# Approach to Systematic Test Signal Definition for Operation Scenarios of Aircraft Systems

1st Dennis Hillig
*Institute of Aircraft Systems Engineering*
*Hamburg University of Technology (TUHH)*
Hamburg, Germany
Dennis.Hillig@tuhh.de

2nd Frank Thielecke
*Institute of Aircraft Systems Engineering*
*Hamburg University of Technology (TUHH)*
Hamburg, Germany
Frank.Thielecke@tuhh.de

***Abstract*—With rapidly increasing levels of automation, modern aircraft systems become increasingly complex. The designed solutions typically achieve multiple functions by using a highly integrated mix of hardware and software components. With regard to automation, particularly the number of software components increases.**
**For verification and validation activities, such as testing, this intensifies the challenge to define all relevant scenarios to test a system's functionalities and particularly its robustness in the case of unusual, but realistic situations. Hidden failure cases might be missed. Additionally, it becomes infeasible to systematically define the test vectors to study a systems behavior thoroughly due to the immense test space. As a consequence, testing gets more work- and time-intensive.**
**This paper presents a stepwise model-based method to define and refine the stimulating part of the test scenarios in a systematic and modular manner to approach this challenge. The aim is to work out diverse and relevant operational scenarios to validate system functions and to set up realistic test vectors for these scenarios. The paper further focuses on the aim to facilitate automation and re-usability of scenario and functional signal elements to enable continuous testing at different stages of the development process. Concepts of a developed prototype are outlined as well. Finally, the current state of the method is discussed and an overview of the way forward is given.**

***Index Terms*—scenario testing, aircraft systems, test design**

## I. Introduction

In recent years new digital and smart embedded system solutions emerged at a very rapid pace. Such solutions are increasingly used to optimize novel aerospace systems, which integrate mechanical, mechatronical and software components, through automation. However, the more capabilities and functions these integrated intelligent solutions implement, the more apparent the need becomes for new verification and validation (V&V) methods, that can handle the complexity. In particular for safety-critical systems this is crucial. In that context, classical approaches for the examination of system's integrated functions and safety, which rely significantly on manual testing and expertise, are a limiting factor to new developments.
Testing in general is the mainly used V&V method. Scenario-based testing more specifically is one technique to assess the operational, functional behavior at system and overall aircraft level. Scenario-like analyses of technical systems are also commonly used for demonstrations of functionality in an operational context. Yet, two main challenges can be identified, when the system complexity rises. Firstly, it becomes difficult to find the relevant scenarios or operational conditions. Secondly, the implementation of the realistic test vectors complicates significantly.
In order to approach this challenge for aircraft systems, a methodical procedure for a structured and model-based scenario definition was developed at the Institute of Aircraft Systems Engineering of the Hamburg University of Technology. It is intended to cope with the system's complexity through the main ideas of modularisation and hierarchical refinement. The contribution is embedded into the activities to develop a seamless avionics tool chain (s. [1]).
The aim of the developed method is clarified with an example use case of an air conditioning system in the following. Figure 1 shows an overview of the system under test (SUT) and some interfacing elements, such as the aircraft's environment conditions, cabin loads and cockpit commands from the overhead panel. The system is modeled in an open-loop manner w.r.t. the interfacing elements. The actions of these interfacing elements should be explicitly integrated in the scenario definition. Given that the physical system was designed with a representation as a physical model, and the control and monitor (C&M) functions (e.g. pressure control) were designed and implemented, the scenario-based test has the following two objectives: On one hand, the system hardware design should be verified, e.g. in terms of sizing and functionality. On the other hand, the software functions should be validated under realistic operational scenario conditions. To achieve that, realistic input signals or models of the interfacing elements are needed for a given set of scenarios, which also have to be identified. This task is not trivial, particularly as interfaces include numerous continuous signals.
The presented definition process should further facilitate automation and reuse at different stages of the development. With regard to the latter, system component models could be replaced by real hardware, for example, and the scenarios should still be applicable with minimal necessary adaptions. A prototype editor for the presented mission scenario and stimulating test signal definition was developed using Matlab/Simulink. Applied concepts are integrated in the following explanations. Specified information can also be captured as an .xml exchange document.

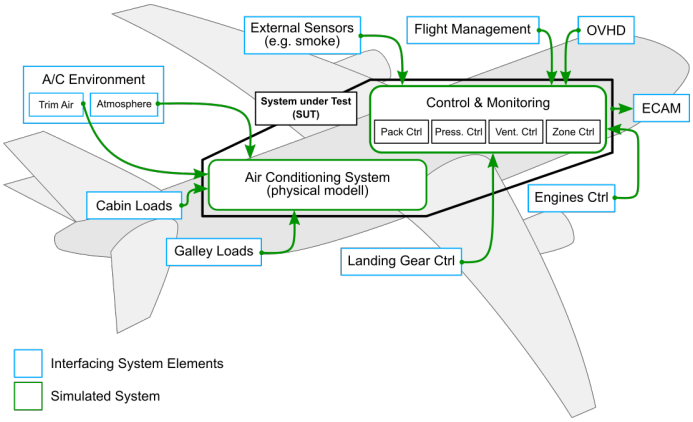


Fig. 1. Example usecase: Air conditioning system.

