generation starts with a compilation process which ends up with an intermediate representation of the model (*model.rtw*). It is then transformed into C or C++ code. Code generation is controlled by TLC scripts. TLC script is introduced as an interpreted programming language that converts a model description into code. The TLC scripts that specify how to generate code from the model are executed by the TLC. This can be categorized as a template based M2T transformation approach. TLC scripts, as the M2T transformation language, utilize meta-markers. During code generation, TLC queries the dynamic content that is specified by metamarkers from *model.rtw* and replaces the markers with values from the model.

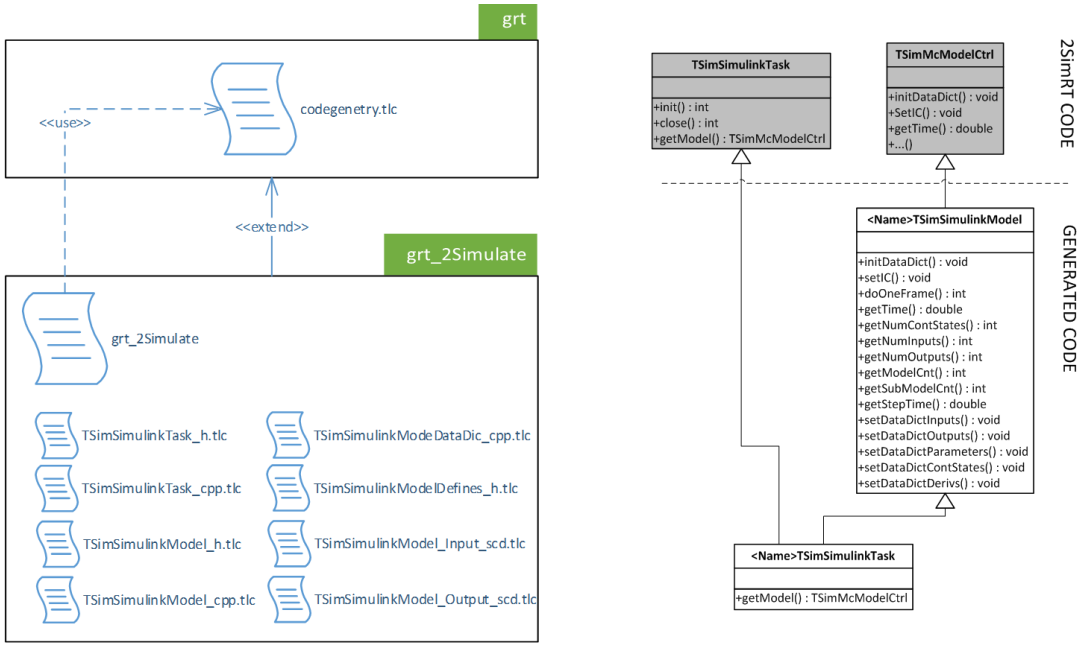

Figure 5. 2Simulate model integration.

TLC scripts provide capabilities to specify targets through customizing the code generation to produce platform or application specific code. For 2Simulate, a target specification called *grt_2Simulate* is implemented by 2SimMC-Scripts TLC files. These files extend a generic real-time target provided by Simulink Coder. During code generation, the top level entry point is the *grt_2Simulate.grt*. It first calls *codegenentry.tlc* to generate model code and then calls all eight 2Simulate TLC files to generate 2SimMC-Component code, whose main static structure is give in Figure 5. The contribution of such an approach lies in the capability it provides to generate the simulator integration code from the air vehicle model. Code generation specified by the TLC scripts in *grt_2Simulate* target leads to implementation of classes that extend the base classes of the real-time simulation infrastructure 2SimRT. Thence, the model is accessable to the 2Simulate programmers as a schedulable real-time task.

IV. TIRS AVES Integration

The integration of TIRS in AVES is carried out in two steps. In the first step, AVES SDK is used to configure a desktop integration and test environment (Figure 6). The model integration infrastructure is employed and the generated TIRS real-time simulation executable is tested in this desktop environment. Upon successful integration in the desktop setting, as the second step, the executable is deployed in the simulator target and tested with the overall software and hardware components of AVES.

