Methodik zur modellbasierten Zuverlässigkeitsanalyse eines multifunktionalen Brennstoffzellensystems

R. Doering, C. Modest, F. Thielecke
Technische Universität Hamburg-Harburg, Institut für Flugzeug-Systemtechnik
Neßpriel 5, 21129 Hamburg, Deutschland

ZUSAMMENFASSUNG

Die vorliegende Veröffentlichung präsentiert eine Methodik zur automatischen Zuverlässigkeitsanalyse anhand des Fallbeispiels „multifunktionales Brennstoffzellensystem“. Im ersten Schritt der Methodik wird ein in MATLAB Simscape implementiertes Modell des Systems um fehlerhaftes Komponentenverhalten erweitert. Anschließend erfolgt durch einen am Institut für Flugzeug-Systemtechnik entwickelten Algorithmus die Identifizierung der Minimalschnitte des Systems bezüglich eines zuvor definierten *Top-Events*. Das Software-Tool SyRelAn erlaubt abschließend unter anderem die Darstellung der Minimalschnitte im Zuverlässigkeitsblockdiagramm. Durch die erfolgreiche Anwendung auf das Fallbeispiel können nicht nur die Performanz der Methodik demonstriert sondern darüber hinaus auch kritische Bereiche der Systemarchitektur des untersuchten multifunktionalen Brennstoffzellensystems identifiziert werden.

1. EINFÜHRUNG

Die gestiegene Komplexität und der hohe Vernetzungsgrad moderner Flugzeugsysteme stellen Ingenieure bei der Durchführung von Zuverlässigkeitsanalysen vor große Herausforderungen. Klassische Methoden wie die in der ARP 4761 [1] beschriebene Fehlerbaum- oder Zuverlässigkeitsblockdiagrammanalyse basieren dabei meist auf manueller Informationsbeschaffung und -aufbereitung und besitzen die damit verbundenen Nachteile. So ist ein solches Vorgehen fehleranfällig, zeit- und kostenintensiv sowie darüber hinaus im hohen Maße von den beteiligten Personen, ihrer Erfahrung und Qualifikation abhängig.

Trotz der genannten Problematik stellen die Ergebnisse von Zuverlässigkeitsanalysen einen zentralen Bestandteil der behördlichen Anforderungen bei der Zulassung eines neuen Flugzeugmusters dar. Weiterhin bilden sie in den Frühphasen der Systementwicklung eine wichtige Grundlage bei der Bewertung und Auswahl unterschiedlicher Systementwürfe.

Eine vielversprechende Möglichkeit die angesprochenen Nachteile zu kompensieren, besteht in der Automatisierung der Zuverlässigkeitsanalyse. Erste Ansätze dazu wurden bereits in den 70er Jahren entwickelt. Wegweisend war die Arbeit von Fussel [2], dessen vorgeschlagene Methodik in einer Software umgesetzt worden ist und es ermöglichte für einige spezielle Typen von elektrischen Systemen automatisch Fehlerbäume zu generieren.

Während mit dieser Methodik lediglich Eindomänen-Systeme geringer Komplexität untersucht werden konnten, stellte der von Lapp und Powers entwickelte Algorithmus [3] bereits eine leistungsfähigere Methode dar, welche neben den angesprochenen Systemen die Betrachtung von Prozess-Regelsystemen erlaubte.

In den folgenden Jahren wurden diese Ansätze von verschiedenen Gruppen erweitert [4] bzw. auch neue, universellere entwickelt [5]. Die angesprochenen Methoden erfordern jedoch ein relativ hohes Maß an manueller Vorarbeit, da zunächst Modelle der betrachteten Systeme erstellt werden müssen, bevor diese analysiert werden und aus den Ergebnissen ein Fehlerbaum erstellt werden kann. Der damit verbundene Arbeitsaufwand kompensiert unter Umständen die Zeitersparnis durch die anschließende automatische Durchführung der Zuverlässigkeitsanalyse.

Der aktuelle Entwicklungstrend hin zur modellbasierten Systementwicklung eröffnet die Möglichkeit diesen Nachteil zu überwinden, da die heutzutage vielfach in frühen Phasen der Systementwicklung existierenden Modelle zur Beschreibung von Systemverhalten für einen solchen Ansatz genutzt werden können.

Eine Gruppe solcher Ansätze bilden auf formalen Sprachen wie AADL [6] oder AltaRica [7] basierende Methoden. Diese ursprünglich aus der Softwareentwicklung stammenden Sprachen ermöglichen eine abstrakte Beschreibung von Systemen. Sie ermöglichen die Beschreibung von System- und Komponentenzuständen und sind in der Lage abzubilden, unter welchen Umständen sich diese ändern. Li *et al.* [8], Bozzano *et al.* [9, 10] und Bieber *et al.* [11] haben solche Methoden in teilweise umfangreichen Softwaretools umgesetzt. Mit all diesen Ansätzen ist es allerdings lediglich möglich, das statische Verhalten eines Systems in abstrakter Form abzubilden,

