REDUNDANZMANAGEMENT FEHLERTOLERANTER FLUGZEUG–SYSTEMARCHITEKTUREN – ZUVERLÄSSIGKEITSTECHNISCHE SYNTHESE UND ANALYSE DEGRADIERTER SYSTEMZUSTÄNDE –

D. Rehage, ***U. B. Carl***, ***A. Vahl***
Technische Universität Hamburg–Harburg
Arbeitsbereich Flugzeug–Systemtechnik, D-21071 Hamburg

ÜBERSICHT

Der Entwurfsprozeß von Flugzeug–Systemarchitekturen moderner Verkehrsflugzeuge wird immer komplexer. Bedingt ist dies durch die wachsenden Anforderungen an Sicherheit und Zuverlässigkeit bei gleichzeitig zunehmenden Funktions– und Leistungsanforderungen. Da diese Maßgaben in einem Spannungsfeld zueinander stehen, sowie der Entwicklungszeitraum an einen engen Zeit– und Kostenrahmen gebunden ist, sind schon in der Frühphase der Systementwicklung rechnergestützte Methoden und Verfahren erforderlich.

Dieser Artikel zeigt den Entwicklungsstand eines Software–Tools, welches den Ingenieuren unterschiedlichster fachlicher Ausrichtungen eine Plattform für die Synthese und Analyse komplexer und eingebetter Hardware– und Software–Systeme bietet.

Basis hierzu ist ein hybrides Systemmodell mit einer strukturorientierten Modellierung der Architektur in Zuverlässigkeitsblockdiagrammen sowie einem neuartigen Konzept der zustandsorientierten Modellierung abgebildeter Komponenten mit hierarchischen, nebenläufigen endlichen Automaten. Zur Analyse von fehlerfreien und degradierten Systemzuständen werden auf das Zuverlässigkeitsblockdiagramm Algorithmen der Zuverlässigkeitsanalyse angewandt. Im Bereich des Redundanzmanagements zur Visualisierung und Analyse von Zustandsübergangsprozessen bei Fehlerinjektion, Fehlerausbreitung und Rekonfiguration sind dies Algorithmen der *Künstlichen Intelligenz.*

SCHLAGWORTE

Endlicher Automat; degradiertes System; Fehlertoleranz; Redundanzmanagement; Systementwurf; Zuverlässigkeitsanalyse; Zuverlässigkeitsblockdiagramm

1 EINLEITUNG

Die Entwicklung von Flugzeug-Systemarchitekturen führt gerade durch die hohen Sicherheits– und Zuverlässigkeitsanforderungen, in Verbindung mit hohen Verfügbarkeiten zur Senkung der Betriebskosten, auf eine fehlertolerante Architektur. Hierbei handelt es sich um eine grundlegende Eigenschaft von Flugzeugsystemen zu deren Voraussetzung Redundanz gehört. Der Entwurf solcher Systeme ist multidisziplinär und erstreckt sich über die Bereiche von Rechnersystemen, Kommunikationssystemen und peripheren Systemprozessen (z.B. Stellsysteme und Sensorik). Rechnerunterstützung ist in diesem Rahmen notwendig, um die Entwurfskomplexität der Systeme besser zu bewältigen und auch bessere Entwürfe in kürzerer Zeit zu ermöglichen, damit die nachfolgend aufgeführten Fragestellungen leichter und auf Basis eines Software–Tools zu beantworten sind:

- *„Erfüllt die Fluzeug–Systemarchitektur, aufgrund der Redunzanordnung die Zuverlässigkeitsanforderung im fehlerfreien Nominalzustand?“*,
- *„Wie rekonfiguriert das Flugzeugsystem unter Fehlereinflüssen?“*,
- *„Ist der Redundanzgrad der Fluzeug–Systemarchitektur hinreichend, um die Zuverlässigkeitsanforderungen in degradierten Systemzuständen zu gewährleisten?“*

Schon in der Phase der konzeptionellen Architekturfindung, wird den Systemingenieuren mit SYRELAN™ (**SY**STEM **REL**IABILITY **AN**ALYSIS) eine rechnergestützte Entwurfsumgebung zur Verfügung gestellt, mit der Architekturen zuverlässigkeitstechnisch synthetisiert und analysiert werden können. Im Rahmen dessen erfolgt zunächst die zuverlässigkeitstechnische Modellierung der fehlertoleranten Architekturen durch logische Verknüpfung in Zuverlässigkeitsblockdiagrammen (RBD, *engl.* **R**eliability **B**lock **D**iagram) in *positiver* Logik. Neben der Modellierung in RBDs und der Analyse der Ausfallwahrscheinlichkeit aus dem fehlerfreien Nominalzustand, basierend auf Exponentialverteilungen der Komponentenausfallraten, wird das Systemverhalten unter Einfluß von Fehlerszenarien im RBD–Modell abgebildet und analysiert. Dazu werden den RBD–Blöcken hierarchische, nebenläufige endliche Automaten (HCFSM, *engl.* **H**ierarchical **C**oncurrent **F**inite **S**tate **M**achine) hinterlegt,

welche die Arbeitsweise des Redundanzmanagements beschreiben und dessen visuelle Abbildung im RBD unterstützen.

2 HIERARCHISCHES MODELL

Die Modellierung der Flugzeug–Systemarchitekturen unterteilt sich in zwei hierarchische Entwurfsumgebungen. Zunächst wird in der oberen Modellebene, der RBD–Umgebung, die fehlerfreie Systemarchitektur durch logische Verknüpfung von RBD–Blöcken abgebildet. Dem schließt sich die Modellierung des Redundanzmangements in der unteren Modellierungsebene durch Definition der im Hintergrund betriebenen HCFSM–Modelle an. Gekoppelt sind die beiden Modelle aus Sicht der RBDs durch Zuweisung von Komponenzuständen an die darüberliegenden RBD–Blöcke (Bild 1). Bei den Systemzuständen werden nur diejenigen angesprochen, die aus zuverlässigkeitstechnischer Sicht relevante Degradationsstufen repräsentieren [7]. Die Zuweisung entsprechender Farben zu solchen Komponentenzuständen zeigt die nachfolgende Auflistung:

„aktiv“ ← **GRÜN**: Die Arbeitskomponente A ist von Missionsbeginn an der vollen Belastung ausgesetzt. Die Ausfallrate ist λ_A.