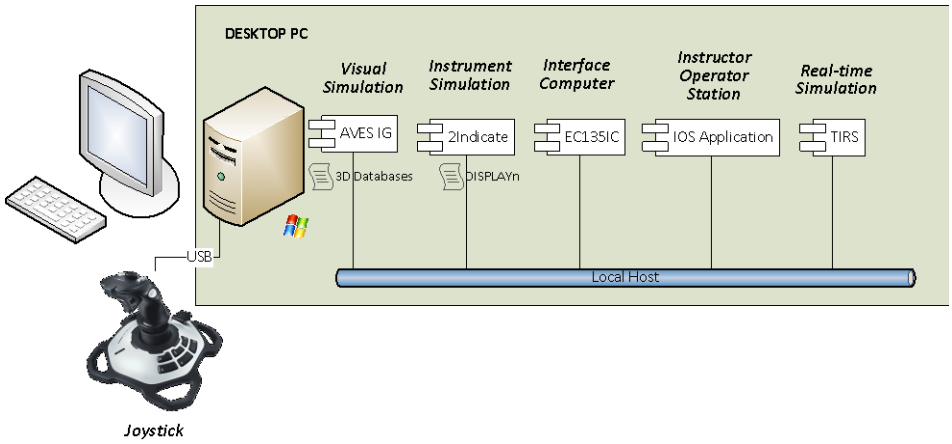

Figure 6. TIRS desktop integration and test environment.

The MATLAB build automation script conducts the automated build (Figure 7). It first checks out the Model Integration Framework, which includes third party dependencies, model application source code and Microsoft Visual Studio Development Environment project. The Model Integration Framework can be introduced as a wrapper for the generated model code. It creates a 2SimRT target using the generated model code. This wrapper code is refactored automatically for model specific parameters. As an example, the solver step size of the model is set as the frequency of the model task in the Model Integration Framework code. After refactoring, the model application code is ready for compilation. Then the code is generation is executed for the model using the target specification.. The next step is to compile the project using the Microsoft command line tool *msbuild*. Finally, the built real-time TIRS executable is copied to the deployment folder.

```
%% TIRS Parameters
ModelName = 'TIRS';
FullModelName = 'model/TIRS';
BaseModelName ='model';

%% SVN Export the Model Integration Framework
cmdStr =['svn export --force  https://svn.dlr.de/AVES/2SimulateRepo ./',...
    ModelName, '_2Simulate/2Simulate'];
status.MIFExport=dos(cmdStr);

%% Refactor the Model Integration Framework
% main
sourceFile = [ModelName,...
    '_2Simulate\2Simulate\dev\src\mdlTarget\mdlTarget.cpp'];
fid  = fopen(sourceFile,'r');

f=fread(fid,'*char')';
fclose(fid);
...
stepSize = num2str(get_param(BaseModelName, 'FixedStep')); %seconds
stepSize=[stepSize, '*1000'];%miliseconds
f = strrep(f,'<model_steptime>',stepSize);
fid  = fopen(sourceFile,'w');
fprintf(fid,'%s',f);
fclose(fid);

%% Build Simulink Model
status.rtwbuild = rtwbuild(FullModelName);

%% Build Windows Project
cmdStr=['msbuild ',  ModelName ,...
    '_2Simulate/2Simulate/dev/win32VC/mdlTarget/TIRS.vcxproj'];
status.MSBuild = dos(cmdStr);
save buildStatus.mat status;

%% Deploy the binary

% Copy image to deploy folder
cmdStr =['cp -f ', ModelName,...
    '_2Simulate\2Simulate\dev\win32VC\bindebug\TIRSD.exe rel\TIRSD.exe'];
status.Deploy = dos(cmdStr);
```

Figure 7. TIRS build automation script.

V. Conclusion

With advances in model-based design, the aeronautics community is practicing agile and iterative design processes utilizing the air vehicle models as the core asset. The flight simulator experimentation is an indispensable activity in air vehicle development efforts. Pilot-in-the-loop testing can be regarded as a part of the x-in-the-loop family. However the flight simulators are complex sytems with various subsystems that posses complicated interdependencies. The integration of air vehicle models to flight simulators are usually labor intensive and error prone.