weshalb der erreichbare Detaillierungsgrad bzw. die Vollständigkeit der Zuverlässigkeitsanalyse auf solches Verhalten beschränkt bleibt.
Methoden, welche auf Modellierungssprachen wie Simulink, Simscape oder Modelica beruhen, sind hingegen zur Abbildung dynamischen Systemverhaltens in der Lage und ermöglichen somit auch dynamische Effekte, wie Schwingungsphänomene, mit in die Zuverlässigkeitsanalyse einzubeziehen. Joshi *et al.* [12] demonstrieren dies anhand eines dynamischen in MATLAB Simulink modellierten Bremssystems, in dessen einzelnen Komponenten fehlerhaftes Verhalten implementiert ist und mittels *switches* manuell oder skriptbasiert ausgelöst werden kann. Der Einsatz eines *Model Checkers* ermöglicht es eine Kombination von Komponentenzuständen zu identifizieren, welche zum Eintritt eines zuvor definierten Systemzustandes führt. Schallert [13] stellt in seiner Arbeit eine in Modelica implementierte Modellbibliothek vor, welche verschiedenste Komponenten des elektrischen Netzes eines Verkehrsflugzeuges enthält, in denen sowohl nominales als auch fehlerhaftes Verhalten abgebildet ist. Ausgehend von dieser Bibliothek lassen sich Modelle des elektrischen Netzes erstellen und mittels des bereitgestellten Algorithmus diejenigen Kombinationen von Komponentenfehlern identifizieren, die zum Eintritt eines Systemfehlers führen.
Alle existierenden Methoden weisen spezifische Stärken und Schwächen auf. Während die auf formalen Sprachen basierenden Methoden vergleichsweise geringe Anforderungen an die Rechenleistung stellen, dafür aber nur wenig detaillierte Ergebnisse liefern, ist die benötigte Rechenleistung für dynamische Methoden ungleich höher und kann dabei sogar zum entscheidenden Faktor für die erfolgreiche Anwendung einer Methodik werden. Abhängig vom Detaillierungsgrad des zugrundeliegenden Modells sind andererseits wesentlich detailliertere Ergebnisse erzielbar. Keine der vorgestellten Methoden erlaubt die grafische Aufbereitung der Ergebnisse in einer den klassischen Darstellungen der Zuverlässigkeitsanalyse. Dies ist aus funktionaler Sicht zwar nicht erforderlich, erleichtert dem Anwender allerdings die Interpretation der Ergebnisse.
Ziel der hier vorgestellten Methodik zur automatisierten Zuverlässigkeitsanalyse ist es, die Vorteile existierender Methoden bezüglich Rechenzeit und Detaillierungsgrad miteinander zu verbinden und darüber hinaus eine Darstellung der Ergebnisse im Zuverlässigkeitsblockdiagramm zu ermöglichen. Es wird hierbei ein generischer Ansatz verfolgt, der es ermöglicht verschiedenste physikalische Systeme zu untersuchen und dem Anwender große Freiheit bei der Wahl der Modellierungssprache sowie der gewünschten Modellierungstiefe lässt.

In Abschnitt 2 dieses Papers wird zunächst die entwickelte Methodik zur automatischen Zuverlässigkeitsanalyse vorgestellt. Anschließend erfolgt in Abschnitt 3 eine Beschreibung des im Rahmen dieses Papers untersuchten Fallbeispiels, bei welchem es sich um ein multifunktionales Brennstoffzellensystem handelt. Die Beschreibung umfasst die Erläuterung des zugrundeliegenden Modellierungsansatzes sowie der gewählten Systemarchitektur. Die Anwendung der Methodik auf das Fallbeispiel wird abschließend in Abschnitt 4 präsentiert.

2. METHODIK ZUR MODELLBASIERTEN ZUVERLÄSSIGKEITSANALYSE

Mit der entwickelten Methodik zur automatischen Zuverlässigkeitsanalyse ist es, ausgehend von einem zum Beispiel in MATLAB Simscape implementierten physikalischen Systemmodell möglich mit geringem manuellem Aufwand eine vollständige Zuverlässigkeitsanalyse durchzuführen. Die Ergebnisse können indem am Institut für Flugzeug-Systemtechnik entwickelten Tool SyRelAn [14] in Form eines Zuverlässigkeitsblockdiagramms dargestellt werden. SyRelAn ermöglicht darüber hinaus weitergehende Zuverlässigkeitsanalysen, wie zum Beispiel Markov- oder Importanz-Analysen.
BILD 1 zeigt den prinzipiellen Ablauf der Methodik, bestehend aus manuellen Vorarbeiten und dem vollständig automatisch arbeitenden **C**ut **S**et **I**dentification **A**lgorithm (CSIA) an dem anschließend die Darstellung und potentielle weiterführende Analyse der Ergebnisse erfolgt.

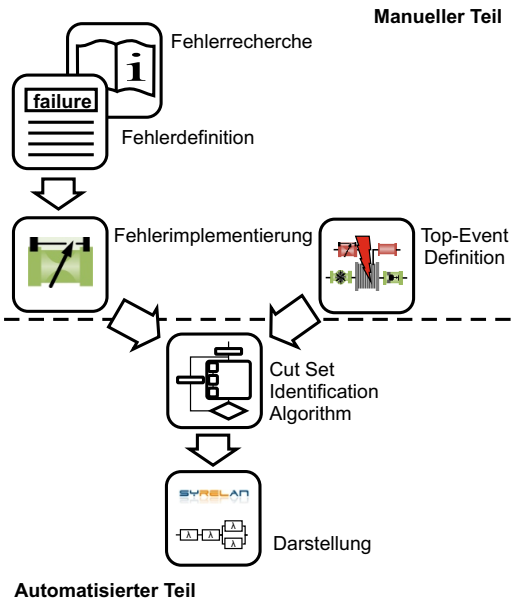

BILD 1: Prinzipieller Ablauf der Methodik zu automatischen Zuverlässigkeitsanalyse

2.1. Manuelle Vorarbeiten

Auch wenn die entwickelte Methodik weite Teile der Zuverlässigkeitsanalyse automatisiert, sind manuelle Vorarbeiten notwendig. Diese umfassen zunächst die Definition möglicher Fehler, sogenannter *failure modes*, der Komponenten aus denen das zu untersuchende System aufgebaut ist. Im Anwendungsfall kann dabei auf Herstellerinformationen zurückgegriffen werden. Liegen solche nicht vor, bieten klassische Werke der Zuverlässigkeitstechnik wie das NPRD95 [15] Anhaltspunkte. Des Weiteren ist es erforderlich jedem *failure mode* die zugehörige Fehlerrate, die *failure rate*, zuzuordnen, welche im Laufe des CSIA benötigt wird.
Im nächsten Schritt müssen die so definierten Fehler in die physikalischen Modelle der Systemkomponenten implementiert werden. Es gilt dabei das fehlerhafte Verhalten phänomenologisch, dass heißt in seinen Auswirkungen auf das Komponentenverhalten abzubilden. Es ist darauf zu achten dies so vorzunehmen, dass der jeweilige Komponentenzustand durch das Setzen von Parametern skriptbasiert steuerbar ist, wie in Abschnitt 3.3 eingehend erläutert wird.
Die beschriebenen Arbeiten sind unabhängig von der konkreten Fragestellung und somit einmalig durchzuführen. Sie fließen in die entsprechenden Modellbibliotheken ein, und können somit als Wissensspeicher für zukünftige Analysen dienen.
Neben dem um fehlerhaftes Komponentenverhalten erweiterten Systemmodell, benötigt der CSIA eine Definition der fehlerhaften Systemzustände auf deren Eintreten hin das System untersucht werden soll. Diese sogenannten *Top-Events* müssen anhand der Simulationsergebnisse skriptbasiert eindeutig identifizierbar sein.