The paper firstly presents the *fundamentals* of scenario-based testing and the applied mission-based scenario approach. Subsequently the developed *scenario definition process* is outlined. This includes an initial *methodical overview* as well as the different fundamental steps being *scenario specification*, *system and environment definition* and *signal and component implementation*. Finally, a short conclusion is given and the future work is described.

## II. Fundamentals

### A. Scenario Testing

The classical scenario test approach can be categorized as a specification-based test design technique (s. [2]). As equivalently called "black-box testing" this means that requirements, specifications or (interface) models etc. are used to derive test cases without knowledge of the internal structure of the tested system [2]. In that respect scenario testing focuses on sequences of interactions with interfacing systems or components. In software testing literature, also user-centric use cases are utilized equivalently to scenarios [3]. Other specification-based testing methods contrarily mainly rely on the systems input-output space for specification verification e.g. by identifying possible input combinations or partitions with equivalent behavior. Therefore, scenario testing is more suitable to assess systems long-running (*end-to-end*) behavior in a realistic operational situation. This story-like character consequently attains motivational and meaningful context (s. [4]) to discover hidden failures and their possible effects. Models, e.g. formulated in formalisms like state charts or sequence diagrams, are commonly used to describe the possible interaction sequences. The typical scenario definition procedure from [2], towards which this work is oriented, can be reduced to the following two steps:

1) Definition of one main scenario, that describes the nominal sequence of actions
2) Derivation of alternative scenarios based on the given main scenario.

The latter can be diverse, as it includes foreseen operational options, exceptions and intended or unintended incorrect entries. Therefore, these scenarios are often further classified e.g. as positive or negative w.r.t. whether the goal of the interaction is expected to be achieved or not. A trivial aircraft example: If the aircraft is on ground, it should not be possible to retract the landing gear.

As outlined here, scenario testing has the potential to assure safer and more robust system-functions through the diversity of realistic scenarios. Particularly automation functions could be validated early. However, it is also clear that a systematic test of the complete input-output space is not the goal. A scenario targets one specific, but complex set of operations and exploring all possible parameter variations would blow up the number of test cases. Other approaches to black-box testing probably perform better in this respect. Moreover, as a black-box testing technique, coverage of all internal system states is also not the primary intend where a structure-based technique may be more suitable.

### B. Mission-Based Scenario Approach

As aircraft systems are typically integrated subsystems of the overall vehicle, the functional behavior is typically oriented towards the flight mission. For that reason, an operational scenario-based test approach was developed in [5], which models the flight mission in a state-chart as a sequence of fundamental phases, that are represented as atomic states along a scenario path. The major aim is to support the overall system integration tests to observe coupling effects between involved components and software functions. The current application focuses on the use for simulation-based pre-design investigations but the principal transferability to final integration testing is desired. Therefore, the approach intends to complement requirements-based test activities by exploring the operational overall system behavior to uncover undesired effects, with a particular desire to find interactions between ATA chapters.

The virtual test architecture from [5] foresees the partition of stimulating 'test cases' though the scenario state-chart and the evaluation, which is done by observing agent modules running in parallel. These modules could check arbitrary functional behavior or performance of the system. With this approach it is possible to realistically set up operational tests, which enable assessment the overall system's functional behavior during complete flight missions. As the integrated system is investigated, multiple functional specification aspects and operational sequences can be observed in an simultaneous manner.

The scenario concept presented in [5] and [6] uses multi-signal segments in time-series format from a segment library to create continuous signal profiles for each phase. Test signals set up for each phase separately are thereby composed to a complete test vector representing the flight mission scenario. To cover multiple scenarios, the concept allows branching of the scenario paths. The time-continuous signal profiles defined within the phase are complemented by parallel events that are triggered at discrete test points during a phase. The events could for example represent cockpit commands and can trigger

alternative phase transitions of the scenario path. This enables the modeling of events such as an emergency during cruise and resulting alternative flight phases, such as emergency landing and evacuation.
In [5] and [6] the outlined concept was successfully demonstrated at a multi-functional fuel-cell system, which included control and automation functions. However, the complex scenario state-chart has to be created manually and is hardly reusable. The presented method in this paper also applies the same mission-based scenario approach. With the foundations from [5], the method tries to offer a more structured, modular and reusable process for the definition of the stimulating scenario state-chart, that can be automated further.