This paper presents an approach to tackle this integration challenge by utilizing a flexible and adaptable simulation infrastructure and an automated model build process, for agile pilot-in-the-loop simulation experimentation. This approach is applied for integration of Turkish Aerospace Industries (TAI) Indigenous Rotorcraft Simulation (TIRS) in German Aerospace Center (DLR) Air Vehicle Simulator (AVES) for supporting rotorcraft development activities with simulator experimentation. The initial model integration effort for the air vehicle model is reduced to a couple of days and the workload of successive integrations is reduced to just hours. With this study, it is demonstrated that extending the code generation practices of model-based design to application specific targets is an effective approach to disseminate the flexibility and agility provided by model-based design. Further, with a flexible and adaptable simulation infrastructure that can be scaled from desktop to real flight simulator, integration risks can be mitigated early in modeler's desktop.

The future work includes extending such an approach to other x-in-the-loop steps, such as software-in-the-loop and hardware-in-the-loop testing, and enable extensive utilization of flight simulator not only for systems level evaluation, such as handling qualities but also subsystem level validation activities, like testing of automated flight control systems.

References

[1]Saager, P., "Real-Time Hardware-in-the-Loop Simulation for 'ATTAS' and 'ATTHeS' Advanced Technology Flight Test Vehicles," *AGARD Guidance and Control Panel*, 50th Symposium, Izmir, Turkey, 1990.

[2]Klaes, S., "ATTAS Ground Based System Simulator -An Update-," *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Denver, CO, 2000.

[3]Sullivan, B. and Soukup, P., "The NASA 747-400 Flight Simulator: A National Resource for Aviation Safety Research," *AIAA Flight Simulation Technologies Conference*, San Diego, CA, 1996.

[4]Smith, R., "A Description of the Cockpit Motion Facility and the Research Flight Deck Simulator," *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Denver, CO, 2000.

[5]Duda, H., Gerlach, T., Advani, S. and Potter, M., "Design of the DLR AVES Research Flight Simulator," *AIAA Modeling and Simulation Technologies (MST) Conference*, Boston, MA, 2013.

[6]White, M. and Padfield, G., "The Use of Flight Simulation for Research and Teaching in Academia," *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Keystone, CO, 2006.

[7]Advani, S., Giovannetti, D. and Blum, M., "Design of a Hexapod Motion Cueing System for NASA Ames Vertical Motion Simulator," *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Monterey, CA, 2002.

[8]Stroosma, O., van Paassen, R. and Mulder, M., "Using the Simona Research Simulator for Human-Machine Interaction Research," *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Austin, TX, 2003.

[9]Muckler, F., and Obermayer, R., "Performance Measurement in Flight Simulation Studies," *AIAA Heterogeneous Combustion Conference*, Palm Beach, FL, 1963.

[10]van Gool, M., and Weingarten, N. "Comparison of Low-Speed Handling Qualities in Ground-Based and in-Flight Simulator Tests," *AIAA 1st Flight Test Conference. Flight Test Conference*, Las Vegas, NV, 1981.

[11]Kiefer, D., and Calvert, J., "Developmental Evaluation of a Centrifuge Flight Simulator as an Enhanced Maneuverability Flying Qualities Tool," *AIAA Flight Simulation Technologies Conference*, New Orleans, LA, 1992.

[12]Anderson, F. and Biezad, D., "A Low-Cost Flight Simulation for Rapid Handling Qualities Evaluations during Design," *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Boston, MA, 1998.

[13]Landry, L., "Application of Modeling, Simulation and Labs to the F-35 Program," *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Honolulu, HI, 2008.

[14]Ruff, R., Stephans, C. and Mahapatra, S., "Applying Model-Based Design to Large-Scale Systems Development: Modeling, Simulation, Test, & Deployment of a Multirotor Vehicle," *AIAA Modeling and Simulation Technologies Conference*. Minneapolis, MN, 2012.

[15]Estrada, R.G., Sasaki, G. and Dillaber, E., "Best practices for developing DO-178 compliant software using Model-Based Design," *AIAA Infotech@Aerospace (I@A) Conference*, Boston, MA, 2013.