2.2. Cut Set Identification Algorithm

Beim CSIA handelt es sich um einen Algorithmus zur automatischen Identifizierung von Minimalschnitten (*Cut Sets*) eines Systems. Ein Minimalschnitt bezeichnet eine minimale Anzahl an Komponentenfehlern, die zum Eintritt eines definierten *Top-Events* führen. Dies kann ein einzelner Fehler sein aber auch eine Kombination aus mehreren Fehlern.
BILD 2 zeigt den iterativen Charakter des CSIA, in welchem für Fehlerfälle aufsteigender Ordnung wiederholend dieselben Schritte ausgeführt werden.

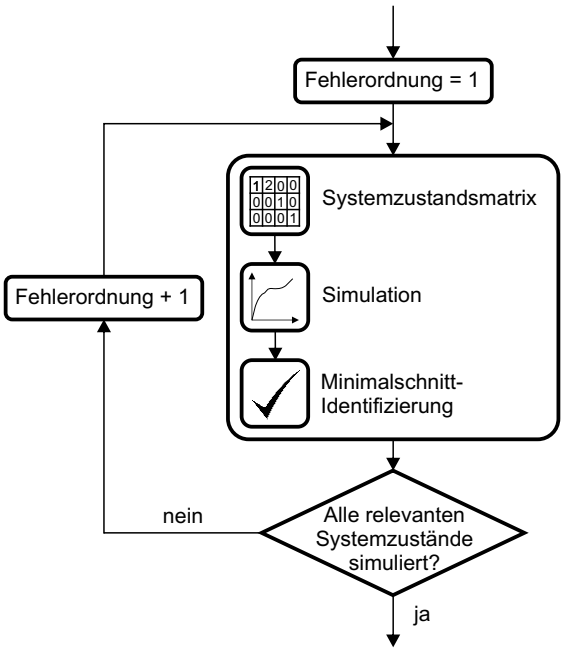

BILD 2: Ablauf des Cut Set Identification Algorithm

Beginnend mit Fehlerfällen erster Ordnung (Einfachfehler) werden zunächst alle denkbaren Zustände des zu untersuchenden Systems in einer Matrix hinterlegt. Die jeweiligen Systemzustände zeichnen sich durch eine eindeutige Kombination von Komponentenzuständen aus. Für Fehlerfälle erster Ordnung bedeutet dies, dass ein Systemzustand eindeutig durch das Eintreten eines Fehlerfalls in einer Komponente bestimmt wird, während alle übrigen Systemkomponenten nominales Verhalten zeigen. Bei Fehlerfällen zweiter Ordnung ist dementsprechend eine eindeutige Kombination zweier Komponentenfehlern erforderlich. Die Anzahl der möglichen Systemzustände ergibt sich somit aus der Anzahl der Komponenten sowie der möglichen Fehlerfälle pro Komponente.
Die so erzeugt Systemzustandsmatrix dient als Eingang für den Simulationsteil des CSIA, hier werden alle Systemzustände nacheinander im Modell skriptbasiert eingestellt, simuliert und die Ergebnisse für den nächsten Schritt gespeichert.
Anschließend wird anhand der hinterlegten Definition überprüft, welche Systemzustände zum Eintritt des *Top-Events* führen. Ein solcher Systemzustand bzw. die entsprechende Kombination aus Komponentenfehlern bilden einen Minimalschnitt des Systems.
Die durch den iterativen Teil des CSIA identifizierten Minimalschnitte werden abschließend dergestalt aufbereitet, dass sie in SyRelAn geladen werden können. Zu diesem Zweck wird mittels MATLAB-Code aus den Minimalschnitten eine SyRelAn-Inputdatei erzeugt, welche anschließend importiert wird. Diese Datei ist dabei so aufgebaut, dass eine möglichst platzsparende Darstellung der Minimalschnitte ermöglicht wird. Zu diesem Zweck findet eine Gruppierung verwandter Minimalschnitte statt, wodurch lange, kettenförmige Zuverlässigkeitsblockdiagramme vermieden werden.

2.2.1. Rechenzeit und Code-Beschleunigung

Es ist offensichtlich, dass es abhängig von der Systemgröße und der Anzahl an Fehlerfällen pro Komponente notwendig werden kann, eine Vielzahl von

Systemzuständen zu simulieren. BILD 3 zeigt exemplarisch, wie sich die Anzahl der theoretisch möglichen Systemzustände unter Annahme eines Fehlerfalles pro Komponente mittels Binomialkoeffizienten abschätzen lässt. Für Systeme mit 5, 10, 15 und 20 Komponenten ist die Anzahl der Systemzustände i dabei logarithmisch über die Fehlerordnung k aufgetragen.

Abhängig von der Komplexität des Modells kann die hohe Anzahl der Systemzustände zu Rechenzeiten führen, die den Einsatz der Methodik nicht sinnvoll erscheinen lässt.

Um dieser Problematik zu begegnen und akzeptable Rechenzeiten zu erreichen sind zwei Methoden in den CSIA implementiert, welche die Anzahl der zu simulierenden Systemzustände signifikant reduzieren. Die Tatsache, dass ein Minimalschnitt eine minimale Anzahl an Komponentenfehlern beschreibt die zum Eintritt des *Top-Events* führen, erlaubt es Systemzustände welche eine Erweiterung eines Minimalschnittes darstellen von der weiteren Betrachtung auszunehmen. Dies geschieht, indem die Systemzustandsmatrix vor der eigentlichen Simulation um die entsprechenden Systemzustände reduziert wird. Es ist offensichtlich, dass die so zu erzielende Rechenzeitersparnis umso größer ist, je geringerer Ordnung die identifizierten Minimalschnitte sind.

Bei Fehlerordnungen höherer Ordnung verspricht eine andere Methode wesentlich effektiver zu sein. Die bei der Fehlerimplementierung hinterlegten Fehlerraten jedes *failure mode* werden genutzt, um die Eintrittswahrscheinlichkeiten der Systemzustände zu berechnen. Die Tatsache, dass in der Luftfahrt sehr zuverlässige Komponenten verwendet werden hat zur Folge, dass häufig bereits Fehlerfälle dritter Ordnung eine sehr geringe Eintrittswahrscheinlichkeit haben. Liegt eine solche Wahrscheinlichkeit um eine bestimmte Anzahl von Größenordnungen unter der geforderten Eintrittswahrscheinlichkeit des untersuchten *Top-Events* können die entsprechenden Systemzustände ebenfalls von weiteren Betrachtungen ausgenommen bzw. aus der Systemzustandsmatrix entfernt werden.

