AIAA Aviation
22-26 June 2015, Dallas, TX
AIAA Modeling and Simulation Technologies Conference

Model-Based Testing for Objective Fidelity Evaluation of Engineering and Research Flight Simulators

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

and

Artur Schmidt and Thorsten Pawletta[2] [3]
University of Wismar, Wismar, 23952, Germany

Simulator fidelity has been defined as the conformance of a flight simulator to the characteristics of the real aircraft. Objective fidelity evaluation is an engineering approach that attacks the fidelity problem with comparison of simulator and the actual system behavior over some quantitative measures. Testing can be pronounced as the fundamental mean for this comparison. From the utilization perspective, flight simulators are classified as research, engineering and training simulators. Research simulators are both test beds for flight simulator research and computational tools for flight systems and human factors research. Engineering simulators are used for systems development and training simulators are utilized for flight training. While training simulators are subject to rare or few upgrades or modifications in their lifespan, engineering simulators are under occasional and research simulators are under frequent change. The test cases to evaluate the fidelity of training simulators are guided by standards whereas for engineering and research simulators, test cases may present a great variation depending on the scope of change and the use case. These two characteristics of engineering and research simulators, combined with the complexity of today's aircrafts necessitate new methodologies for efficient and effective testing. Model-Based Testing (MBT) targets flexibility and adaptability via utilization of models for specification of test cases and proposes workflows for automatic test case generation. The paper presents an MBT approach for objective fidelity evaluation of engineering and research simulators. The proposed approach is exercised with an infrastructure implementation and an example case study. Thus, evidences are collected that indicate increased efficiency and an effective test process.

Nomenclature

A	=	Acceptor
AL	=	Autopilot Landing
BM	=	Basic Model
CP	=	Cruise Performance
DEVS	=	Discrete Event System Specification
EF	=	Experimental Frame
EM	=	Executable Model
G	=	Generator
GA	=	Ground Acceleration
ICAO	=	International Civil Aviation Organization

[1] Research Scientist, Flight Dynamics and Simulation, Lilienthalplatz 7, 38108 Braunschweig, Germany, umut.durak@dlr.de, AIAA Member.
[2] Ph.D.Student, Computational Engineering & Automation Research Group, PF 1210, D-23952 Wismar, Germany, artur.schmidt@hs-wismar.de.
[3] Professor, Computational Engineering & Automation Research Group, PF 1210, D-23952 Wismar, Germany, thorsten.pawletta@hs-wismar.de.

MB = Model Base
MBT = Model Base Testing
MUT = Model Under Test
PES = Pruned Entity Structure
SES = System Entity Structure
STF = Source Test File
SUT = System Under Test
T = Transducer
TE = Test Environment
TS = Test Scenario

I. Introduction

FLIGHT simulators have been used by aeronautics research community now for more than 30 years in developing and experimenting advanced concepts and conducting aviation human factors 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 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.

Fidelity in flight simulation can be defined as the degree to which a flight simulator matches the characteristics of the real aircraft.[9] As well as on the training efficiency and transfer of training,[10] fidelity of the simulator has an important effect on the quality of the results of simulation experiments for research and development. Objective simulator fidelity assessment provides an engineering standard to qualify the degree of fidelity through objective measures. It approaches the fidelity problem with comparison of simulator and the actual flight over some quantitative cues. Objective simulator fidelity assessment is a tedious and labor intensive effort. Along with that, due to their intended use, engineering and research simulators are subject to a constant change. Furthermore, while the test cases to evaluate the fidelity of training simulators are guided by standards, for engineering and research simulators, test cases may present a great variation depending on the scope of change and the use case.

Regarding the inevitable changes during the lifecycle and the variability in the test cases for engineering and research simulators, combined with the complexity of today's aircrafts, it is required to develop new methodologies for efficient and effective testing. Model Based Testing (MBT) was introduced as a proposal for automating test case generation from a test specification, also called test model, instead of implementing test cases manually.[11] MBT not only automates the testing process, but also enhances the flexibility and adaptability of the testing infrastructure via automating the test case design.[12] Despite the fact that it is widely used in the software testing community, its application in modeling and simulation is quite limited.[13]

Model-based methodologies ask for metamodels to express models. Metamodeling on the other hand requires a complete and accurate specification of concepts. In this study, we referred to simulation theory to fulfill these preconditions. Experimental Frame (EF) is employed for formally specifying simulation test cases, and System Entity Structure (SES)[14] is used for metamodeling. The concept of EF originates from the Discrete Event System Specification (DEVS)[15]. The objective has been an explicit separation between a dynamic model and any experiment with it. EF formally specifies a limited set of circumstances under which a model has to be observed. SES can be defined as an ontology with a limited set of elements that are used to describe various system structures.[16]

Test cases are specified following the formal structure of EF. For generating an executable EF, configurable Basic Models (BMs) for objective fidelity evaluation are provided by a Model Base (MB). BMs usually correspond to atomic or coupled models which are used to compose modular, hierarchical models.[16] The SES is represented by a directed and labeled tree with links to BMs in the MB.

This paper is based on two previous studies: Following the methodology introduced by Schmidt et al.[14], the SES ontology is used for specification of all abstract test cases. Based on the SES and MB, a specific executable test case or a test suite is automatically generated for a flight simulation model under test. The specification of objective fidelity evaluation test cases in SES ontology are mostly adopted from Objective Fidelity Evaluation Ontology of Durak et al.[17]Utilizing the testing infrastructure from Schmidt et al.[14], implementation is carried out in MATLAB/Simulink; BMs are developed as Simulink block and SES is described using SES Toolbox for MATLAB/Simulink.[18] The approach is exemplified with the construction of a sample test suite.