„aktiv–heiss“ ← **GELB**: Reserveelement H ist von Missionsbeginn an der gleichen Belastung ausgesetzt wie die eigentliche Arbeitskomponente A. Für die Ausfallrate gilt $\lambda_H = \lambda_A$.

„passiv–warm“ ← **ORANGE**: Reserveelement W ist bis zum Ausfall der Arbeitskomponente A (oder bis zum eigenen vorzeitigen Ausfall) einer geringeren Belastung ausgesetzt. Für die Ausfallrate gilt $0 < \lambda_W < \lambda_A$.

„passiv–kalt“ ← **HELLBLAU**: Reserveelement K ist bis zum Ausfall der Arbeitskomponente A keiner Belastung ausgesetzt. Für die Ausfallrate gilt $\lambda_K = 0$.

Aus Sicht der HCFSM–Modelle besteht die Kopplung durch Fehlerinjektion in den RBD–Block, da zu dem Komponentenausfall im RBD Zustandsübergänge im HCFSM–Modell hervorgerufen werden.

2.1 RBD–Modell

Ausgangspunkt der Modellierung von Systemarchitekturen bildet die strukturelle Abbildung des fehlertoleranten Systems in RBDs. Dazu muß in einem ersten Schritt ein TOP–EVENT definiert werden, welches den zu analysierenden Ausfallzustand beschreibt. Dieser Zustand repräsentiert entweder das fehlerfreie oder das degradierte System in Abhängigkeit der Komponentenzustände. Mathematisch wird dies durch die BOOLSCHE–Algebra beschrieben, die auf einer *zweiwertigen* Logik basiert [6]. Für die Darstellung der einzelnen Komponenten wird die *stochastisch unabhängige* Indikatorvariable K_i eingeführt, für die gilt [6]

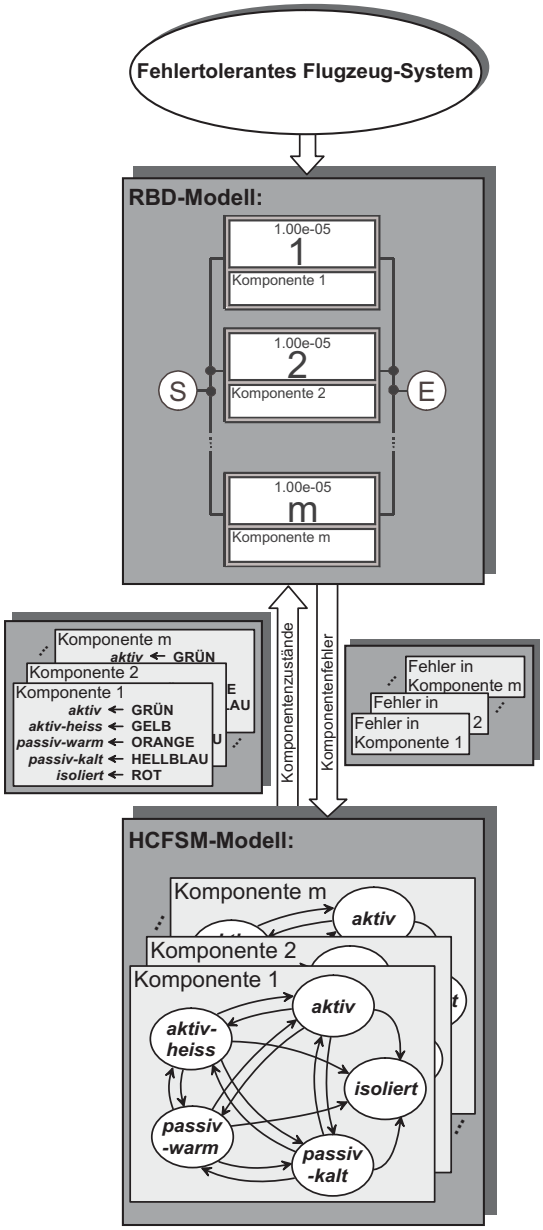


BILD 1: Kopplung der hierarchischen Modellierungsumgebungen von RBD und HCFSM

(1) $K_i = 1$ Komponente K_i ist funktionsfähig

und

(2) $K_i = 0$ Komponente K_i ist ausgefallen.

Die Bildung des Erwartungswertes der Indikatorvariablen ist die Wahrscheinlichkeit, daß die Komponente K_i funktionsfähig oder ausgefallen ist [6]

$$(3) \quad E[K_i] = 0 \cdot P[K_i = 0] + 1 \cdot P[K_i = 1] = P[K_i = 1].$$

Dies ist gleich der Komponentenzuverlässigkeit R_i [6]. Im Bereich von Flugzeugsystemen werden im Allgemeinen Komponentenausfälle altersunabhängig und damit rein zufällig angenommen, so daß die Komponenten-

zuverlässigkeit durch folgende *Exponentialverteilung* beschrieben wird [6]

(4) $R_i(t) = e^{\lambda_i t}$.

Die Verknüpfung der Variablen zu einer Systemarchitektur erfolgt über logische Operatoren **UND**, **ODER** und **NICHT**, die sich auch durch reell–algebraische Schreibweise ausdrücken lassen [6].
Unter der Annahme, daß die BOOLSCHEN Systeme Monotoniebedingungen erfüllen, kann die Systemfunktion ϕ auf Basis logisch verknüpfter Komponenten gebildet werden [6]:

(5) $\phi(\mathbf{K}) = 1$ System ϕ ist funktionsfähig

und

(6) $\phi(\mathbf{K}) = 0$ System ϕ ist ausgefallen.

Das RBD–Modell $\phi(\mathbf{K})$ des gesamten Flugzeug–Systems ist eine Funktion der in ihm enthaltenen Blöcke **K**

(7) $\phi(\mathbf{K})$ mit $\mathbf{K} = (K_1, ..., K_i, ..., K_m)^T$.