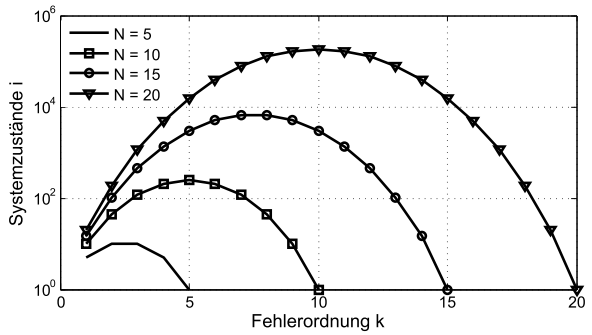

BILD 3: Systemzustände je Fehlerordnung für Systeme mit N = 5, 10, 15, 20 Komponenten

Um wie viele Größenordnungen es sich dabei handelt, wird im Einzelfall unter Berücksichtigung der Anzahl der theoretisch möglichen Systemzustände und der maximalen Eintrittswahrscheinlichkeit eines Systemzustandes der betreffenden Fehlerordnung bestimmt.

3. ERWEITERTES MODELL EINES MULTIFUNKTIONALEN BRENNSTOFFZELLENSYSTEMS

Aktuelle Forschungsaktivitäten beschäftigen sich mit der Integration von multifunktionalen Brennstoffzellensystemen in zukünftige Verkehrsflugzeuge. Diese Systeme dienen nicht nur der alleinigen Erzeugung elektrischer Energie, vielmehr sollen durch die zusätzliche Nutzung der anderen Reaktionsprodukte weitere Systeme entlastet bzw. ersetzt werden. Beispielhaft sei hier die Nutzung der sauerstoffreduzierten Abluft zur Feuerunterdrückung im Frachtraum genannt. Aufgrund der sich hieraus ergebenden hohen Systemkomplexität und dem hohen Vernetzungsgrad von multifunktionalen Brennstoffzellensystemen bieten sich diese besonders für eine automatische Zuverlässigkeitsanalyse an.

Die vorgestellte Methodik erfordert die Simulation vieler verschiedener Systemzustände, was neben der angesprochenen Rechenzeitproblematik noch eine weitere Konsequenz hat: Einige der Zustände weisen große Abweichung zum Nominalverhalten des Systems auf, was dementsprechend hohe Anforderungen an die Robustheit der Simulation stellt. Einen in diesem Zusammenhang geeigneten Ansatz verfolgt Grymlas [16] bei einem von ihm in MATLAB Simscape erstellten Brennstoffzellensystem. Hierbei handelt es sich um eine physikalische Modellierungssprache, welche auf dem „Physical Network Approach“ basiert. Bei diesem Ansatz werden die das physikalische Verhalten einer Komponente beschreibenden Gleichungen in Blöcken implementiert, welche über sogenannte Domains miteinander verbunden sind. Die Domains enthalten Potential- und Flussvariablen, über welche der Energiefluss zwischen den verschiedenen Blöcken berechnet wird.

Die besondere Stärke des umgesetzten Modellierungsansatzes liegt in der konsequenten Trennung der zugrundeliegenden physikalischen Effekte wie resistiven oder kapazitiven Verhalten. Durch die alternierende Anordnung von Elementen dieser Klassen wird ein Gleichungssystem erzeugt, welches aus nummerischer Sicht besonders gut konditioniert ist [17].

Obwohl die Verwendung der Modellierungssprache Simscape einige Vorteile bietet, ist die Methodik nicht auf selbige beschränkt, sondern erlaubt vielmehr den Einsatz beliebiger Sprachen, solange diese eine Fehlerimplementierung gemäß Abschnitt 3.3 erlauben.