The paper starts with introducing objective fidelity evaluation and automated testing. Then the model-based testing approach for simulations will be presented. The fourth section will proceed with proposing a methodology for model based testing for objective fidelity evaluation. Before the concluding remarks, a sample case study is presented.

II. Objective Fidelity Evaluation and Automated Testing

There is no consensus among researchers on a single index of measurement for simulator fidelity. Further it has strongly been related to the task to be performed with the simulator. There are two approaches to evaluate the fidelity of a simulator. These are subjective fidelity evaluation and objective fidelity evaluation. In subjective approach, user feedback is structured using rating scales to identify the degree of realism felt by the user.[19] But the individual opinions and bias of raters makes it hard to generalize the evaluations across the scales.[20] Objective approaches attack the fidelity problem with the comparison of simulator and the actual flight over some quantitative cues.

As the well accepted global standard for qualification of flight training devices, International Civil Aviation Organization (ICAO) 9625 Manual of Criteria for the Qualification of Flight Training Devices, 3rd Edition[21] specifies the set of test cases for objective validation of simulators. These test cases include comparison of the results from tests conducted in the simulator and aircraft validation data. As an example, in Ground Acceleration Time and Distance (1.b.1[21]) test, it is required to demonstrate that the time and distance required for the simulator to perform a takeoff run conform to the real aircraft. It is recommended to perform a normal takeoff ground roll and to record the time and distance from break release to rotation speed. The test mandates a conformance in time either ±1.5 sec. or ±5% and in distance ±61 m (200Ft) or again ±5%.

Additionally, Aeroplane Simulation Training Device Evaluation Handbook Vol. 1 Objective Testing[22] of the Royal Aeronautical Society (RAeS) explains the implementation of each test and introduces some example cases with some plots, thus enhances the understanding of objective tests introduced in ICAO 9625.

Automated testing can defined as the use of software to control the execution of tests and a comparison of actual outcomes to the predicted ones. Automated testing in objective fidelity evaluation is promoted by the RAeS regarding its benefits; repeatability, ease and rapidity of conducting tests. The features of an automatic testing system is introduced in the RAeS handbook as initializing the simulator with the test initial conditions, trimming the aircraft, creating the stimulus if required, using flight controls and finally checking the simulator output against test criteria. [22]

Braun and Galloway[23] reported their automated fidelity test system that compares directly the flight test results and manual execution of flight tests in simulators. Wang et al.[20, 24] presented Automated Test System that measure force function, evaluation function and transport delay with its non-intrusive interface with operator station. Both efforts on automated testing for objective flight simulator evaluation utilized fixed test descriptions and targets at automation of test case execution. With the MBT approach, we not only target at test execution automation, but also enable automated test case generation to tackle high flexibility and efficiency requirements of objective fidelity assessment for engineering and research flight simulators.

III. Model Based Testing of Simulations

A. Model Based Testing

MBT often targets the functional testing of a System Under Test (SUT) .[25] One interpretation of MBT is shown in Fig.1. Model driven approaches suggest deriving a formal systems model based on the system requirements. System model represents a simplification of the structural and behavioral relationships of the components. In the next step, executable components can be generated from the formal system model. These components form the SUT. Model based testing promotes that the same system requirements are used to derive a formal test model. They describe the intended behavior of the SUT that needs to be tested. From the test model, a specific test case or a collection of test cases, called a test suite, can be generated for an SUT.[26] The idea is that test cases are abstracted in a test model, and then an MBT tool is employed to generate a set of test cases from that model.[12]

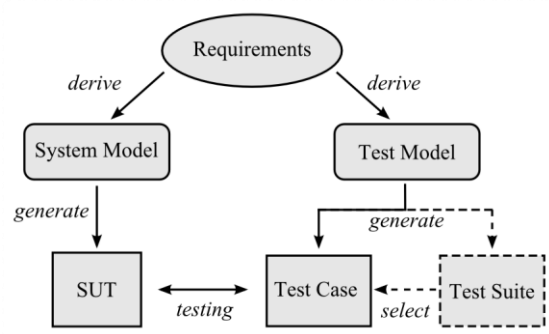

Figure 1 Model Based Testing approach[26]

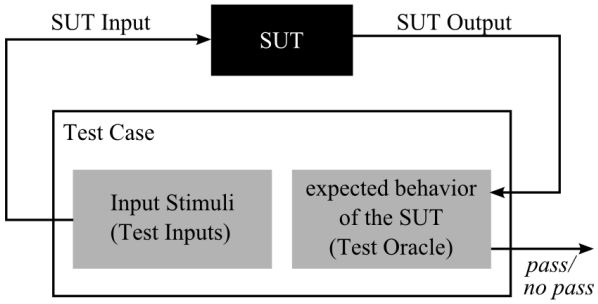

Figure 2 Structure of a Test Case

Weissleder defines a test case as the combination of a sequence of input stimuli to be fed into a SUT, called test inputs, and the expected behavior of a SUT (Fig.2). [25] The expected behavior is often produced using a test oracle. A test oracle contains a judgment unit to make a pass or no pass decision.

MATLAB/Simulink is a popular environment for model based systems development from MathWorks, Inc.[27] It provides a graphical editor, customizable block libraries and numerical solvers for modeling and simulation of dynamic systems. It is widely utilized in flight simulation model development for AVES. Hereby, in this study, we adopt the MBT practices from software testing and propose a testing methodology for flight simulation models developed in MATLAB/Simulink.