Im Rahmen der Systemmodellierung werden verschiedenartige Komponenten verwandt. Sie lassen sich in drei Block–Kategorien einordenen: *Hardware–Block*, *multifunktionaler Hardware–Block* und *multifunktionaler Software–Block*. In den beiden Hardwarebereichen sind den Blöcken entsprechende zuverlässigkeitstechnische Eigenschaften wie eine konstante oder exponentialverteilte Ausfallwahrscheinlichkeit $F(t)$, eine konstante Ausfallrate λ sowie ein Checkintervall zuzuordnen.
Über die logische Verknüpfung dieser Blöcke, entsprechend ihrer funktionellen Abhängigkeit im System, wird die Architektur abgebildet. Dies schließt auch eine Mehrfachverwendung einzelner Blöcke im RBD nicht aus, da alle Blöcke in der zuverlässigkeitstechnischen Analyse durch Orthogonalisierung der BOOLSCHEN Systemfunktion nur als einmal physikalisch auftretend bewertet werden.

2.2 HCFSM–Modell

Die untere Modellierungsebene besteht aus hierarchischen, nebenläufigen endlichen Automaten, die den RBD–Blöcken hinterlegt werden können. Das Bild 2 zeigt die in das RBD eingebetteten HCFSMs, sowie die Eingriffsmöglichkeit in das Gesamtmodell über den Komponenten–Fehlervektor **F** (Eingangsvektor) und die Ausgabe des aktuellen Zustands **Y** (Ausgangsvektor).

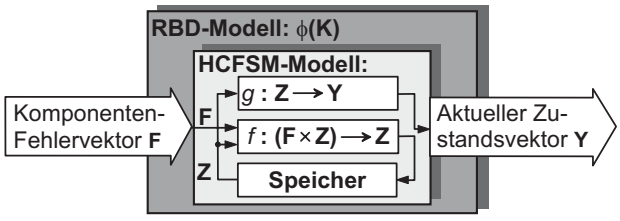


BILD 2: In das RBD–Modell eingebette HCFSM–Modell

Beschrieben werden können die HCFSMs durch einen 6–Tupel $(\mathbf{F}, \mathbf{Y}, \mathbf{Z}, \hat{\mathbf{Z}}, f, g)$[2]:

- **F** ist der Eingangsvektor,
- **Y** ist der Ausgangsvektor,
- **Z** ist die endliche nichtleere Menge interner Zustände,
- $\hat{\mathbf{Z}} \subseteq \mathbf{Z}$ ist die Menge der Anfangszustände,
- $f : \mathbf{F} \times \mathbf{Z} \rightarrow \mathbf{Z}$ heißt Übergangsfunktion,
- $g : \mathbf{Z} \rightarrow \mathbf{Y}$ heißt Ausgangsfunktion.

Eine spezielle Eigenschaft dieser Automaten ist, daß die Ausgaben **Y** von den internen Zuständen **Z** über die Ausgangsfunktion g erzeugt werden und damit unabhängig von der Eingabe sind. Graphisch werden diese sogenannten MOORE–HCFSMs durch Zustandsdiagramme repräsentiert. Das sind gerichtete Graphen $G(V,E)$ in denen jeder Knoten $v \in V$ einen Zustand und jede Kante $e \in E$ eine Transition darstellt [2, 5]. Tritt in einem aktuellen Zustand $Z \in \mathbf{Z}$ ein entsprechendes Eingabeereignis $F \in \mathbf{F}$ ein, dann erfolgt ein Zustandsübergang unter der Voraussetzung, daß das Eingabeereignis die Transitionsbedingung T erfüllt.
Die Hierarchisierung der endlichen Automaten wird durch die Zerlegung von Superzuständen in Subzustände erzielt [2]. Als ein anschauliches Beispiel erweist sich die Unterteilung des fehlerfreien Anfangszustands eines Systems in die Menge einzelner Anfangszustände $\hat{\mathbf{Z}}$ der HCFSMs, die den entsprechenden RBD–Blöcken hinterlegt sind.
Desweiteren sind diese endlichen Automaten durch Nebenläufigkeit charakterisiert. Bei dieser Modellform können einerseits die Transitionen zwischen den Zuständen eines HCFSM–Modells in Abhängigkeit von Zuständen benachbarter HCFSMs schalten [5]. Bild 3 zeigt die Serienverknüpfung von zwei Komponenten im RBD, deren HCFSMs beispielhaft über den Zustand *„aktiv“* von Komponente 1 und die Transition T_1 von Komponente 2 gekoppelt sind. Andererseits treten Verkopplungen immer dann auf, wenn durch ein Eingabeereignis die Transitionsbedingungen mehrer HCFSMs erfüllt sind. Diese Art der Verkopplung wird in Bild 3 durch die Transition T_2 dargestellt, wo beide HCFSMs zu den Komponenten 1 und 2 auf Basis desselben Eingabeereignisses F_2 schalten können.