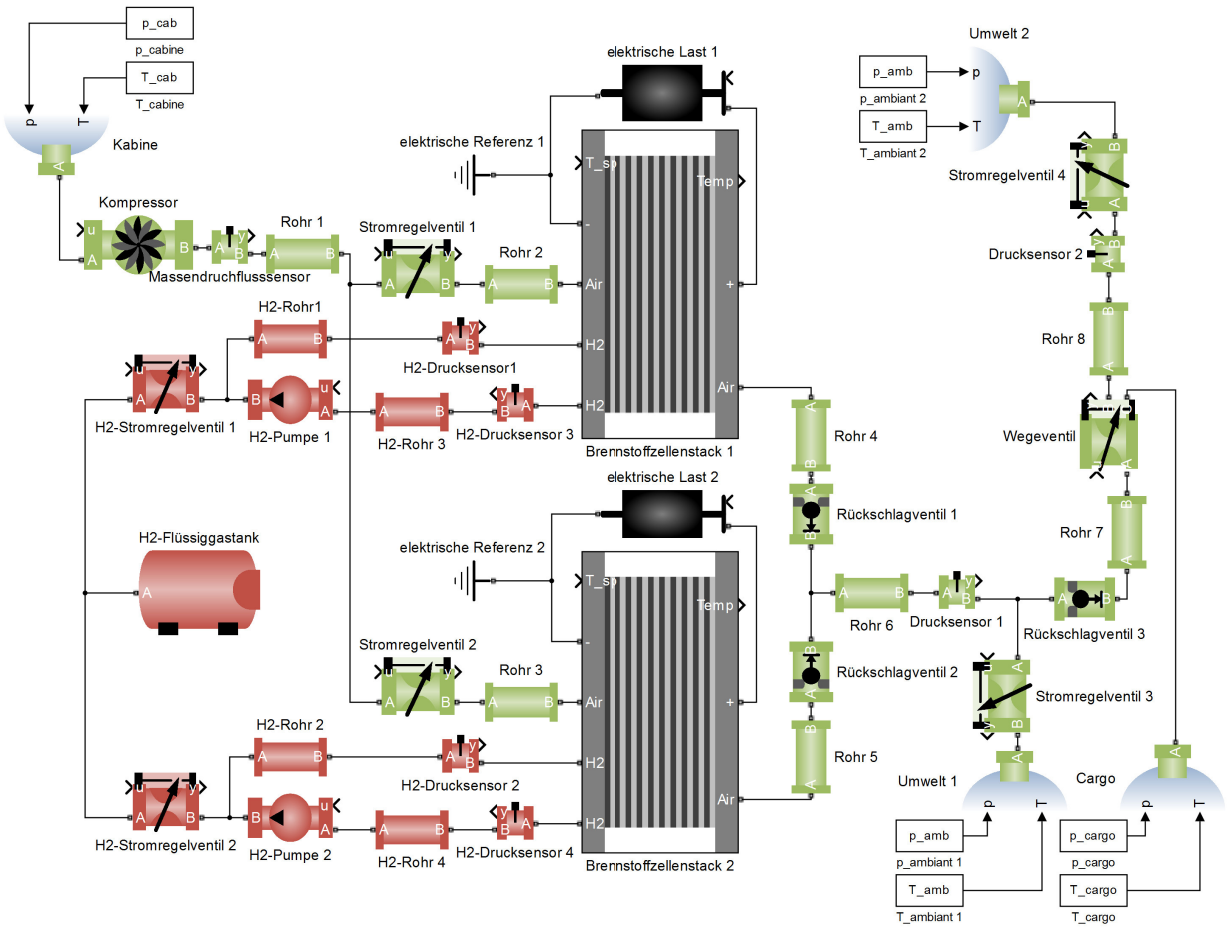

BILD 4: Systemarchitektur des betrachteten multifunktionalen Brennstoffzellensystems, bestehend aus Frischluft- (grün, links), Abluft- (grün, rechts) und Wasserstoffsystem (rot) sowie Brennstoffzellenstacks und elektrischen Lasten

3.1. Modellbibliothek

Das in diesem Beitrag untersuchte Brennstoffzellensystem basiert auf den Arbeiten in [16, 17]. Die von Ihm erstellte Modellbibliothek enthält Komponenten zur Modellierung des Luft- und des Wasserstoffsystems, welche im Wesentlichen aus Rohrleitungen, Pumpen, Kompressoren, Sensoren und Ventilen bestehen. Weiterhin beinhaltet sie einige elektrische Komponenten sowie eine in Anlehnung an Pukrushpan [18] modellierte Brennstoffzelle, in welcher durch die elektrochemische Reaktion von Wasserstoff und Sauerstoff elektrische Leistung erzeugt wird [19].

3.2. Systemarchitektur

Die Architektur des untersuchten multifunktionalen Brennstoffzellensystems ist in BILD 4 abgebildet. Es besteht aus zwei redundanten Brennstoffzellenstacks, welche durch ein Frischluft und ein Wasserstoffversorgungssystem mit den für die Leistungserzeugung notwendigen Edukten versorgt werden. Die Frischluftversorgung wird durch einen Kompressor sichergestellt, welcher Luft aus der Kabine ansaugt, verdichtet und über Rohrleitungen zu den Brennstoffzellenstacks leitet. Durch jeweils ein Stromregelventil unmittelbar vor den Stacks, kann die Frischluftversorgung der Stacks exakt geregelt werden. Darüber hinaus befinden sich die zur Regelung der Frischluftversorgung erforderliche Sensoren im System.

Als Wasserstoffquelle dient ein Flüssigwasserstofftank, dessen Wasserstoffabgabe ebenfalls durch Stromregelventile kontrolliert wird. Rohleitungselemente dienen dem Transport des Wasserstoffes zu den Stacks, eine Pumpe am Wasserstoffauslass eines jeden Stacks dient der Rezirkulation des unverbrauchten Wasserstoffes, Sensoren erheben die zur Regelung erforderlichen Messgrößen.

Die in den Brennstoffzellen produzierte sauerstoffreduzierte Abluft kann mittels des Wegeventils entweder zur Inertisierung des Cargo Compartements genutzt oder über Bord abgeleitet werden. Rückschlagventile unterbinden eine mögliche Umkehr des Massenflusses in Richtung der Brennstoffzellen.

Zugunsten einer verringerten Systemkomplexität, ist die Abfuhr der ebenfalls produzierten Abwärme ist nicht modelliert. Es wird vielmehr eine konstante Temperatur der Brennstoffzellen angenommen.

Die Regelung des Systems erfordert die Vorgabe einer benötigten Leistung, was durch eine variable

Last an den elektrischen Ausgängen der beiden Brennstoffzellen realisiert wird. Die implementierten Regler sorgen dafür, dass sich entsprechend geforderten Last die notwendigen Eduktströme einstellen. Insgesamt besteht das multifunktionale Brennstoffzellensystem aus den in TAB 1 aufgeführten 37 Komponenten.

Komponente	*Anzahl*	*failure modes*
Rohrelement	12	2
Sensor	7	3
Stromregelventil	6	4
Rückschlagventil	3	4
Pumpe	2	4
Brennstoffzellenstack	2	5
Elektrische Last	2	-
Kompressor	1	4
Wasserstofftank	1	-
Wegeventil	1	2

TAB 1: Komponenten des multifunktionalen Brennstoffzellensystems inklusive Anzahl der *failure modes*

3.3. Fehlerimplementierung

Die im Rahmen der Methodik durchzuführende Fehlerimplementierung soll anhand eines Stromregelventils exemplarisch demonstriert werden. TAB 2 zeigt die Ergebnisse der Fehlerrecherche und Fehlerdefinition. Basierend auf Daten aus dem NPRD95 und dem FMD97 [20] lassen sich vier verschiedene *failure modes* mit den zugehörigen Eintrittswahrscheinlichkeiten R_h pro Flugstunde identifizieren. Neben der internen Leckage bestehen diese aus Klemmfällen in drei verschiedenen Positionen: Geschlossen, geöffnet und halb geöffnet.

Failure Mode	*Fehlerart*	R_h *[1/FH]*
1	interne Leckage	9e-6
2	klemmt halb geöffnet	2e-6
3	klemmt geschlossen	2e-6
4	klemmt geöffnet	1e-6