Between 2005 and 2008 Zander developed early ideas of employing MBT in a MATLAB/Simulink environment.[28,29] Her Model-in-the-Loop for Embedded System Test – Test Harness (MiLEST) infrastructure provides well-structured libraries for test data generation, test control and test validation functions. MathWorks on the other hand has been providing Simulink Verification and Validation[30] since 2006 for the realization of MBT in MATLAB/Simulink. As in the MiLEST, Simulink Verification and Validation is also providing library blocks that target test functions. Both of this efforts aim at providing a methodology to test the controller models that will be used to generate code to be deployed in an embedded target. Thus, the model that is subject to a test in these two approaches is not necessarily a simulation model or a flight simulation model. In this paper, we propose an MBT approach based on the system theoretical methodologies that are adopted in the simulation theory. Before the concluding remarks, a prototype infrastructure implementation and sample case study will be adduced to make the evidences of applicability traceable.

B. Experimental Frame and System Entity Structure and Model Base Framework

The concept of the Experimental Frame and the System Entity Structure and Model Base (SES/MB) framework was introduced by Zeigler and his colleagues as a part of their system theoretical based approaches for modeling and simulation, DEVS specification.[15,16]

An EF specifies a limited set of circumstances under which a model has to be observed. Following Zeigler,[15] the formal specification of EF is given by the 7-tupel:

$$\mathbf{EF = < T, I, O, C, \Omega i, \Omega c, SU >}$$

where:
T is the time base,
I is the set of input variables,
O is the set of output variables,
C is the set of control variables,
Ωi is the set of admissible input segments,
Ωc is the set of admissible control segments and
SU is a set of summary mappings.

Reader can refer to Traoré et al.[31] for further definition of EFs.

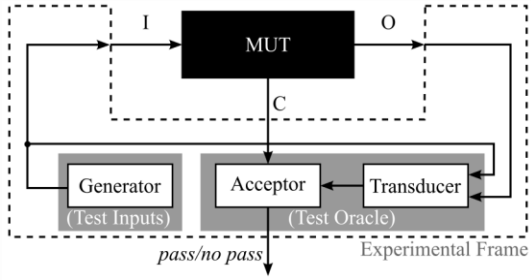

Figure 3 Realization of Experimental Frame

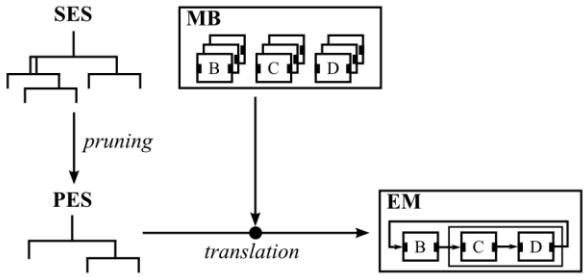

Figure 4 SES/MB Framework[16]

While the EF can be implemented in various ways, Zeigler[15] recommends the implementation of EF as a coupled model, consisting of a generator, acceptor and a transducer, which are connected to the model. We call this model, Model Under Test (MUT). The realization of EF coupled to an MUT is presented in Fig.3 schematically. Test inputs are produced by the generator. The acceptor and transducer form a test oracle. Based on the output variables, the transducer calculates outcome measures in the form of performance indices, comparative values, statistics etc. The acceptor corresponds to a decision unit that decides if an experiment is valid or not. It monitors its inputs and maps them to a specified admissible control segment. The experiment is invalid in the case of a violation of the admissible control segment.

The SES is represented by a directed and labeled tree with links to BMs in the MB. MB can be defined as a repository for BMs that describe the dynamic behavior and represent atomic or coupled systems. Moreover, a set of elements and axioms have been provided in SES to describe system structures.[32] These elements include four node types: (i) entity, (ii) aspect, (iii) specialization and (iv) multiple aspect. Entity represents real or artificial system components. The other node types describe the relationships between their parent and child entities. While aspect nodes denote the decomposition relationship of an entity, specialization nodes represent the taxonomy of an entity. The multiple aspect nodes, finally, represent a multiplicity relationship which specifies that the parent entity is a composition of multiple entities of the same type. Furthermore, specific suffixes are used for a clear separation of the node types. All aspect nodes have the suffix *Dec*, specialization nodes the suffix *Spec* and the multiple aspect have the suffix *MAsp*. Nodes without the defined suffixes correspond to entity nodes.

Figure 4 shows the fundamental structure of the SES/MB framework. The framework combines the SES ontology with the classical workflow of modeling and the simulation of modular, hierarchical systems.[16] It promotes the methodologies for an automatic generation of an executable simulation models using the specification of the system structure in SES and the executable model components in MB.

Pruning is the operation by which a distinct system structure can be derived from an SES. The result is called Pruned Entity Structure (PES). SES Variables represent a kind of user interface and are the basis for the pruning operation. There are two types of SES Variables: (i) related to the system structure and (ii) related to the parameter setting of the nodes. After pruning, *translation* operation is conducted to generate an executable simulation model (EM) based on the information of the PES and BMs from the MB.

C. Proposed Approach

We propose to employ SES/MB for MBT of simulation models. An SES needs to be constructed for specifying the test case structure based on EFs. The proposed top level SES based upon our previous study[17] is as presented in Fig. 5. The node TestScenarioDec indicates the decomposition of entity TestScenario in the two entities (i) MUT and (ii) EF. Referring to Fig.3, an EF consists of a generator (G), an acceptor (A) and a transducer (T). In SES, attributes can be attached to any node. Aspect nodes, such as TestScenarioDec and EFDec, define the coupling relationship between their direct predecessors and successors as attributes. The tuple (MUT.out, EF.in) shows that the output of the MUT is connected with the input of the EF, etc. The entity nodes G, A and T needs to be specialized by their successor nodes, GSpec, ASpec and TSpec and application specific generators, acceptors and transducers need to be define as leaf nodes. Then in these leaf nodes, it is required to define attributes that references to BMs in the MB and the parameter setting of BMs.