[16]Miller, R., "Automatic Code Generation at Nortrop Grumman," *Mathworks Aerospace and Defence Conference*, Manhattan Beach, CA, 2007.

[17]Fielding, C., "Model-Based Design og Flight Control Systems," *Mathworks Model-Based Design Conference*, Daventry, UK, 2010.

[18]Nixon, D.W., "Flight Control Law Development for the F-35 Joint Strike Fighter," *The Mathworks International Aerospace and Defence Conference*, Newton, MA, 2004.
[19]Allerton, D., *Principles of Flight Simulation*, John Wiley & Sons, Chichester, 2009.
[20]Gerlach, T., Durak, U., and Gotschlich, J., "Model Integration Workflow for Keeping Models up to Date in a Research Simulator," *Proceedings of the 4th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, SCITEPRESS, Wien, Austria, 2014, pp. 125-132.
[21]Gerlach, T, and Durak, U., "AVES SDK: Bridging the Gap between Simulator and Flight Systems Designer," *AIAA Modeling and Simulation Technologies Conference*, Dallas, TX, 2015.
[22]Gotschlich, J., Gerlach, T. and Durak, U., "2Simulate: A Distributed Real-Time Simulation Framework," *ASIM STS/GMMS Workshop 2014*, Reutlingen, Germany, 2014.
[23]Sansal, K., Kargin, V., and Zengin, U., "A Generic Ground Dynamics Model for Slope Landing Analysis," *AHS 72nd Annual Forum*, West Palm Beach, FL, 2016.
[24]Howlett, J., "UH-60A Black Hawk Engineering Simulation Program. Volume I. Mathematical Model," NASA Contractor Report 66306, 1981.
[25]Peters, D., He, C. "Correlation of Measured Induced Velocities with a Finite State Wake Model," *Journal of AHS*, Vol.36, No.3, 1991.
[26]"Military Specification, Flying Qualities of Piloted Airplanes," MIL-F-8785C, 1980.
[27]Durak, U., Öztürk, A. and Katircioglu, M., "Simulation Deployment Blockset for MATLAB/Simulink," *SpringSim-TMS/DEVS*, Pasedina, CA, 2016.
[28]Greenberg, M.D., *Advanced Engineering Mathematics*, Prentice Hall, Inc., New Jersey,1998.
[29]Ockier, C., and Butter, U., "ACT/FHS–an airborne rotorcraft simulator for technology development and research," *AIAA Modeling and Simulation Technologies Conference*, Denver, CO, 2000.
[30]Schneckenburger, N., Klein, C., and Schnell, M., "OFDM based data link for the DLR research aircraft ATRA," *IEEE Integrated Communications, Navigation and Surveillance Conference (ICNS)*, Herndon, VA, 2011.
[31]Wüstenberg, H., Gotschlich, J. and Durak, U., "Anbindung eines aktiven Steuerkraftsystems an eine Echtzeitsimulation," *ASIM STS/GMMS Workshop 2016*, Lippstadt, Germany, 2016.
[32]Seehof, C., Durak, U. and Duda, H., "Objective Motion Cueing Test–Experiences of a New User," *AIAA Modeling and Simulation Technologies Conference*, Atlanta, GA, 2014.
[33]Gerlach, T, Knüppel, A., Durak, U., and Rambau, T., "Running HLA in Real-Time for Flight Simulator Integration," *AIAA Modeling and Simulation Technologies Conference*, Washington, DC, 2016.
[34]Klaes, S., "ATTAS & ACT/FHS System Simulation for Pre-Flight Software and Hardware Testing," *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Monterey, CA, 2002.
[35]Brambilla, M., Cabot, J., and Wimmer, M., *Model-driven software engineering in practice*. Morgan & Claypool Publishers. 2012.
[36]Topcu, O., Durak, U., Oguztuzun, H., and Yilmaz, L., *Distributed Simulation – A Model Driven Engineering Approach*. Springer, Cham, 2016, Chaps. 2, 9.
[37]The Mathworks, Inc., "Simulink® Coder™ User's Guide," Natick, MA, 2015.