TAB 2: *failure modes* und entsprechende Fehlerraten eines Stromregelventils

Die Implementierung des fehlerhaften Verhaltens geschieht direkt im Quellcode des Stromregelventils. Der Massendurchfluss $\dot{m}$ durch das Ventil wird dort durch Gleichung 1 bestimmt:

$$\dot{m} = K_v \cdot \sqrt{(\Delta p * p_{out} * \rho / T)} \quad (1)$$

Er hängt neben der Dichte ρ, der Temperatur T, dem Ausgangsdruck p_{out} und der anliegenden Druckdifferenz Δp maßgeblich vom Durchflusskoeffizienten K_v ab. Im Nominalfall wird über eine Regelstrecke der gewünschte Massenstrom durch die Wahl des entsprechenden Durchflusskoeffizienten eingestellt. Es liegt nahe, die definierten *failure modes* durch die Manipulation des Durchflusskoeffizienten abzubilden. So wird z.B. für den *failure mode* „klemmt geschlossen“ der Durchflusskoeffizient gleich null gesetzt wird. Mit den anderen *failure modes* wird analog verfahren.
Damit die einzelnen *failure modes* durch den CSIA aber auch manuell ausgelöst werden können, erfolgt die Implementierung der vier *failure modes* entsprechend eines universellen Vorgehens. Hierzu wird ein Parameter „failure_mode“ in den Quellcode des Ventils implementiert, dessen jeweiliger Wert den entsprechenden *failure mode* auslöst. Durch eine im Quellcode der Komponenten und in Code 1 dargestellte *if-else*-Abfrage werden die entsprechenden Gleichungen zur Beschreibung des Komponentenverhaltens aktiviert, im Falle des Stromregelventils also dem Durchflusskoeffizienten der entsprechende Wert zugewiesen.

```
if failure_mode == 1
        K_v =  0,05 * K_v_max
elseif failure_mode == 2
        K_v = 0,5 * K_v_max
elseif failure_mode == 3
        K_v = 0 * K_v_max
elseif failure_mode == 4
        K_v = 1 * K_v_max
else
        K_v = K_v_nominal
end
```

Code 1: *if-else*-Abfrage zur Zuweisung des Durchflusskoeffizienten in Abhängigkeit des *failure modes*

4. DRUCHFÜHRUNG EINER ZUVERLÄSSIGKEITSANALYSE

Die vorgestellte Methode zur automatischen Zuverlässigkeitsanalyse soll nun auf das beschriebene multifunktionale Brennstoffzellensystemmodell angewandt werden. Die erforderliche manuelle Fehlerimplementierung für die einzelnen Komponenten des Systems erfolgt In analoger Weise zu dem soeben beschriebenen Vorgehen. In den meisten Fällen kann die phänomenologische Modellierung von fehlerhaften Komponentenverhalten durch die Manipulation von Wirkungsgeraden oder anderen Koeffizienten, ähnlich dem Durchflusskoeffizienten des Stromregelventils, erreicht werden. Insgesamt ist es erforderlich in 34 Komponenten zusammen 105 Fehlerfälle zu implementiert. TAB 1 listet die Anzahl der *failure modes* je Komponente auf.
Exemplarisch soll das multifunktionale Brennstoffzellensystem auf Eintreten eines *Top-Event* hoher Kritikalität hin untersucht werden. Der komplette Verlust der elektrischen Leistungsversorgung stellt so einen Fehlerfall dar und soll daher als beispielhaftes *Top-*

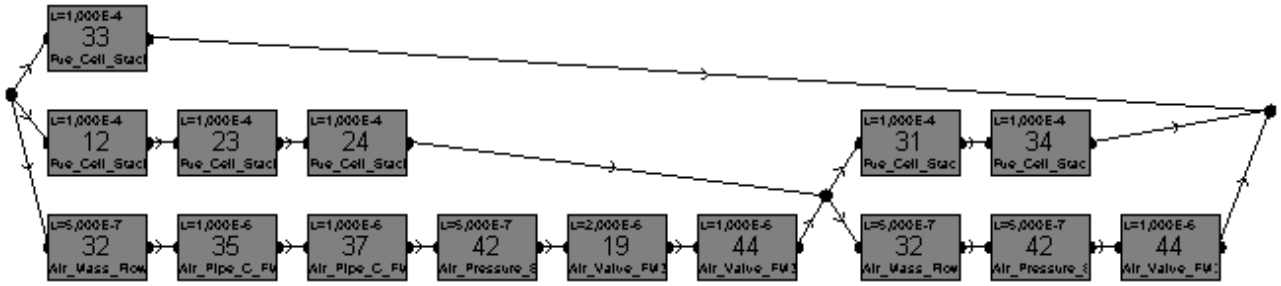

BILD 5: Ausschnitt aus dem automatisch erzeugten Zuverlässigkeitsblockdiagramm des Brennstoffzellensystems für das *Top-Event* „keine Leistungsabgabe“

Event „keine Leistungsabgabe“ dienen. Der erforderliche Identifikations-Code basiert auf dem Monitoring der beiden elektrischen Lasten des Systems. Mittels einer *if-else*-Abfrage wird überprüft, ob in den Lasten elektrische Leistung umgesetzt wird. Dies ist im Nominal- und degradiertem Zustand der Fall, während im Fehlerfall die Lasten von den Brennstoffzellen getrennt werden.
Zusammen mit dem um fehlerhaftes Verhalten erweiterten Systemmodell dient dieser Identifikations-Code als Input für den CSIA, welcher die zum Top-Event gehörenden Minimalschnitte identifiziert und selbige mithilfe von SyRelAn darstellt.

4.1. Interpretation der Ergebnisse

TAB 3 zeigt die Anzahl der durch den CSIA identifizierten Minimalschnitte des Systems bezüglich des *Top-Events* „keine Leistungsabgabe“.