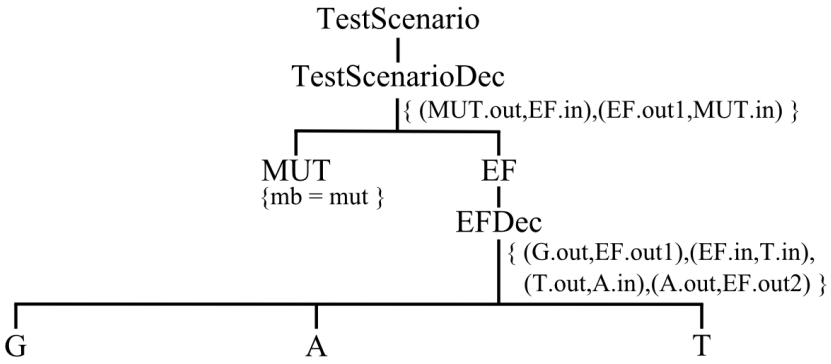

Figure 5 SES structure for EF

The MB will provide a set of BMs, which map different types of generators (G), acceptors (A) and transducers (T). MB will also contain MUTs for the generation of executable test scenarios.

As presented in Fig. 6, Source Test Files (STF) are introduced as scripting interface for the Test Environment (TE). Using STF, the user can specify the MUTs and the test cases that shall be performed on the MUTs. A specific MUTi and EFj will be selected via value assignments to the SES Variables in the STF. TE then will interpret the STF for each test case. First, for each MUTi, the corresponding set of EFs will be identified. Then for each EFj from this set, a specific SES Variable configuration will be picked and sent to the SES/MB framework. The SES/MB framework will generate an executable test scenario (TS) as a coupled system of MUTi and the EFj. The TE then will execute the TS; collect the actual test results and proceeds with the next EFj. After all the specified EFs are executed on the MUTi, the next MUTi and EF set will be selected from the STF and the test cycle will run again. Finally, the results of all the tests will be interpreted by the Test Evaluator. This component is proposed to compute additional statistics, prepare documentation and present the results to the user.

The prototype infrastructure implementation of the proposed approach is done in MATLAB/Simulink and validated using a test case from robotics domain in our previous study.[14] The following section will try to present how this proposed approach can be employed for objective validation of flight simulation models.

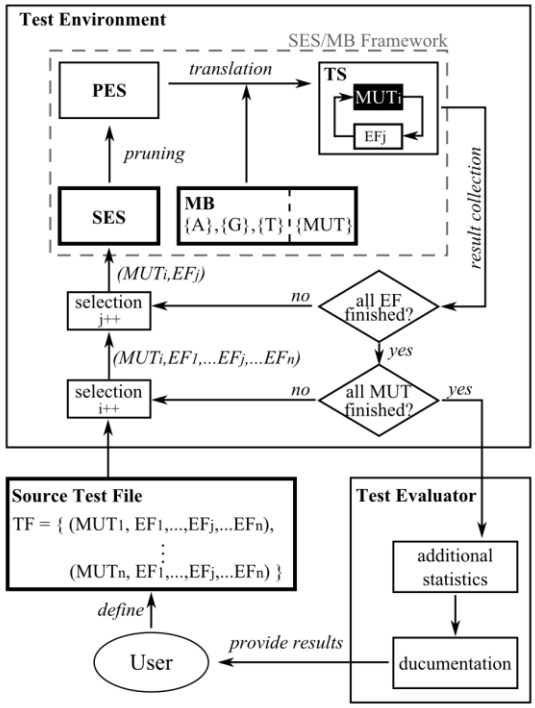

Figure 6 Testing Infrastructure based on SES/MB Framework[14]

IV. Model Based Testing for Objective Fidelity Evaluation

To apply the proposed approach for objective validation, based on our ontology for objective flight simulator fidelity evaluation,[17] an SES has been constructed. A simplified excerpt from this SES has been depicted in Fig. 7. In this graph, *G* is specialized in an entity *Ground Acceleration (GA), Autopilot Landing (AL)* or a *Cruise Performance* (*CP*), *T* in *Ground Acceleration Time* (*GA_Time*) or *Ground Acceleration Distance* (*GA_Distance*) and *A* in *Initial* or *Continuation*.