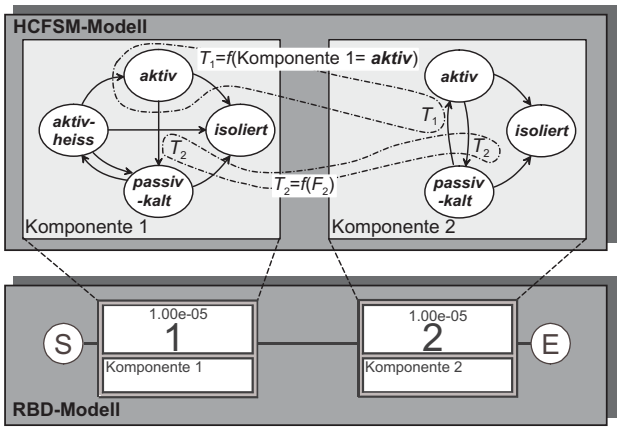


BILD 3: Kopplung der HCFSMs

Die formelle HCFSM–Modellbeschreibung erfordert zunächst die Darstellung der Modellierungsoptionen im Breich der HCSFMs in Abhängigkeit von den darüberliegenden RBD–Blöcken. Die Optionen sind anwendungsspezifisch und unterteilen sich in drei Kategorien (Bild 4).
Der einfache Hardware–Block in Bild 4a wird eingesetzt, um Hardware–Komponenten wie beispielsweise Stellsysteme, Spannungsversorgungen etc. abzubilden. Um diesen Block im Rahmen des Redundanzmanagements einzusetzen, wird eine einzelne HCFSM im Hintergrund betrieben.
Ein Spezialfall des Hardware–Blocks ist der multifunktionale Hardware–Block in Bild 4b. Eine Abbildung dieses Blocks erfolgt bei Hardware–Komponenten, die von verschiedenen Anwendungen – multifunktional – genutzt werden. Hierzu zählen beispielsweise Eingangs–/Ausgangs–Einheiten auf Rechnersystemen, Bussysteme, Internet–Switches etc.. Der Hintergrundbetrieb verschiedener HCFSMs ist bei diesen Anwendungen notwendig.
Der multifunktionale Software–Block in Bild 4c wird verwandt, um einzelne bzw. auch integrierte Software–Applikationen abzubilden, die auf einer gemeinsam genutzten Rechnerressource durch Partitionierung betrieben werden. Dies erfordert den Hintergrundbetrieb mehrerer HCFSMs.
Aufgrund des RBD–Hintergrundbetriebs der HCFSMs läßt sich dieses Modell nur in Abhängigkeit der RBD–Komponenten K_i beschreiben. Dies führt im HCFSM–Modell auf einen Automatenvektor (8), der aus einzelnen Automaten $A_{ij(i)}[K_i]$ besteht. Der Index $j(i) \in \mathbb{N}$ beschreibt in diesem Zusammenhang die Anzahl der im Hintergrund betriebenen HCFSMs zu einzelnen Komponenten $i = 1,\ldots,m$

$$(8) \quad \mathbf{A} = \begin{pmatrix} \{A_{11}[K_1],\ldots,A_{1j(1)}[K_1]\} \\ \vdots \\ \{A_{i1}[K_i],\ldots,A_{ij(i)}[K_i]\} \\ \vdots \\ \{A_{m1}[K_m],\ldots,A_{mj(m)}[K_m]\} \end{pmatrix} \quad \forall \quad j(i) \in \mathbb{N}.$$

Jeder der endlichen Automaten im Automatenvektor (8) kann durch ein Zustandsdiagramm mit bis zu fünf Zuständen und 16 Transitionen repräsentiert werden (Bild 5).

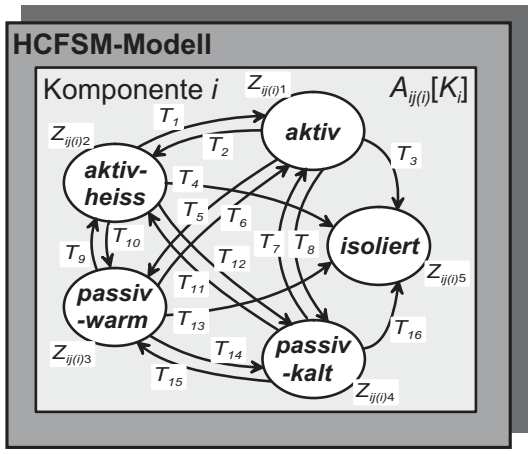


BILD 5: Zustände und Transitionen einer HCFSM

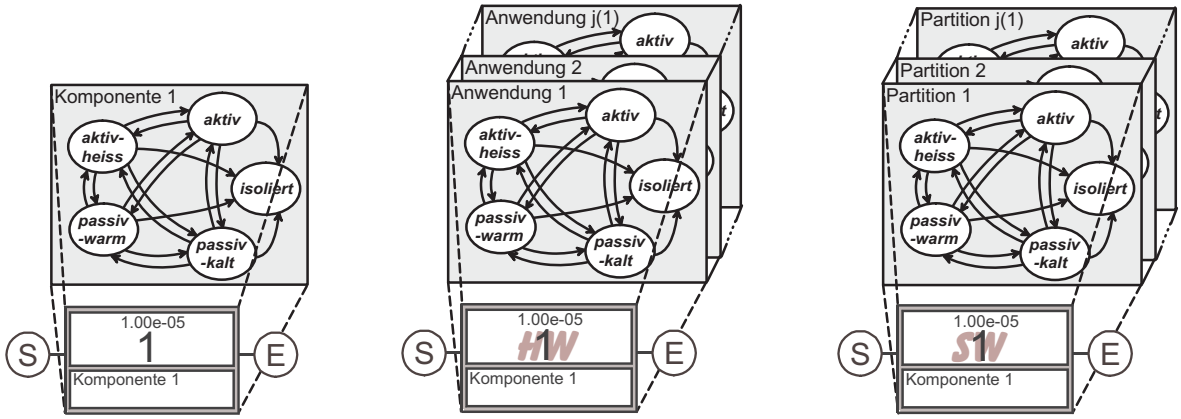
Die maximale Zustandsanzahl einzelner HCFSMs aus dem Automatenvektor (8) wird durch folgende Zustandsmenge beschrieben

$$(9) \quad \tilde{\mathbf{Z}}_{ij(i)} = \{Z_{ij(i)1},\ldots,Z_{ij(i)5}\}, \quad \forall\ i = 1,\ldots,m \quad \text{und} \quad j(i) \in \mathbb{N}.$$

Die Belegung der HCFSMs–Zustände mit entsprechenden Redundanzmerkmalen ist in Tabelle 1 aufgeführt.