Fehlerordnung	*Anzahl Minimalschnitte*
1	4
2	84
3	71
gesamt	159

TAB 3: Minimalschnitte je Fehlerordnung

Ohne jeden der Minimalschnitte einzeln analysieren zu wollen, soll eine Übersicht über kritische Komponenten und typische Fehlerkombinationen gegeben werden.
Bei den vier Minimalschnitten erster Ordnung handelt es sich jeweils um Fehler, welche zum Einbruch der luftseitigen Druckdifferenz über den Brennstoffzellen führt. Bei den Fehlerfällen, die einen derartigen Zustand zu Folge haben, handelt es sich entweder um eine große Leckage in einem der drei Rohrelemente des Frischluftversorgungssystems oder um das Blockieren und damit den Ausfall des Kompressors.
Die Minimalschnitte zweiter Ordnung lassen sich in zwei Kategorien einteilen. Zum einen in Kombinationen aus Einzelfehlern, welche zum Ausfall je eines Brennstoffzellenstacks führen und zum anderen in Kombinationen von Komponentenfehlern, bei denen dies nicht der Fall ist. Zu den Komponentenfehlern der ersten Gruppe gehören klemmende Stromregelventile im Frischluft- oder Wasserstoffversorgungssystem aber auch mögliche alterungsbedinge Effektivitätsverluste der Brennstoffzellenstacks. Die zweite Gruppe bilden Kombinationen aus einigen Komponentenfehlern derjenigen Komponenten, welche sich im nicht-redundanten Teil des Frischluft- bzw. Abluftsystems befinden. Exemplarisch sei hier der Ausfall des Massendurchflusssensors hinter dem Kompressor, in Kombination mit einem Klemmen des Rückschlagventils im Abluftsystem genannt.
Die Gesamteintrittswahrscheinlichkeit des *Top-Events* liegt bei $F_S = 2,1388 \cdot 10^{-6} \cdot t$. Wie zu erwarten war, dominiert der Ausfall des sich im nicht-redundanten Teil des Systems befindliche Kompressor diese Eintrittswahrscheinlichkeit. Zwar führen auch die angesprochenen drei weiteren Einzelfehler zum Eintritt des *Top-Events*, jedoch weisen diese eine deutlich geringere Fehlerrate auf als der *failure mode* „Blockieren des Kompressors“ ($\lambda = 2 \cdot 10^{-6}$). Gleiches gilt für die Fehlerfälle höherer Ordnung, deren Fehlerraten um Größenordnungen niedriger liegen.
Gemäß der in der Luftfahrt gängigen Klassifizierung von Systemausfallwahrscheinlichkeiten [21], darf mit der gewählten Systemarchitektur das untersuchte *Top-Event* auf Flugzeugebene lediglich Ausfallfolgen der Kategorie „*Major*“ haben. Ist eine höhere Klassifizierung gewünscht, muss die Systemarchitektur, z.B. durch Verwendung eines redundant angeordneten Kompressors, entsprechend angepasst werden.

4.2. Darstellung der Ergebnisse

Aufgrund der großen Anzahl an identifizierten Minimalschnitten, ist lediglich ein Teil des erzeugten Zuverlässigkeitsblockdiagramms in BILD 5 dargestellt. Jeder Block repräsentiert einen Komponentenfehler. Parallel angeordnete Blöcke bilden einen Minimalschnitt. Wie zu sehen ist, erzeugt die vorgestellte Methodik eine sehr kompakte Darstellung der Minimalschnitte, welche es ermöglicht in dem vergleichsweise

kleinen Ausschnitt insgesamt 24 Minimalschnitte dritter Ordnung darzustellen.

4.3. Bewertung der implementierten Beschleunigungsmaßnahmen

Der Vergleich der nicht reduzierten mit der reduzierten Systemzustandsmatrix einer jeden Fehlerordnung erlaubt eine Bewertung der CSIA-Beschleunigungsmaßnahmen anhand des Beispiels des multifunktionalen Brennstoffzellensystems durchzuführen. TAB 4 zeigt, wie sich die Anzahl der zu simulierenden Systemzustände für das *Top-Event* „keine Leistungsabgabe“ unter Anwendung der beiden implementierten Beschleunigungsmaßnahmen reduziert. Es zeigt sich, dass insbesondere die Reduzierung der Systemzustände auf Grundlage der Eintrittswahrscheinlichkeiten (ΔN_{R_h}) der einzelnen Fehler sehr effektiv ist. Bereits bei Fehlerfällen dritter Ordnung können auf diese Weise 97.6% der Systemzustände vernachlässigt werden. Trotz dieser enormen Reduktion bleibt aufgrund der hohen Zahl an möglichen Systemzuständen noch immer eine signifikante Anzahl an relevanten Systemzuständen bestehen. Ein weiterer Vorteil dieses Ansatzes besteht in seiner Unabhängigkeit von Systemarchitektur und untersuchtem *Top-Event*, da für seine Anwendung lediglich auf die Eintrittswahrscheinlichkeiten der einzelnen Fehlermodi zurückgegriffen werden muss. Diese Tatsache ermöglicht es, die zu erwartende Rechenzeitersparnis vor der eigentlichen Simulation zu berechnen.

Fehler-ordnung	N_0	ΔN_{MS}	ΔN_{R_h}	ΔN_{ges}
1	105	0	0	0
2	5334	-404	-177	-524
3	174632	-27862	-170442	-171848

TAB 4: Systemzustände je Fehlerordnung mit und ohne Reduzierung

Ein solches Vorgehen ist bei der Reduktion basierend auf den identifizierten Minimalschnitten (ΔN_{MS}) nicht möglich. Auch zeigt sich, dass die für das definierte *Top-Event* erreichbare Reduktion an Systemzuständen mit 7,6% für Fehlerfälle zweiter Ordnung bzw. 15,5% für Fehlerfälle dritter Ordnung deutlich geringer ausfällt, was allerdings auch in der Art des *Top-Events* begründet liegt. Aufgrund der Kritikalität des betrachteten *Top-Events* führen nur sehr wenige Systemzustände zu dessen Eintritt, entsprechend gering fällt die Anzahl an Erweiterungen dieser Systemzustände aus. Es ist zu erwarten, dass dieser Ansatz bei weniger kritischen Systemfehlern, wie zum Beispiel „Reduktion der Leistung um 10%“ deutlich effektiver arbeitet, da sich bereits deutlich mehr Fehlerfälle erster Ordnung identifizieren lassen, welche zum Eintritt eines solchen führen.
Die kombinierte Anwendung beider Beschleunigungsmaßnahmen ermöglicht die Einsparung von 9,8% bzw. 98,4% der Systemzustände für Fehlerfälle zweiter bzw. dritter Ordnung. Das diese nicht die Summe der Ersparnisse beider Maßnahmen entspricht, liegt in der Tatsache begründet, dass einige Systemzustände durch beide Maßnahmen als vernachlässigbar identifiziert werden.