### C. Related Work

*1) Testing using Scenario-Based System Specifications:* Several works in the recent past adressed the model-based specification of system behavior in the form of scenarios and the test activities on the basis of this specification. To clarify the distinction in the light of scenario-based testing, some works in that field are discussed in the following.
For the specification of reactive system behavior, message sequence diagrams are often used in literature on software engineering to visually define wanted and unwanted system scenarios. This particularly applies in the context of testing. One example is the standardized test specification language TTCN-3 [17] with primary applications for communication systems, which allows the formulation of test cases in the form of sequence diagrams. It is pointed out in [9], that the formalism has the advantage to focus on inter-object communication scenarios without having to specify the objects internal behaviors. Moreover, extensions such as live sequence charts (LSC) [12] exist, which allow the precise definition of system specifications with different modalities, e.g. what must and what may happen. Such sequence charts are also extended in research to describe time constraints (s. [11]) or generally visualize temporal properties such as linear temporal logic (LTL) formulas (s. [13]).
The work in [10] further demonstrates a way how automatic test generation can be achieved with modal scenario specifications. The approach allows the specification of multiple sequence charts for the system functions that can be active in parallel, if an initial sequence of the respective scenario occurs. Due to the granularity of scenario definitions, the method seems quite scalable. Moreover, a coverage criterium was developed to minimize redundant scenario combinations and thus the amount of test cases.
However, the scenario approach of this work is principally different. As mentioned in the previous section it should complement the test activities on the basis of specified requirements. Instead of using a specification model of the system the presented approach models the stimulating environment in an operational context on aircraft level. Expectations on system's behavior are not modeled in the current scenario state-chart and must be evaluated in parallel observation modules or manually in post processing. Reactive behavior (and timeouts) may be used to model environment reactions on system outputs. The presented approach is understood to be more explorative and investigative in the sense to move away from the scenarios that are explicitly defined through the specification. It should be emphasized that running scenario-based specification models in parallel to observe required functional behavior is possible and intended but exceeds the scope of this paper.

*2) Automotive Street Scenarios:* In the context of the validation of highly automated driving functions in the automotive sector, the works in the context of the PEGASUS research project [15] follow a similar operational scenario philosophy as presented in this paper. Street scenarios, defined in a standardized and structured format [14], include static content of the street situation and dynamic elements such as moving cars or pedestrians. The concept allows to execute the scenarios at simulation as well as real world level. Scenarios can also be to remodeled from real operation for further investigation. Nevertheless, a difference for the application in the aircraft domain is the diversity and number of systems in comparison to the variety of traffic situation in the automotive sector.

*3) Signal Segments in Testing:* In [16] a segment-based approach is presented to generate complex signal inputs with a low number of parameters for each segment. This principle is similar to the definition of signal functions in the context of this work. However, the work in [16] conducts a structural test of automotive Simulink models using a search-based algorithm and does not predefine operational scenarios.

## III. Scenario Definition Process

The developed definition process pursues three basic incentives following the modularization and hierarchy principles:

1) Easy set up and management of complex scenario tests makes them less error prone and more complete for increasingly complex systems.
2) Give context and improve automated analysis and documentation by association of signal segments with specific scenario story elements or actions of the SUT and its interfacing environment elements.
3) Maximize the reuse of scenario tests and its elements to reduce effort and make tests more comparable.

The concrete concepts are discussed in the following.

### A. Methodical Process Overview

The developed procedure is structured into 5 fundamental steps. An overview of these methodical steps for the scenario definition is depicted in Fig. 2. The steps can be grouped into three types of activities: *Scenario specification*, *system and environment definition* as well as *signal assignment*.
The first abstracts from the concrete system interface or signals and tries to identify all relevant mission scenarios and their logical sequence of phases and events. At the end of this activity, the structural definition of the state chart is complete and specified up to its basic phase elements and a graphical representation can be produced (e.g. in Matlab/Simulink Stateflow). Furthermore, different scenario test cases can now be