Zustände der HCFSMs	Redundanz–merkmale	Endung
$Z_{ij(i)1}$	„aktiv“	a
$Z_{ij(i)2}$	„aktiv–heiss“	h
$Z_{ij(i)3}$	„passiv–warm“	w
$Z_{ij(i)4}$	„passiv–kalt“	k
$Z_{ij(i)5}$	„isoliert“	i

TAB 1: Zuordnung der Komponentenzustände



(a) Hardware–Block (b) Multifunktionaler Hardware–Block (c) Multifunktionaler Software–Block

BILD 4: *RBD–Blöcke mit hinterlegten HCFSMs*

Da es bei den meisten Systemarchitekturen nicht notwendig ist, alle fünf Zustände im Systemmodell abzubilden, wird eine Teilmenge verwandt

$$(10)\quad \begin{aligned} \mathbf{Z}_{ij(i)} &\subseteq \tilde{\mathbf{Z}}_{ij(i)}, \\ \forall\ i &= 1,\ldots,m \quad \text{und} \quad j(i) \in \mathbb{N}. \end{aligned}$$

Aus der Beschreibung der HCFSM–Zustandsteilmengen (10) wird der Zustandsvektor bestimmt

$$(11)\quad \mathbf{Z} = \begin{pmatrix} \{\mathbf{Z}_{11},\ldots,\mathbf{Z}_{1j(1)}\} \\ \vdots \\ \{\mathbf{Z}_{i1},\ldots,\mathbf{Z}_{ij(i)}\} \\ \vdots \\ \{\mathbf{Z}_{m1},\ldots,\mathbf{Z}_{mj(m)}\} \end{pmatrix} \quad \forall\ j(i) \in \mathbb{N}.$$

Ein Spezialfall des Zustandsvektors (11) ist der Vektor der Anfangszustände zum Zeitpunkt der Inbetriebnahme t_0

$$(12)\quad \hat{\mathbf{Z}}(t_0) \subseteq \mathbf{Z}.$$

In diesem Vektor wird der fehlerfreie Anfangsbelegung der HCFSMs festgelegt, von dem aus das System während des Betriebs $t_s \geq t_0$ unter Fehlereinflüssen degradiert.
Die Transitionen der HCFSMs in Bild 5 beschreiben die Übergänge zwischen den Zuständen (9). Bei einer vollständigen HCFSM werden die maximal 16 Transitionen durch nachfolgende Transitionsmenge charakterisiert

$$(13)\quad \begin{aligned} \tilde{\mathbf{T}}_{ij(i)} &= \{T_{ij(i)1}(Z_{ij(i)2}|Z_{ij(i)1}),\ldots \\ &\quad \ldots,T_{ij(i)16}(Z_{ij(i)4}|Z_{ij(i)5})\}, \\ \forall\ i &= 1,\ldots,m \quad \text{und} \quad j(i) \in \mathbb{N}. \end{aligned}$$

Die Bedingungen zu denen die Zustandsübergänge erfolgen sollen, werden über logische Gleichungen definiert. Die Gleichungen, die in diesem Rahmen von SYRELAN^{TM} verarbeitet werden können, stützen sich auf eine spezielle Syntax, welche die Komponentenzustände über die Endungen aus Tabelle 1 adressieren. Nachfolgend wird beispielhaft das Ansprechen des Zustands „aktiv–heiss“ einer HCFSM zum Hardware–Block von Komponente K_1 aufgezeigt

(14) 1_h.

Darüber hinaus muß bei den HCFSMs der multifunktionalen Hardware– und Software–Blöcke ein Unterscheidungskriterium bereitgestellt werden, das ein einzelnes Ansprechen der verschiedenen endlichen Automaten ermöglicht. Dazu wird neben der Identifikation des RBD–Blocks zur Komponente K_1 und der Zuordnung des Zustands „passiv–warm“ eine Identifikation hinzugefügt, die in diesem Beispiel die HCFSM „Partition1“ anspricht

(15) 1_Partition1_w.

Verknüpft werden die adressierten Zustände im Software-Tool über die logischen Operatoren **UND**, **ODER** und **NICHT** (&, v,-).

Auch die Transitionen der HCFSM einer Komponente K_i sind abhängig vom Gesamtmodellkontext in den die Komponente eingebettet ist. Dies hat zur Konsequenz, daß die Menge der Transitionen nur eine Teilmenge von (13) ist, weil die meisten Systeme eine nicht vollständige Zustandsübergangsbeschreibung wie in Bild 5 benötigen

$$(16)\quad \begin{aligned} \mathbf{T}_{ij(i)} &\subseteq \tilde{\mathbf{T}}_{ij(i)} \\ \forall\ i &= 1,\ldots,m \quad \text{und} \quad j(i) \in \mathbb{N}. \end{aligned}$$

Auf Basis dieser Zustandsteilmengen, läßt sich der Transitionsvektor des Gesamtsystems aufstellen

$$(17)\quad \mathbf{T} = \begin{pmatrix} \{\mathbf{T}_{11},\ldots,\mathbf{T}_{1j(1)}\} \\ \vdots \\ \{\mathbf{T}_{i1},\ldots,\mathbf{T}_{ij(i)}\} \\ \vdots \\ \{\mathbf{T}_{m1},\ldots,\mathbf{T}_{mj(m)}\} \end{pmatrix} \quad \forall\ j(i) \in \mathbb{N}.$$

Der Eingangsvektor $\mathbf{F}$ enthält die ins System zu injizierenden Fehler $\neg A_{ij(i)}[K_i]$. Beim einfachen Hardware–Block bewirkt dies den Übergang der einzelnen HCFSM in den Fehlerzustand „isoliert“, sowie bei den multifunktionalen Hardware–Blöcken den simultanen Übergang sämtlicher HCFSMs in den Zustand „isoliert“. Im Gegensatz dazu kann bei den multifunktionalen Software–Blöcken eine einzelne HCFSM individuell durch Fehlerinjektion in den Zustand „isoliert“ transformiert werden. Der entsprechende vollständige Fehlervektor lautet

$$(18)\quad \begin{aligned} \mathbf{F} &= \begin{pmatrix} \{\neg A_{11}[K_1],\ldots,\neg A_{1j(1)}[K_1]\} \\ \vdots \\ \{\neg A_{i1}[K_i],\ldots,\neg A_{ij(i)}[K_i]\} \\ \vdots \\ \{\neg A_{m1}[K_m],\ldots,\neg A_{mj(m)}[K_m]\} \end{pmatrix} \\ &\forall \quad j(i) \in \mathbb{N}. \end{aligned}$$