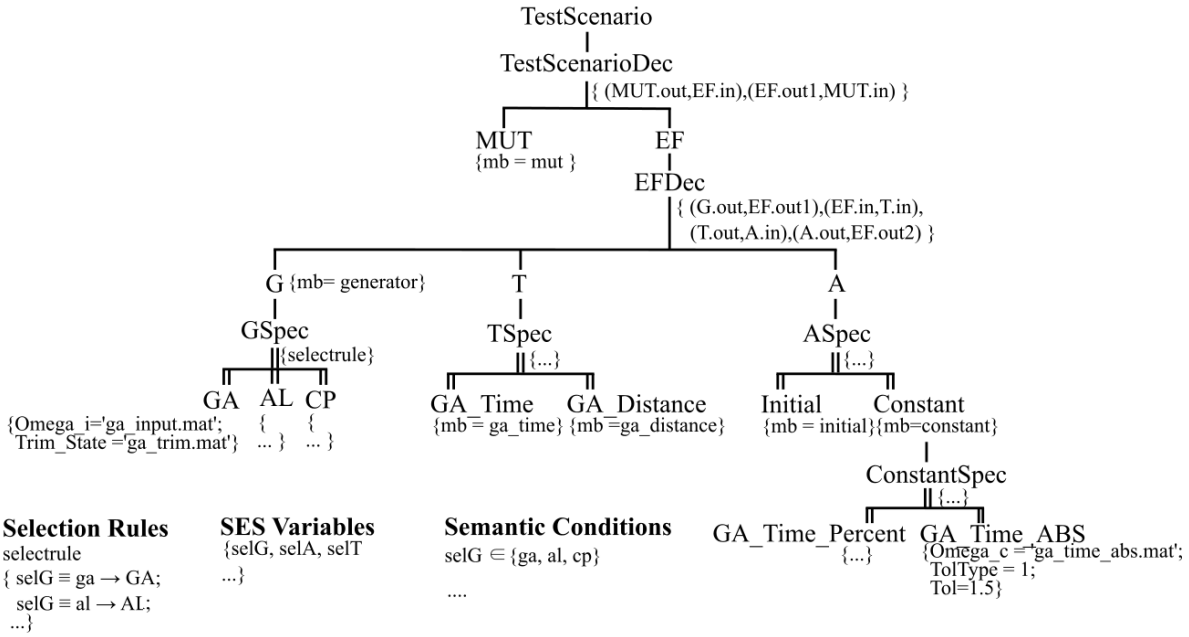

Figure 7 A Simplified Excerpt from Objective Validation SES

A test case that is specified in ICAO 9625 is capture in SES with a generator, transducer and an acceptor. Generators references to a BM in the MB that basically initializes the model with the defined trim file and applies the defined inputs to the model. While the reference to the MB is specified as the attribute of G, the inputs and trim files is cited in the leaf nodes. For GA, the inputs are defined with the attribute *Omega_i* which references to *ga_input.mat* file and trim file is defined with attribute *Trim_State* that references to *ga_trim.mat* file. Transducers interpret the outputs of the model and compute the outcome measures that will be subject to comparison for validity. Since every test case defined in ICAO 9625 possesses its own outcome measure, in SES a transducer is defined for each test case. *GA_Time* refers to the transducer for Ground Acceleration test and computes time to reach rotate speed, Vr, from break release. The BM that conducts this computation, namely *ga_time*, is referred in *mb* attribute of *GA_Time* entity. Acceptors decide upon the validity of the experiment by comparing the outcome measures with the admissible control segments. In case of flight simulation, admissible control segments are flight test data and tolerances to be checked against have been defined in the ICAO 9625. There may be various kinds of acceptors but two of the well applied ones can be seen in Fig. 7. Initial refers to BM in the MB that compares the initial value and Constant refers to the BM that makes the comparison for a constant value. In the specializations of these entities, the admissible control segments and tolerance are defined as attributes. As an example the acceptor entity, Ground Acceleration Time Absolute Value (*GA_Time_ABS*) indicates that the control segment is given in *ga_time_abs.mat* file in it attribute *Omega_c*. Referring to the previous explanation of Ground Acceleration Time and Distance test, it specifies the tolerance type as absolute value with setting *TolType* attribute to 1 and sets the tolerance as 1.5 by *Tol* attribute, which means ±1.5 sec conformance in time between the simulator and the real aircraft.

$$\textbf{SES\_Vars = \{selG=ga, selT=ga\_time, selA=ga\_time\_abs\}} \quad (1)$$

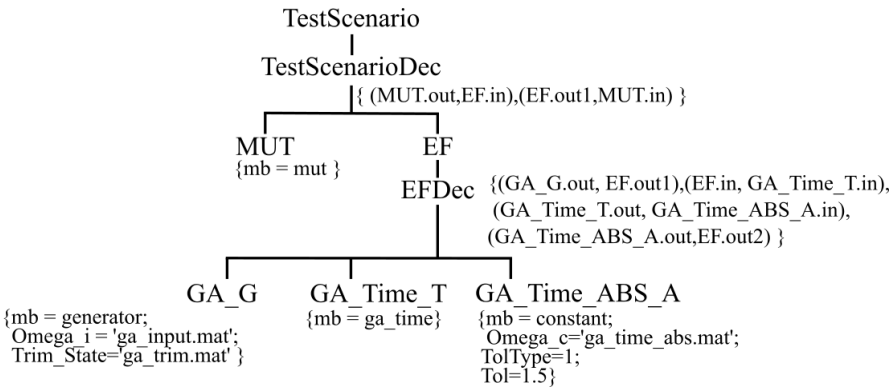

Figure 8 PES Example

As already introduced, SES Variables are utilized as a kind of user interface for the pruning operation. The *pruning* operation targets at deriving a decision-free tree, called PES, with corresponding parameter settings from an SES. Before the pruning operation, all SES Variables must be assigned a value from the range set specified in Semantic Conditions. By evaluating the SES Variables and Selection Rules, all variability in structure and parameter setting will be resolved during the pruning operation. As an example following the SES Variable configuration given in (1) will lead to PES that is depicted in Fig. 8.

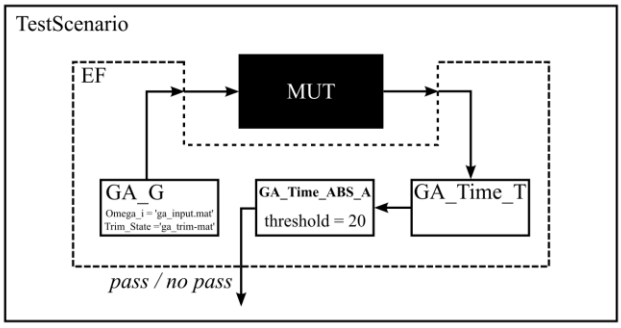

Figure 9 Generated executable test scenario based on the PES and the corresponding MB

The required information to generate an executable test scenario, such as references to the BMs in the MB, their parameter setting and the modular, hierarchical structure as well as the coupling relationships is available in the derived PES. The translation operation is carried out by scripts that accesses all the required information from the PES tree and uses the BMs in MB to generate an executable test scenario. The representative structure for the generated executable test scenario using the PES provided in Fig. 8 is presented in Fig.9.