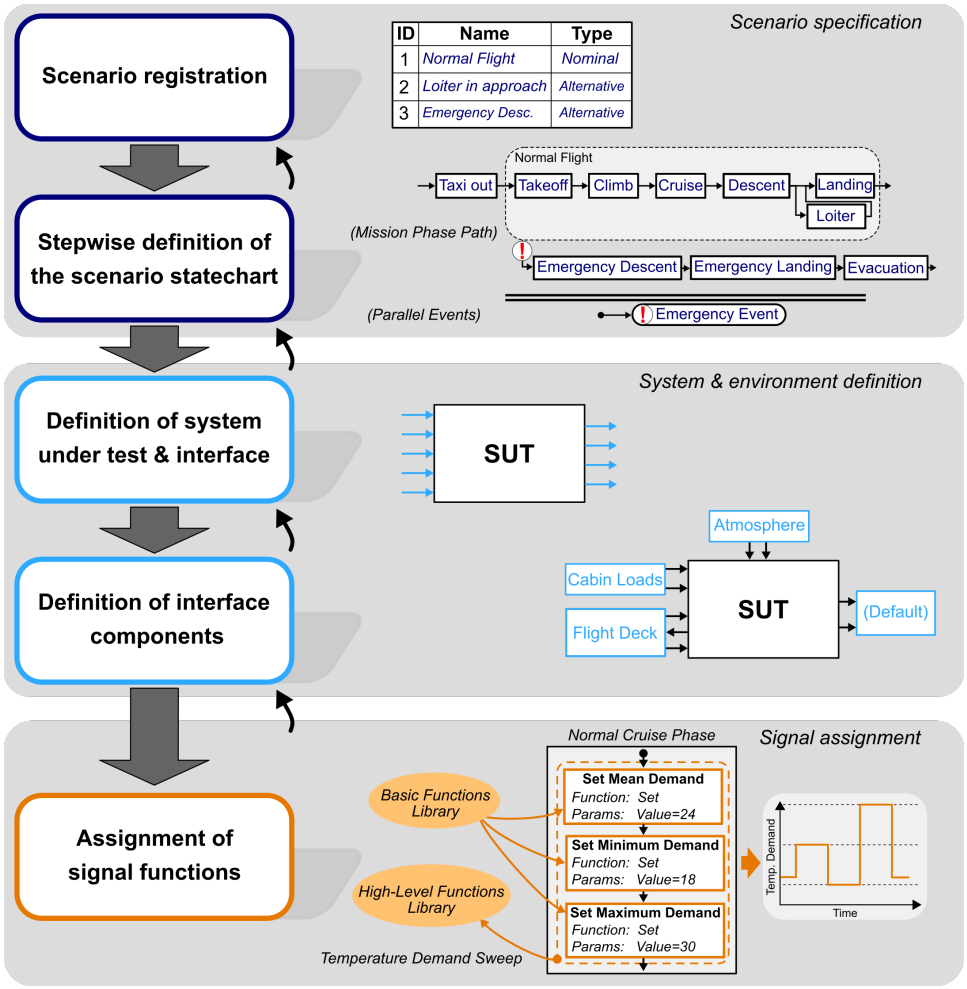


Fig. 2. Methodical steps in the scenario definition process.

derived on the basis of the transition paths through the statechart. However, as the basic phase states are still empty and no interface signals are defined yet, the test cases cannot be applied.

The second activity defines the concrete SUT interface and its environment elements. When finished, all interface signals for the system under test are registered and grouped according to potential environmental elements. These elements can be represented as parallel states within a compound phase state. The scenario test cases defined at this stage could be executed, but do not exercise the scenarios correctly, as no signal profiles are defined yet.

The third and final activity assigns concrete signal profiles to the defined signals. For each phase and each environment element, sequences of signal segment models can be assigned. For re-usability, these models could be linked from a library as parameterized elements. In such way generic signal segments, such as a ramp or set functions can be applied, as well as very specific profiles such as the pressure sensor signal including noise during an depressurization event.

Following the signal assignment the scenario definition process is completed. Enough information is gathered to create executable scenario test cases. Due to the generic setup, an export to arbitrary test languages is principally possible. The implemented prototype realizes executable test cases as state charts in Matlab/Simulink Stateflow and the use with Simulink Test is foreseen. Furthermore the export to the state machine notation SCXML [7] was tested, which is currently further extended as a generic exchangeable test language in a research project.

A more detailed description for the concepts in each step is presented in the following.

### B. Scenario Specification Layer

The scenario specification layer has the intention to identify and structure all relevant operational scenario conditions by performing the two steps *Scenario registration* and *Definition of the scenario statechart*.

*1) Scenario Registration (AC/System-level):*

In this initial step, the scenarios are registered by a textual definition and classified. The primary classification is to whether a scenario represents a nominal sequence of events or an alternative one. Further classification of the latter e.g. as

*positive*, *exception*, *negative* and *misuse* with context to system and aircraft functions still need to be worked out. Optionally a classification for expected frequency of occurrence or links to requirement elements can be inserted. Such additional information should later be used for test case prioritisation and filtering.

For the mission-based scenario approach, two different categories of scenarios were identified:

- Aircraft mission scenarios
- System operational scenarios

The first includes scenarios such as nominal and alternative flight phases (e.g. loiter in approach) or emergency descent. The latter focuses on the system functions and exploits different system operations or configurations and internal or external failures.

*2) Stepwise Definition of the Scenario State-Chart:*

As a second step, each scenario is setup in one integrated state chart structure. Scenarios are implemented by sequences of phase states or event triggers, that could occur during selected phases or phase groups (e.g. in flight) at defined test points, or combinations. Firstly one flight-mission phase path is defined for the main nominal scenario. This scenario should equivalently represent the nominal system operation. Subsequently the alternative scenarios can be defined by branching the nominal scenario at a selected phase and proceeding with alternative mission phases and by defining parallel trigger events that mark the start of the scenario. Phase end conditions could be defined textually for the transition between the phase states, which then have to be refined by state chart variables later. Scenarios can partially share a scenario phase path and end by final phases or explicitly branching into other scenario paths (e.g. alternative flight mission with the same landing procedure). An implemented example for the developed prototype is depicted in Figure 3 - 5. As shown in Figure 4, a scenario path, such as the nominal flight mission in the given example, can be constructed with regard to the taken transition path. Alternatively, as depicted in 5 for the emergency case, a tabular view of the state path could be used. The second figure shows, how the alternative scenario is constructed by branching out of the nominal one. This structure-based editing can be further complemented by a hierarchical tree view for all existing states, given on the left side of the figures. In addition to this structural edition of scenarios, the illustrated conceptual prototype realizes a graphical state machine view in Matlab/Simulink Stateflow. As both perspectives are transferable, a graphical state chart as depicted in Figure 3 can be set up easily by this hybrid editing that combines graphical and structural information. Parallel events can implement