Da es sich bei den HCFSMs um einen MOORE–Automaten handelt, wird jedem Zustand im System eine Ausgabe in Form einer Farbzuweisung an den darüberliegenden Block zugeordnet. Die Ausgänge einer HCFSM mit fünf Zuständen, werden durch die nachfolgende Menge beschrieben, deren Elemente eine Farbzuordnung entsprechend Tabelle 2 erhalten

$$(19)\quad \begin{aligned} \tilde{\mathbf{Y}}_{ij(i)}[Z_{ij(i)}] &= \{Y_1[Z_{ij(i)1}],\ldots,Y_5[Z_{ij(i)5}]\}, \\ \forall\ i &= 1,\ldots,m \quad \text{und} \quad j(i) \in \mathbb{N}. \end{aligned}$$

Ausgänge der HCFSM–Zustände	**Farbzuordnungen**
$Y_1[Z_{ij(i)1}]$	GRÜN
$Y_2[Z_{ij(i)2}]$	GELB
$Y_3[Z_{ij(i)3}]$	ORANGE
$Y_4[Z_{ij(i)4}]$	HELLBLAU
$Y_5[Z_{ij(i)5}]$	ROT

TAB 2: Farbzuordnungen zu den Ausgängen

Durch die Verknüpfung der Ausgänge mit entsprechend ausgewählten Zuständen (10) kann auch der Ausgangsvektor eines Systems nur durch Teilmengen der Ausgänge (19) repräsentiert werden

$$(20)\quad \mathbf{Y}_{ij(i)}[Z_{ij(i)}] \subseteq \tilde{\mathbf{Y}}_{ij(i)}[Z_{ij(i)}] \qquad \forall\ i = 1,\ldots,m \quad \text{und} \quad j(i) \in I\!N.$$

Aus dieser Beziehung läßt sich der vollständige Ausgangsvektor eines Systems aufstellen

$$(21)\quad \mathbf{Y} = \begin{pmatrix} \{\mathbf{Y}_{11}[\mathbf{Z}_{11}],\ldots,\mathbf{Y}_{11}[\mathbf{Z}_{1j(1)}]\} \\ \vdots \\ \{\mathbf{Y}_{i1}[\mathbf{Z}_{i1}],\ldots,\mathbf{Y}_{ij(i)}[\mathbf{Z}_{ij(i)}]\} \\ \vdots \\ \{\mathbf{Y}_{m1}[\mathbf{Z}_{m1}],\ldots,\mathbf{Y}_{mj(m)}[\mathbf{Z}_{mj(m)}]\} \end{pmatrix} \qquad \forall \quad j(i) \in I\!N.$$

3 REDUNDANZMANAGEMENT

Die fehlertolerante Auslegung von Systemen ist gerade im Bereich der Flugzeugsysteme das gängige Konzept im Entwurf solcher Systeme, um neben der System–Funktionalität auch die Zuverlässigkeits– und Sicherheitsanforderungen zu erfüllen. Die Fehlertoleranz bezeichnet die Fähigkeit des Systems, auch mit einer begrenzten Anzahl fehlerhafter Komponenten die spezifizierte Funktion zu erfüllen. Dies erfordert Redundanz. Die Redundanz ist dabei das Vorhandensein von mehr technischen Mitteln als zum Betrieb des Systems notwendig wären [1].

Damit ein fehlertoleranter Systembetrieb realisierbar ist, muß das System über zwei Eigenschaften verfügen. Das ist einerseits die Fehlerdiagnose, bei der Redundanz in Form von BITE–Komponenten (BITE, *engl.* **B**uild **I**n **T**est **E**quipment) eingesetzt wird, um eine fehlerhafte zu detektieren und zu lokalisieren [1]. Andererseits wird Redundanz zur Fehlerbehandlung verwandt. Nach Identifizierung eines Fehlers, wird dieser durch die Fehlerbehandlung eingegrenzt. Dem schließen sich strukturelle Maßnahmen zur Funktionserhaltung an. Die sogenannte Fehlerausgrenzung entfernt die fehlerhaften Komponenten aus dem System und sichert durch die Rekonfigration eine Überführung in einen funktionsfähigen Zustand [1].

Dies erfordert eine Strategie die festlegt, wie sich das System unter Fehlereinflüssen zu verhalten hat, um die gewünschte Funktionalität wiederzuerlangen. Dieser Prozeß aus *Fehlerausbreitung* und *Rekonfiguration* bezeichnet die Arbeitsweise des *Redundanzmanagements.*

3.1 Fehlerausbreitung und Rekonfiguration

Die Durchführung des Redundanzmangements stützt sich nach der Injektion eines Fehlers auf zwei Prozesse, die Fehlerausbreitung und die Rekonfiguration. Damit diese beiden Prozesse eine realitätsnahe und zeiteffektive Umsetzung erfahren, werden bei der algorithmischen Implementierung neben RBD–Eigenschaften auch Prinzipien von fehlertoleranten und rechnergestützten Systemen angewandt.

Das Bild 6 zeigt zur Veranschaulichung des Redundanzmanagement–Algorithmus das RBD einer Höhenrudersteuerung eines Verkehrflugzeugs mit hinterlegten HCFSMs. Die aktuellen Zustände der einzelnen endlichen Automaten sind hier aus Übersichtlichkeitsgründen durch angefügte Zustandsattribute nach den Tabellen 1 und 2 den Blöcken zugeordnet, damit die Unterscheidung zwischen den Zuständen möglich ist. Ausgenutzt wird in diesem Rahmen die strukturelle Eigenschaft des RBD–Modells. Da dieses Modell die Information über die Verknüpfung von Komponenten besitzt, können sich die Fehlerausbreitungs– und Rekonfigurationsprozesse an den logischen Abhängigkeiten benachbarter Blöcke orientieren.

Dem Fehlerausbreitungsprozeß wird in diesem Zusammenhang eine Suchstrategie für benachbarte Blöcke überlagert. Differenziert wird dabei zwischen der fehlerhaften Komponente sowie deren direkten Nachbarn. Das Kriterium ist, daß die Suchstrategien sich an deren Zuständen „aktiv“ oder nicht „aktiv“ auf Basis der HCFSMs ausrichten. Zu den Zuständen nicht „aktiv“ zählen „aktiv–heiss“, „passiv–warm“, „passiv–kalt“ und „isoliert“.