V. The Prototype Infrastructure Implementation and an Example Case Study

The prototype infrastructure implementation aims at exercising the presented approach in order to collect evidences of its applicability. In this prototype implementation an SES has been constructed that targets a small subset of test cases that are defined in ICAO 9625. This subset includes Ground Acceleration Time and Distance (1.b.1[21]), Autopilot Landing (2.e.5[21]) and Cruise Performance (1.d.3[21]). Generators, transducers and the acceptors are specified for these tests using the MATLAB frontend. The prototype implementations of the corresponding BMs have been conducted and a representative MB has been constructed. The implementation of the automation scripts for test case generation is prototyped. And finally a sample test case generation is exercised.

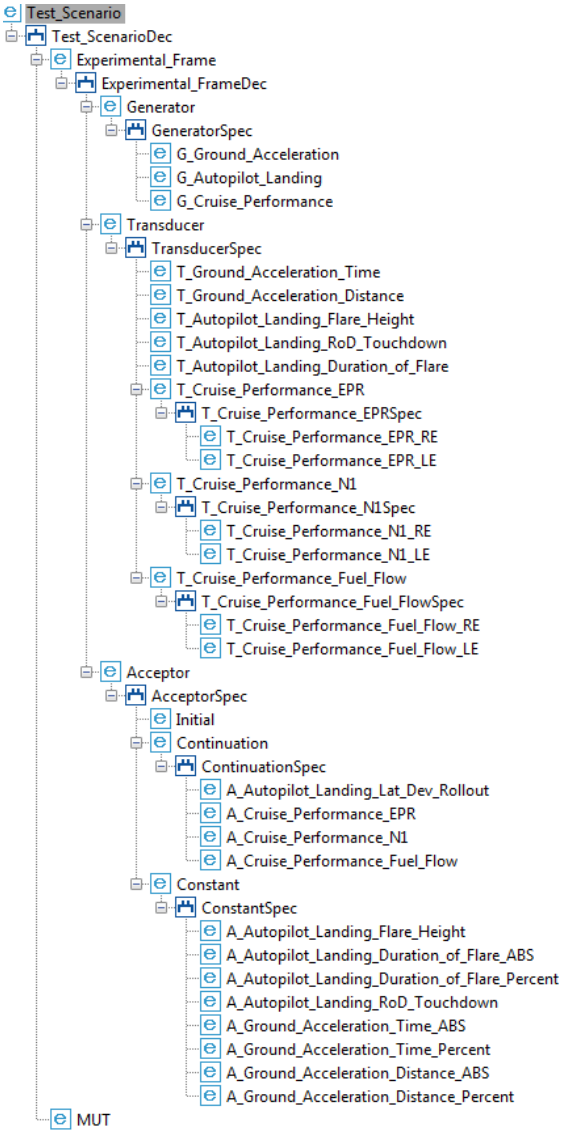

Figure 10 Sample SES Implementation

	Source Comp.	From Port	Sink Comp.	To Port
1	Experimental_Frame	1	Transducer	1
2	Generator	1	Experimental_Frame	1
3	Generator	1	Transducer	2
4	Transducer	1	Acceptor	1

Figure 11 Sample Couplings of the Experimental_FrameDec node

	Attribute Name	Value
1	Omega_c	'cruise_performance_n1.mat'
2	TolType	0
3	Tol	0.03
4	ResultsFile	'cruise_performance_n1_results.mat'

Figure 12 Sample Attributes of the A_Cruise_Performance_N1 entity

	Selection	Condition
1	A_Autopilot_Landing_Lat_Dev_Rollout	Acc_Choice == 21
2	A_Cruise_Performance_EPR	Acc_Choice == 22
3	A_Cruise_Performance_N1	Acc_Choice == 23
4	A_Cruise_Performance_Fuel_Flow	Acc_Choice == 24

Figure 13 Sample Selection Conditions of ContinuationSpec node

	Variable Name	Variable Value
1	Acc_Choice	21
2	Tra_Choice	6
3	Gen_Choice	1
4	Acc_Type_Choice	2
5	Engine_Choice	0

Figure 14 Sample SES Variables

	Semantic Conditions
1	ismember(Gen_Choice,[1:3])
2	ismember(Tra_Choice,[1:8])
3	ismember(Acc_Type_Choice,[1:3])
4	(Acc_Type_Choice == 1) | (Acc_Type_Choice == 2 & ismember(Acc_..
5	ismember(Engine_Choice,[0,1])

Figure 15 Sample Semantic Conditions

Figure 10 presents the SES, which is created using the SES Toolbox for MATLAB/Simulink[18]. Test Scenario is decomposed into the experimental frame and the MUT, which in this case is a Flight Dynamic Model. Experimental frame is decomposed into a generator, transducer and an acceptor. A sample coupling is depicted in Fig. 11 for the elements of the experimental frame. Attributes are defined for each entity. Fig.12 exemplifies this for the acceptor for checking N1 value in the cruise performance test (*A_Cruise_Performance_N1*). It says that the reference value, control segment, will be read from a mat file name *cruise_performance_n1.mat*. The tolerance type is relative tolerance, specified by the value *0* and value of the tolerance *%3*. It also specifies a file that the results shall be recorded.