- initial trigger for scenario paths, e.g. emergency,
- trigger for usual phase-internal operations without effect on transitions (e.g. different cabin loads),
- trigger for failures (e.g. cabin smoke / fan failure) or
- combinations.

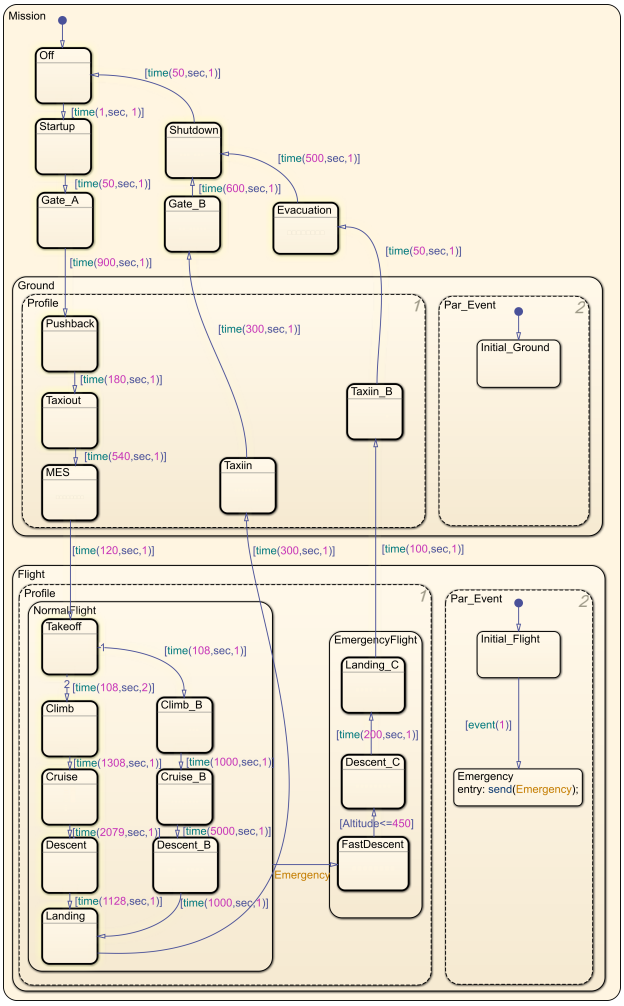


Fig. 3. Exemplary scenario state chart (in orientation to [5]).

They occur at discrete test points during the phases. In the current concept and in orientation to [5], test points apply for phases that have a fixed minimum execution time defined (e.g. cruise time). By defining an integer sampling number for each phase, the given time frame is partitioned in as many parts, where each partitions beginning marks a test point. The defined sampling number therefore has a significant effect on the resulting count of test cases.

Several parallel events could occur in sequence for one scenario when suitable transition conditions are defined (e.g. failure followed by a corrective cockpit action after a defined time). However, it needs to be taken care that transitions to other phases in between these events do not happen.

A method still needs to be worked out how to consistently define different scenarios, that are triggered subsequently.

Based on the defined scenario state chart, scenario test cases can by identified by traversing along the path and taking into account the parallel events. An algorithm for automatic path identification and subsequent test case generation was presented in [6] for scenario state charts in the format that was adopted from [5]. The test cases end at final phase states