Bild 6 zeigt bei der Fehlerinjektion in Komponente K_{16} den Fehlerausbreitungsprozeß $FA1_1$, der sich an den benachbarten Blöcken im Zustand „aktiv“ orientiert. Ausgehend von der fehlerhaften Komponente wird dabei zunächst die Fehlerausbreitung hin zum START–Block erfolgen. Um anschließend eine schnelle Rekonfiguration einleiten zu können, werden die Rekonfigurations–Knotenpunkte $\{RK_1, RK_2, RK_3\}$ des RBDs gespeichert, die aus Sicht der Komponente mit dem aktuell schaltbaren HCFSM linksseitig liegen und zu dieser mindestens eine parallele Komponente besitzen müssen. Der zweite Teil der Fehlerausbreitung $FA1_2$ wirkt von der fehlerhaften Komponente K_{16} hin zum END–Block. Für beide Fehlerausbreitungsrichtungen gilt natürlich, daß bei RBD–Knotenpunkten sämtliche Verzweigungen in parallele Pfade untersucht werden.

Bei der Fehlerinjektion in einer Komponente deren HCFSMs sich in einem der Zustände nicht „aktiv“ befindet, wird nach benachbarten Komponenten gesucht, die schaltbare Transitionen besitzen. Auch hier gilt, ähnlich wie bei der Suche nach Komponenten im Zustand „aktiv“, daß ausgehend von der fehlerhaften Komponente K_{18} eine Fehlerausbreitung $FA2_1$ hin zum START–Block einschließlich der Speicherung von Rekonfigurations–Knotenpunkten $\{RK_4, RK_2, RK_3\}$ erfolgt, sowie die Fehlerausbreitung $FA2_2$ hin zum END–Block (Bild 6).

Die Rekonfiguration erfolgt dann über die Komponenten die rechtsseitig der Rekonfigurations–Knotenpunkte liegen. Dies schließt natürlich die Komponenten aus, deren HCFSMs bei der Fehlerausbreitung schon geschaltet haben.

Bei den meisten rechnergestützen Systemen werden detektierte Fehler, sofern sie nicht auf der Rechnerressource selbst auftreten und vom Betriebssystem verarbeitet werden, in der Applikations–Software behandelt.

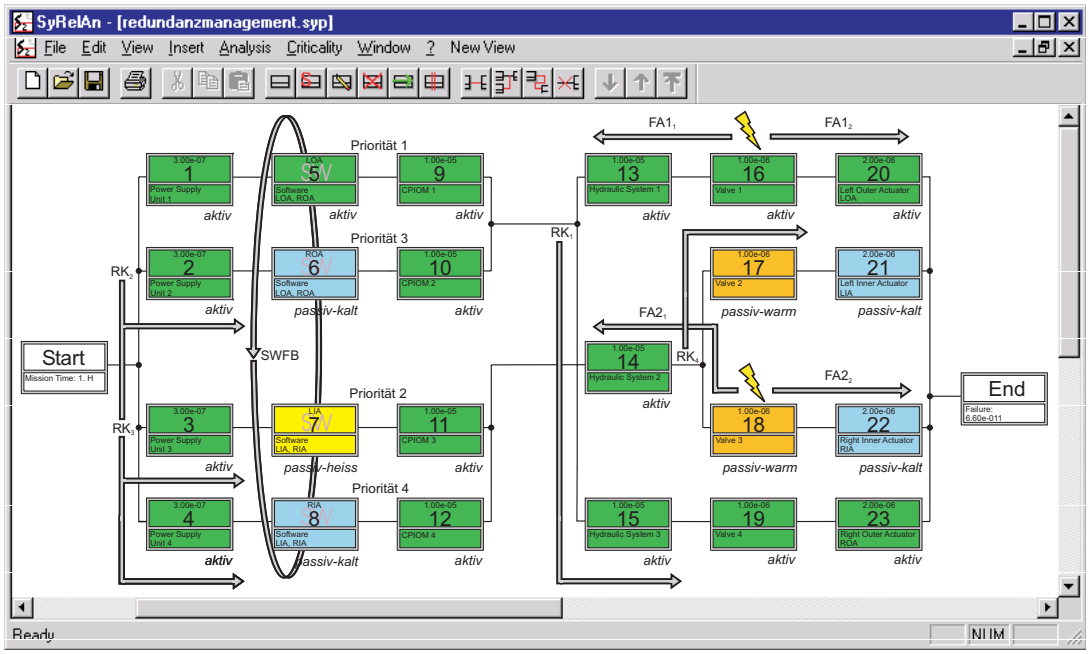


BILD 6: Fehlerausbreitungs– und Rekonfigurationsprozesse innerhalb des Redundanzmangements

Dementsprechend enthält die Applikations–Software bestimmte Routinen, wie im Fehlerfall zu verfahren ist, damit Systemfunktionalität gewährleistet werden kann. Diese Maßnahmen werden auch im Rahmen der Visualisierung des Redundanzmangements ausgenutzt. Nach der Fehlerausbreitung und der Rekonfiguration über die Rekonfigurationsknotenpunkte werden deshalb die HCFSMs der Software–Applikationen hinsichtlich schaltbarer Transitionen untersucht. Bild 6 zeigt den Schleifendurchlauf der Software–Fehlerbehandlung (SWFB) für Software–Applikationen, der bei einer schaltbaren HCFSM unterbrochen wird. In diesem Fall schließt sich eine Fehlerbehandlung wie bei der Suche nach schaltbaren Blöcken nicht „aktiver“ Komponenten ($FA2_1$ und $FA2_2$) sowie die Rekonfiguration über die darüber aufgenommenen Rekonfigurationsknotenpunkte an. Beendet wird die Schleife, wenn keine HCFSM einer Software–Applikation schaltbar ist.
In seltenen Fällen besteht dennoch die Möglichkeit, daß die bisher vorgestellten Maßnahmen nicht alle schaltbaren HCFSMs berücksichtigen. Dies bedingt, daß sich der Software–Schleife noch eine Überprüfung sämtlicher HCFSMs (KFB) entsprechend zugehöriger Block–Identität (Block–ID) anschließt. Wird in diesem Zusammenhang eine HCFSM mit schaltbarer Transition lokalisiert, dann wird wie bei den zuvor vorgestellten Software–Applikationen verfahren. Die Überprüfung der HCFSMs ist genau dann abgeschlossen, wenn keine mehr schaltbar ist.