Specialization nodes enable to define different kinds of the parent entity. Specific generators, transducers and accepters are structured using specialization nodes. As an example, reader can follow from Fig. 10 that three different generators are defined for ground acceleration, autopilot landing and cruise performance tests. Selection conditions are used to set the specific configuration for a particular test case. In other words, as shown in Fig. 13, the conditions of selecting a particular child entity of a specialization node are specified. Selection values that are

presented in Fig. 14 are then used during pruning. The selection variables (SES Variables) are constraint by semantic conditions which specify the valid ranges of the variables. An extract from the semantic conditions is given in Fig. 15.

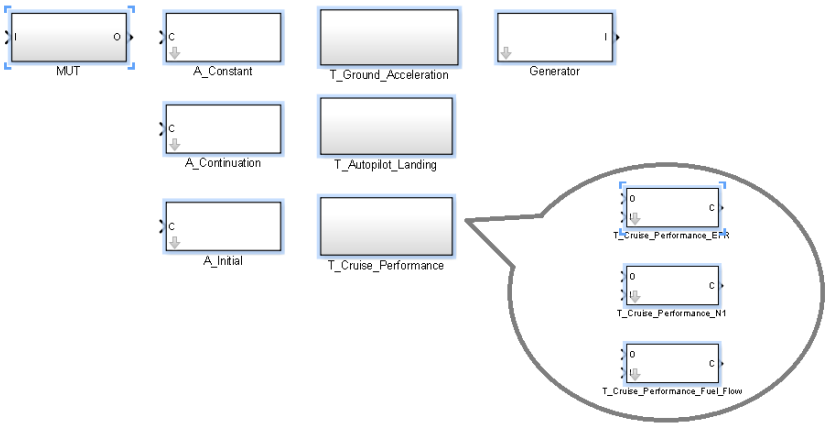

Figure 16 An Excerpt from Model Base

Figure 16 presents an excerpt from the MB which contains the blocks that the test case is made up of. Source Test File is then scripted as exemplified below in Fig. 17 for setting SES variables, pruning the SES and generating the test model via translation.

```
% Source Test File

%% Load a SES
objSES = load('..\SES_DLR\AVES_Objective_Validation.mat');
objPES = pes(objSES.new);

%% Set SES Variables
SES_Vars = { {'Acc_Choice',31},...
             {'Acc_Type_Choice',3},...
             {'Tra_Choice',6},...
             {'Engine_Choice',1},...
             {'Gen_Choice',3} };

%% Pruning
objPES.prune(SES_Vars);

%% Translation
modelTestcase = new_system(objPES.getRootPath());
open_system(modelTestcase);
modelBase = load_system('..\Data_Base_and_Model_Base_DLR\AVES_Objective_Validation_Model_Base.mdl');
modelTranslator(objPES,modelTestcase,modelBase);
```

Figure 17 Example Test Source File

With the SES variables specified in Test Source File, the test case is automatically generated. The *MUT* is copied and connected to an *Experimental_Frame* block. In this block, appropriate generators, transducer and acceptor are copied from the MB and connections are made using the coupling specifications in SES. Figure 18 presents the automatically generated test scenario in MATLAB/Simulink.

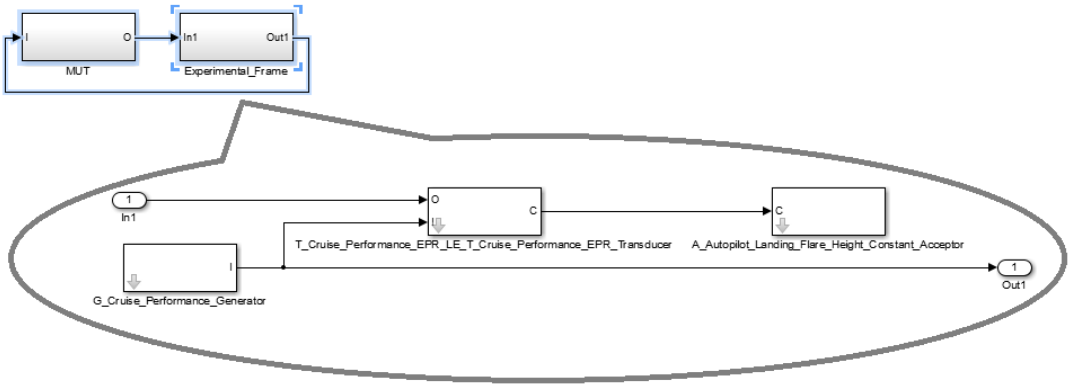

Figure 18 Automatically Generated Test Case

VI. Conclusion

The fidelity of the engineering and research simulators has an important effect on the quality of the results of simulation experiments for research and development. Furthermore their fidelity evaluation is more challenging then training simulators since engineering and research simulators are subject to more frequent change and the test cases for engineering and research simulators may present a great variation depending on the scope of change and the use case. These two characteristics of engineering and research simulators, combined with the complexity of today's aircrafts require more efficient and effective testing methodologies.

In this study, Model-Based Testing approach is presented for flight simulator objective fidelity evaluation. The approach is developed based on Experimental Frame and, System Entity Structure and Model Base Framework. Test models are specified using System Entity Structure and transformed into executable tests employing components from a Model Base that consists of basic blocks for Experimental Frames. Thus, not only the execution of the test cases, but also the generation of the test cases is automated.

As the Model Base that encompasses Experimental Frame components constitutes a reusable asset library for model testing, the adaptability is fostered. The transformation tool automatically generates executable test cases from a test specification model. Hereby, it advances the efficiency and the effectiveness.