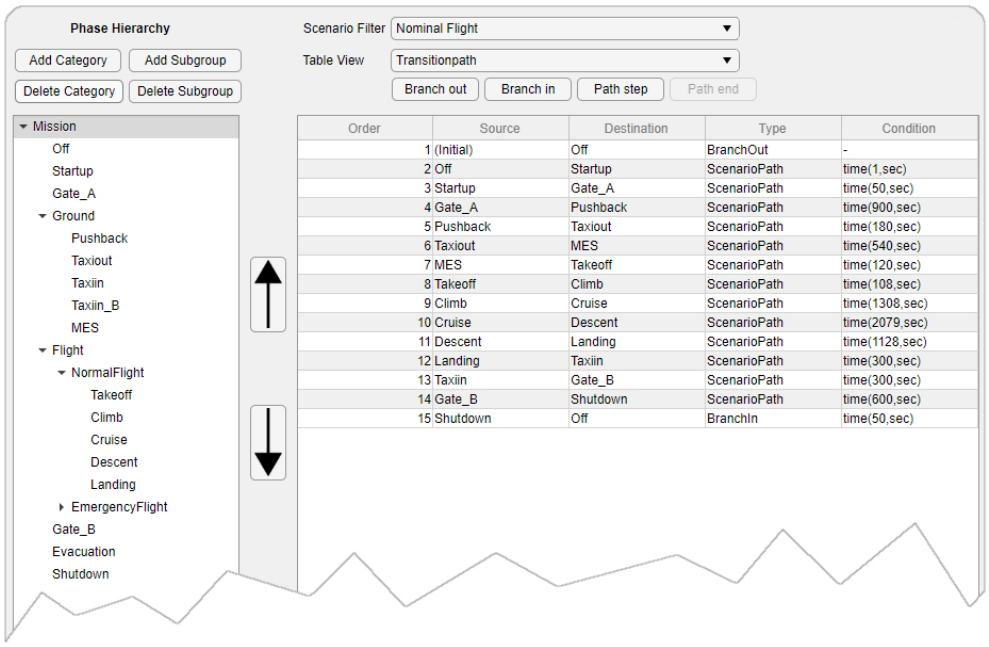


Fig. 4. Exemplary tabular transition path for a nominal flight scenario.

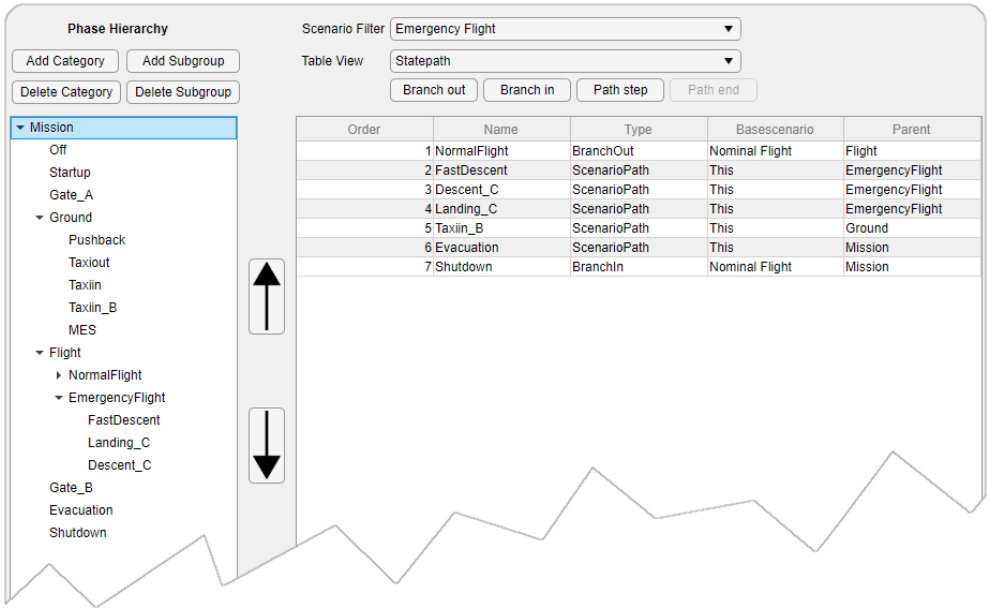


Fig. 5. Exemplary tabular state path for an emergency flight scenario.

or when the path loops. They are executable either by suitable control mechanisms to enable specific paths for the holistic state chart or by extracting unbranched scenario paths. The first option was demonstrated in [5].

### C. System and Environment Definition Layer

The activities described in the section aim to define structured objects in the environment of the system under test. This should enable a more straight forward and reusable definition of the signal profiles that are applied to the SUT interfaces for each scenario phase.

*1) System Under Test (SUT) Interface Definition:* Before defining the environmental elements, the actual interface of the system under test has to be defined, which should be used for the scenario test. This predominantly includes the declaration of the SUTs' input and output signals by defining signal name, data-type, initial value for inputs and units. Moreover, special inputs, such as fault trigger for system model components, can be defined that may be needed to reach

certain scenarios. To be able to reuse the test scenarios at different designed stages, all signals are symbolic variables and abstract from real transmission mechanisms. If more realistic transmission are needed, e.g. for hardware in the loop (HITL) integration, specific drivers need to be set up outside of the test definition process or manually integrated into the SUT model. These drivers could be defined on a target test system, such as dSPACE Scalexio, TechSAT ADS2 or Vector test hardware, which provides the hardware interface e.g. to send signals as CAN messages to real control modules in a HITL test.
The interface information could be loaded from an interface document file, that may be available from a model-based development process.