5. FAZIT

Die vorgestellte Methodik zu automatisierten Zuverlässigkeitsanalyse ermöglicht es ausgehend von einem bestehenden physikalischen Systemmodell eine zu weiten Teilen automatisierte Zuverlässigkeitsanalyse durchzuführen. Die dennoch notwendige manuelle Fehlerimplementierung in die Komponenten des Systems ist einmalig durchzuführen und steht so auch für spätere Analysen zur Verfügung. Der entsprechende Zeitanteil sinkt somit mit dem Umfang der untersuchten Systemfehler.
Bei der Wahl der zugrundeliegenden Modellierungssprache, -tiefe sowie Art der analysierbaren Systeme zeigt die Methodik große Flexibilität. So ist es sowohl möglich einfache statische als auch komplexe, dynamische Systemmodell zu untersuchen. Darüber hinaus ist sie nicht auf die im untersuchten Fallbeispiel angewendete Modellierungssprach MATLAB Simscape beschränkt sondern kann bei korrekter Umsetzung der Fehlerimplementierung auch in Verbindung mit anderen Modellierungssprachen genutzt werden. Weiterhin gibt es keine Beschränkung auf eine bestimmte Art von Systemen, die Methodik ist prinzipiell auf jedes physikalische multidomänen System anwendbar.
Anders als beim manuell durchgeführten Zuverlässigkeitsanalysen, ist die Vollständigkeit der Ergebnisse bezüglich des zugrundeliegenden Systemmodells sichergestellt. Es bleibt allerdings zu beachten, dass ein solches Modell immer nur mit Einschränkungen in der Lage ist die Realität abzubilden.
Einen weiteren Vorteil der Methodik stellt die Möglichkeit der Darstellung der Ergebnisse dar. Obwohl aus technischer Sicht nicht erforderliche, erleichtert sie dem Anwender die Interpretation.
Kritisch zu betrachten ist die Komplexität der zu analysierenden Modelle und damit die erforderlichen Rechenzeiten. Zwar konnte durch die umgesetzten Beschleunigungsmaßnahmen im Falle des analysierten Fallbeispiels die Zahl der zu untersuchenden Systemzustände auf ein akzeptables Maß gesenkt werden, dies muss allerdings nicht notwendigerweise auch für Systeme größerer Komplexität gelten.
Trotz dieser Unsicherheiten, konnte gezeigt werden, dass die entwickelte Methodik ein großer Schritt hin

zu einer vollwertigen automatischen Zuverlässigkeitsanalyse darstellt.

6. Danksagung

Die vorgestellte Methodik ist im Rahmen einer Masterarbeit am Institut für Flugzeug-Systemtechnik der Technischen Universität Hamburg-Harburg in Zusammenarbeit mit SAREL Consult entstanden. Der Autor möchte sich bei den beteiligten Personen für die freundliche Unterstützung und Begleitung bedanken.

LITERATUR

[1] SOCIETY OF AUTOMOTIVE ENGINEERS INC.: *Certification Considerations for Highly-Integrated or Complex Aircraft Systems, ARP 4754.* 1996

[2] FUSSELL, J.B. ; VESELY, W.E.: A new methodology for obtaining cut sets for fault-trees. In: *Transactions of the American Nuclear Society* 15 (1972), S. 262–263

[3] LAPP, S.A. ; POWERS, G.J.: Computer-aided Synthesis of Fault-trees. In: *IEEE Transactions of Reliability* (1977)

[4] VRIES, R.C. D.: An Automated Methodology for Generating a Fault Tree. In: *IEEE Transactions on Reliability* 39 (1990)

[5] SALEM, S.L. ; APOSTOLAKIS, G.E. ; OKRENT, D.: A computer-oriented approach to fault-tree construction / University of California, Los Angeles. 1976. – Forschungsbericht

[6] SOCIETY OF AUTOMOTIVE ENGINEERS INC.: *Architecture Analysis and Design Language (AADL)*. 2004

[7] GRIFFAULT, A. ; POINT, G. ; RAUZY, A. ; SIGNORET, J.P. ; THOMAS, P.: The AltaRica Language. In: *European Safety and Reliability Conference*, 1998

[8] LI, Y. ; ZHU, Y. ; MA, C. ; XU, M.: A Method for Constructing Fault Trees from AADL Models. In: *Lecture Notes in Computer Science* 6906 (2011), S. 243–258

[9] BOZZANO, M. ; CIMATTI, A. ; KATOEN, J.-P. ; NGUYEN, V.Y. ; NOLL, T. ; ROVERI, M. ; WIMMER, R.: A Model Checker for AADL. In: *Lecture Notes in Computer Science* 6174 (2010), S. 562–565

[10] BOZZANO, M. ; CIMATTI, A. ; KATOEN, J.-P. ; NGUYEN, V.Y. ; NOLL, T. ; ROVERI, M.: Safety, Dependability and Performance Analysis of Extended AADL Models. In: *The Computer Journal* 24 (2011)

[11] BIEBER, P. ; CASTEL, C. ; SEGUIN, C.: Combination of Fault Tree Analysis and Model Checking for Safety Assessment of Complex System. In: *Lecture Notes in Computer Science* 2485 (2002), S. 19–31

[12] JOSHI, A. ; HEIMDAHL, M. ; MILLER, S. ; WHALEN, M.: Model-Based Safety Analysis / University of Minnesota, Rockwell Collins, Inc. 2006. – Forschungsbericht

[13] SCHALLERT, C.: Ein integriertes Entwurfswerkzeug für elektrische Bordsysteme. In: *Deutscher Luft- und Raumfahrtkongress*, 2010

[14] VAHL, A.: *Interaktive Zuverlässigkeitsanalyse von Flugzeug-Systemarchitekturen*, Technische Universität Hamburg-Harburg, Diss., 1998

[15] DENSON, W. ; CHANDLER, G. ; CROWELL, W. ; CLARK, A. ; JAWORSKI, P.: *Nonelectronic Parts Reliability Data 1995*. Reliability Analysis Center, 1995

[16] GRYMLAS, J. ; THIELECKE, F.: Virtual Integration and Testing of Multifunctional Fuel Cell Systems in Commercial Aircraft. In: *SAE Int. J. Aerosp.* 6(2) (2013)

[17] GRYMLAS, J. ; LÜDDERS, H.P. ; STRUMMEL, H. ; THIELECKE, F.: Modellbasierter Entwurfsprozess für Brennstoffzellensysteme unter Verwendung eines mehrstufigen Bibliothekskonzepts. In: *Deutscher Luft- und Raumfahrtkongress*, 2012

[18] PUKRUSHPAN, J.T.: *Modeling and Control of Fuel Cell Systems and Fuel Processors*, University of Michigan, Diss., 2003

[19] O'HAYRE, R. ; CHA, S.-W. ; COLELLA, W. ; PRINZ, F.B.: *Fuel Cell Fundamentals*. Wiley, 2009

[20] DENSON, W. ; CROWELL, W. ; JAWORSKI, P. ; MAHAR, D.: *Failure Mode/Mechanism Distribution*. Reliability Analysis Center, 1997

[21] EUROPEAN AVIATION SAFTY AGENCY: Certification Specification for large Aeroplanes, Airworthiness Code, CS 25.1309. 2010. – Forschungsbericht