A prototype infrastructure implementation is carried out in MATLAB/Simulink. The proposed approach and developed infrastructure is exercised in a case study. A sample SES is constructed for a small subset of objective tests described in ICAO 9625 as well as the implementation of the corresponding Basic Models that constitutes a sample Model Base. As an example, a test case is automatically generated. With the implementation and the test case, valuable evidences are collected. Whilst efficiency is implied by the success of automation in test case generation, effectiveness of the test approach is indicated by the effectual employment of the reusable asset library.

As the applicability and the productivity of the approach are attested, the next step is planned for the development of a production SES and Model Base that will composed of a wide selection of test cases from ICA9625. Then the infrastructure is planned to be released to the flight dynamics model developers in DLR Institute of Flight 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, 22-25 May, 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] Rehmann, A. J., Mitman, R. D., and Reynolds, M. C., " A Handbook of Flight Simulation Fidelity Requirements for Human Factors Research," Crew System Ergonomics Information Analysis Center, Wright-Patterson Air Force Base, OH, 1995.

[10]Longride, T., Bürki-Cohen, J., Go, T. and Kendra, A., "Simulator Fidelity Considerations for Training and Evaluation of Today's Airline Pilots," *Proceedings of the 11th International Symposium on Aviation Psychology*, Columbus, OH, 2001.

[11]Zander, J., Schieferdecker, I. and Mosterman, P., "A Taxonomy of Model-Based Testing for Embedded Systems from Multiple Industry Domains," *Model-Based Testing for Embedded Systems*, Boca Rato, CRC Press, 2012, pp. 3-23.

[12]Utting, M., and Legeard, B., *Practical Model-Based Testing: A Tools Approach*, Morgan Kaufmann Publishers Inc., 2007.

[13]Hollmann, D., A., Cristia, M. and Frydman, C., "Adapting Model-Based Testing Techniques to DEVS Models Validation," *Proceedings of the 2012 Symposium of Theory of Modeling and Simulation – DEVS Integrative M&S Symposium*, San Diego, CA, 2012.

[14]Schmidt, A., Durak, U., Rasch, C., and Pawletta, T., "Model-Based Testing Approach for MATLAB/Simulink using System Entity Structure and Experimental Frames," *International Workshop on Model-driven Approaches for Simulation Engineering*, SpringSim'15, Alexandria, VA, 2015.

[15]Zeigler, B., P., *Multifaceted Modelling and Discrete Event Simulation*, Academic Press Professional Inc., London, 1984.

[16]Zeigler, B., P., Praehofer, H., Kim, T., G., *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*, 2nd ed., Academic Press, 2000.

[17]Durak, U., Schmidt, A., Pawletta, T., "Ontology for Objective Flight Simulator Fidelity Evaluation," *SNE Simulation Note Europe*, ARGESIM/ASIM pub. TU Vienna, Vol. 24, No. 2, 8/2014, pp. 69-78.

[18]Pawletta, T., Pascheka, D., and Schmidt, A., "Ontology-Assisted System Modeling and Simulation within MATLAB/Simulink," *SNE Simulation Note Europe*, ARGESIM/ASIM pub. TU Vienna, Vol. 24, No. 2, 8/2014, pp. 59-68.

[19]Perfect, P., Timson, E., White, M., Erdos, R., Gubbels, A. and Berryman, A., "A Rating Scale for Subjective Assessment of Simulator Fidelity," *37th European Rotorcraft Forum*, Gallarate, Italy, 2011.

[20]Wang, C., He, J., Li, G., and Han, J., "An Automated Test System for Flight Simulator Fidelity Evaluation", *Journal of Computers*, Vol. 4, No. 11, 2009, pp. 1083-1090.

[21]International Civil Aviation Organization, "Manual Criteria for the Qualification of Flight Training Devices", 3th ed., ICAO, Quebec, Canada, 2009.

[22] Royal Aeronautical Society, "Aeroplane Flight Simulation Training Device Evaluation," *Handbook Vol.1 Objective Testing*, RAeS, London, 2009.

[23]Braun, D., and Galloway, R., "Universal Automated Flight Simulator Fidelity Test System," *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Rhode Island, 2004.

[24]Wang, C., Han, J., Li, G., and Jiang, H., "Flight Simulator Fidelity Evaluation Automated Test System Analysis," *2008 International Workshop on Education Technology and Training*, Shanghai, China, 2008.

[25]Weissleder, S., "Test Models and Coverage Criteria for Automatic Model-Based Testing Generation with UML State Machines," Ph. D. Dissertation, Humboldt University zu Berlin, 2010.

[26]Roßner, T., Brandes, H., Götz, H. and Winter, M., *Basiswissen Modellbasierter Test*, dpunkt.verlag, Heidelberg, 2010.

[27]The MathWorks, Inc., "Simulink: Getting Started Guide," [online database], URL: http://www.mathworks.com/help/pdf_doc/simulink/sl_gs.pdf [cited: 13 May 2015].

[28]Zander, J., "Model-based testing of Real-Time Embedded Systems in the Automotive Domain," Ph. D. Dissertation, TU Berlin, 2008.

[29] Zander, J., "Model-in-the-Loop for Embedded System Test – Test Harness," [online database], URL: http://www.mathworks.com/matlabcentral/fileexchange/44328-model-in-the-loop-for-embedded-system-test-test-harness [cited: 13 May 2015].

[30]The MathWorks, Inc., "Model-Based Testing," [online database], URL: www.mathworks.de/discovery/model-based-testing.html, [cited: 13 May 2015].

[31] Traoré, M. and Muzy, A., "Capturing the dual relationship between simulation models and their context," *Simulation Modelling Practice and Theory*, Vol. 14, No.2, 2006, pp. 126-142.

[32]Zeigler, B., P. and Hammonds, P., E., *Modeling and Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange*, Academic Press, 200.7