*2) Environment Component Definition:*
As a second step, the definition and categorization of environmental elements is carried out. Each previously defined SUT interface signal can subsequently be assigned to one interface element. Furthermore, it is intended to allow the definition of signal groups within environmental elements, if it is desired to define a single maneuver or action in the environment that acts on several signals. One example could be the use of two redundant sensors or protocol-based signal groups. Based on the given signal classification, parallel compound state processes for every environmental element are inserted into each of the scenario phases of the state chart structure. Similarly, single signals or signal groups form parallel processes within these compound states. In that way, sequences of actions or signal profiles could be assigned completely separate for each signal or signal group of each environmental element within each flight phase. An example is visualized on the left side of Figure 6, which includes the environmental elements *Cabin_Loads*, *Environment*, *Flight_Management* and *OVHD*, that contains the signal *Temp_Demand*.
In the case that it is desired to include single mathematical calculation models that apply continuously during the complete mission irrespective of any scenario, a parallel process to the scenario state chart is foreseen. For instance this could be models that calculate the atmospheric pressure based on a height profile that results from the scenario path. It should be mentioned that in principal, these models could also be set up outside of the state chart. The parallel calculation model process can be structured with the same environment component definition as for the phases. However, assigned signals and signal groups are then excluded from the phases. The modular setup as described prepares a structured and realistic signal assignment and facilitate re-usability through component libraries. Moreover, it creates context for reporting and analysis.

### D. Signal and Component Implementation Layer

Having defined the hierarchical structure of scenario phases and groups of signals as environmental elements in the previous steps, the concrete signal profiles are set up in this last step. For each phase state, the information gathered before includes the phase name, scenarios that include the phase, applicable parallel events and it's trigger for each scenario and the environmental component structure.
Based on this, sequences of signal functions and segments are now assigned to the signal variables as defined in the previous step. Following the objective of re-usability, a library concept for parameterizable signal functions is used. Two fundamental function types are foreseen:

- (Atomic) basic functions, that have a representation for the test system the scenario is intended to run on.
- Higher level functions, that are state chart compounds containing basic functions, states and transitions.

The atomic function library should be fixed while higher level functions for component models can be defined individually as needed. The library concept makes it easier to set up realistic scenarios by having (signal-based) interface component models with parameters such as sensors including noise, different atmosphere models etc. When assigning the library elements to the scenario the function symbols have to be mapped to the scenario signals. The sequence of library elements for each signal is defined by an unbranched transition path where a transition is taken after predefined times or arbitrary expressions. The use of common atomic functions at lowest level enables the translation of the test scenarios to other test scripting languages (e.g. SCXML [7] in its extended form or proprietary solutions). The assignment of parallel calculation models is done in the same fashion as for the phase signals. An example for the definition of sequential linked basic functions is depicted in Figure 6 for a temperature demand signal set from the flight deck during cruise phase. The functional elements, which are all set functions in the example, are defined and parameterized in a tabular form. The depicted sequence of linked states, which creates the signal shown on the right side, can then be automatically generated. To complete the signal assignment, phase end conditions need to be defined to trigger the transition to the next phase. These could be already specified textually during the scenario definition steps e.g. cruise time and emergency trigger event or can be individually defined according to the assigned profile (e.g. speed$\leq$0). For continuous signals care has to be taken, that the steadiness is fulfilled when transitioning between the phases. Useful mechanisms will still need to be worked out.

## IV. Conclusion and Future Work

In this paper, a structured procedure was presented to collectively define flight mission-based scenarios in a state-chart formalism and refine them to create realistic signals for test stimulation. The aim is the verification and validation of integrated aircraft systems in an operational context. Due to the mostly generic and modular approach, the complex scenario definition can partitioned into small manageable parts. Hence, the method would speed up the process and be reusable at different stages of the system design. The definition of system scenarios in the first step is as far as possible abstracted from the concrete system implementations and can be set up independently, early in the process. The approach of hierarchical objects for signal groupings in the further modeling achieves a

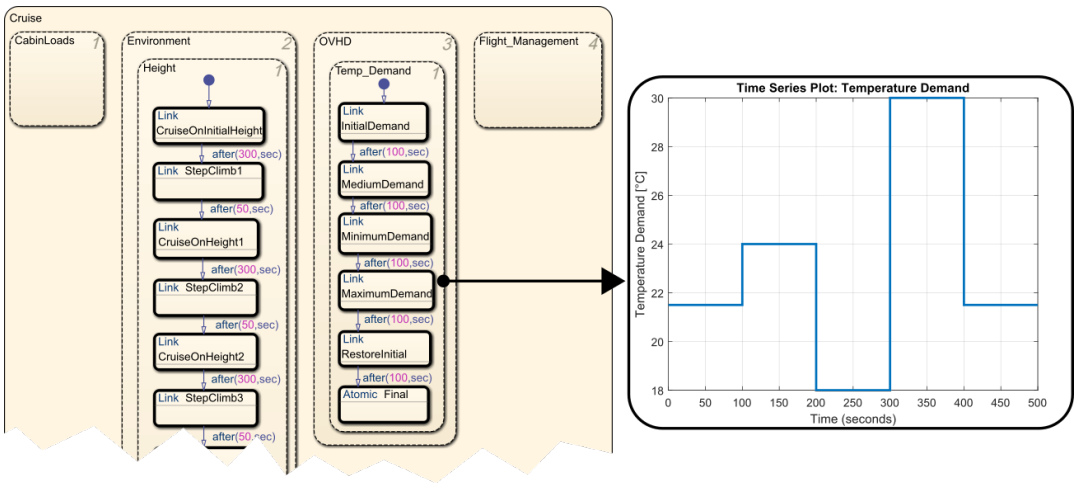


Fig. 6. Component-embedded sequence of basic functions (left) and resulting signal (right).

highly reusable structure, which could be adapted easily with regard to design changes. Finally, with the concept of basic atomic signal functions as library elements, independence from test automation systems can be reached, given that suitable conversion mechanisms exist. However, the concept is yet to be applied in a complete system test campaign.

In future works, it still needs to be investigated, if the scenario definition based on paths and parallel events is mature enough to handle very large and complex systems or if further mechanisms need to be developed. The procedure was implemented as a prototype in Matlab/Simulink using mixed hybrid graphical and structured editing, which was also presented in this paper.

Another open topic is the automatic evaluation of test runs. As pointed out in the presentation of the mission-based scenario approach, the defined scenario state-chart represents only the stimulation part for the test execution. Therefore the definition of parallel observer logics for evaluation is a major point to be addressed. The evaluations could include specified system functional or performance requirements.

Moreover, open parameter definitions shall enable the systematic exploration of specific scenarios. The transfer of resulting system states over repetitive scenario runs is also considered in that context, e.g. to assess wear effects. In that context, the remodeling of realistic scenarios from operational data and investigation of parameter variations is could be addressed. The integration of operational recordings of signals from real operations will also be worked out for this purpose. This should expand the segment-based approach that was already demonstrated in [8].

Finally, due to intend for automatic test execution the export to common test automation engines should be investigated.

## Acknowledgment

I would like to thank Martin Halle, Hauke Höber and the anonymous reviewers of the AvioSE'20 for valuable comments on the draft version. The presented results are part of the work in the research project AGILE-VT (contract code: 20X1730J), which is supported by the Federal Ministry of Economic Affairs and Energy in the national LuFo V-3 program.

## References

[1] M.Halle and F.Thielecke, "Tool Chain for Avionics Design, Development, Integration and Test", 1st Workshop on Avionics Systems and Software Engineering (AVIOSE'19), Stuttgart, Germany.

[2] ISO/IEC/IEEE International Standard 29119-4:2015 - Software and systems engineering – Software testing – Part 4: Test techniques.

[3] Everett, Gerald D.; McLeod, Raymond: "Software testing. Testing across the entire software development life cycle." Hoboken, New Jersey: IEEE Press Wiley-Interscience a John Wiley & Sons Inc. Publication, 2007.

[4] Cem Kaner (2003): "An Introduction to Scenario Testing." In: Software Testing & Quality Engineering (STQE).

[5] Dipl.-Ing. J.Grymlas (2017): "Systemautomatisierung für den multifunktionalen Betrieb von Brennstoffzellen in Verkehrsflugzeugen", PHDThesis, Hamburg University of Technology, Shaker Verlag, 2017.

[6] J. Grymlas, and F. Thielecke, "Virtual Integration and Testing of Multifunctional Fuel Cell Systems in Commercial Aircraft," SAE Int. J. Aerosp. 6(2):2013, doi:10.4271/2013-01-2281.

[7] World Wide Web Consortium (W3C). (2015) State chart XML (SCXML): State machine notation for control abstraction. [Online]. Available: https://www.w3.org/TR/scxml/.

[8] J. Grymlas, et al., "Scenario-based testing of multifunctional fuel cell systems using the virtual integration platform VIPER", in International Conference on Fundamentals and Developement of Fuel Cells (FDFC), Toulouse, 2015.

[9] Harel, D. et al. (2012): "Multi-modal scenarios revisited: A net-based representation." In: Theoretical Computer Science 429, S. 118-127.

[10] La Panzica Manna, Valerio et al. (2015): "Synthesizing tests for combinatorial coverage of modal scenario specifications.", MODELS 2015.

[11] Harel, D. and Marelly, R. (2003): "Playing with time: on the specification and execution of time-enriched LSCs.", MASCOTS 2002.

[12] Damm, W. and Harel, D. (2001): "LSCs: Breathing Life into Message Sequence Charts.", In: Formal Methods in System Design 19 (1), S. 45-80.

[13] Autili, M. et al. (2007): "Graphical scenarios for specifying temporal properties: an automated approach.", In: Autom Softw Eng 14 (3), S. 293-340.

[14] ASAM e. V. (2020): ASAM OpenSCENARIO. Online: https://www.asam.net/standards/detail/openscenario/

[15] German Aerospace Center, DLR (2020): Pegasus Research Project. Online: https://www.pegasusprojekt.de/en/home

[16] Windisch, Andreas (2011): "Suchbasierter Strukturtest für Simulink Modelle", PHDThesis, Berlin Institute of Technology, 2011.

[17] ETSI (2020): Testing and Test Control Notation Version 3 (TTCN-3). Online: http://www.ttcn-3.org/