3.2 Redundanzmanagement Algorithmus

Eine Verknüpfung der vorgestellten Fehlerbehandlungs– und Rekonfigurations–Prozesse zu einem gesamten Algorithmus erfolgt in diesem Unterkapitel.
Eine Eigenschaft dieses Algorithmus ist, daß immer nur eine Komponente über einen RBD–Block als fehlerhaft deklariert werden kann. Das hat zur Konsequenz, daß die Ausfallwahrscheinlichkeit der Komponente auf $F(t) = 1$ gesetzt wird und die unterliegende HCFSM in den Zustand „isoliert“ übergeht. Eine Unterscheidung wird in diesem Rahmen zwischen den multifunktionalen Hardware– und Software–Blöcken getroffen. Das Kriterium ist, daß bei den Hardware–Blöcken ein Simultanausfall aller HCFSMs erfolgt und die Ausfallwahrscheinlichkeit der abgebildeten Komponente auf $F(t) = 1$ gesetzt wird. Dem entgegen werden bei den Software–Blöcken die HCFSMs einzeln als fehlerhaft deklariert.
Um bei den Fehlerausbreitungsprozessen die beiden Suchstrategien nach Komponenten im Zustand „aktiv“ und im Zustand nicht „aktiv“ durchzuführen, wird ein Suchalgorithmus aus der *Künstlichen Intelligenz* mit TIEFE–ZUERST–SUCHE als Suchstrategie eingesetzt [4]. Dieser Such–Algorithmus bewegt sich sowohl im RBD–Modell, wo die logische Abhängigkeit benachbarter Komponenten ausgenutzt wird, als auch im HCFSM–Modell, dessen aktuelle Zustände in Verbindung mit schaltbaren Transitionen als Orientierung im Rahmen der Suchstrategien verwandt werden.
Bild 7 zeigt die Bestandteile des Redundanzmanagement–Algorithmus, dessen erste Abfragen sich auf fehlerhaft deklarierte Komponenten sowie deren Nachbarkomponenten beziehen und testen, ob diese sich im Zustand „aktiv“ oder nicht „aktiv“ befinden. Auf Basis der Auswahl wird der entsprechende Fehlerausbreitungsprozeß angewandt, bei dem Rekonfigurationsoptionen über Rekonfigurations–Knotenpunkte abgespeichert werden.
Anhand der Struktur in Bild 7 ist ersichtlich, daß der Algorithmus das Auftreten einer Komponente i mit unterschiedlichen Block–IDs zuläßt. Dies unterstützt eine wichtige Eigenschaft der Zuverlässigkeitsanalyse auf Basis des RBD–Modells, wo im Rahmen der RBD–Modellierung

das mehrfache Auftreten einer Komponente *i* im Modell zulässig ist, diese jedoch physikalisch in der Analyse nur als einmalig auftretend berücksichtigt wird. Eine Unterscheidung identischer Komponenten findet über unterschiedliche Block–IDs statt.
Die zu durchlaufenden Fehlerausbreitungen und Rekonfigurationen sind im Rahmen dieses Algorithmus erst dann beendet, wenn keine schaltbaren Transitionen mehr im System verfügbar sind und damit ein stabiler aber degradierter Systemzustand erreicht ist.

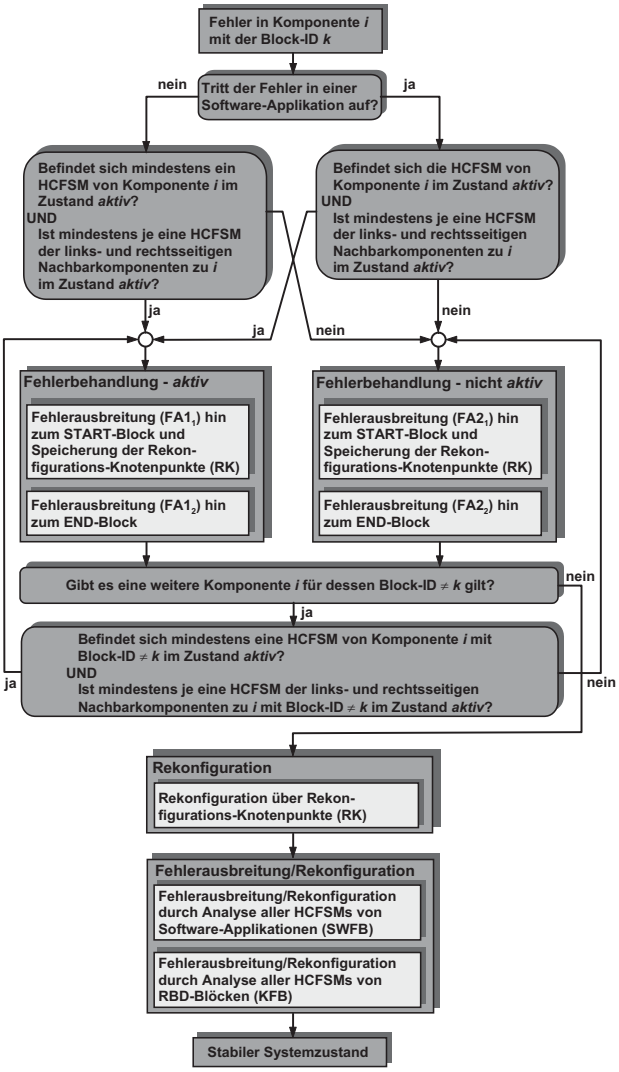


BILD 7: Redundanzmanagement–Algorithmus

4 SYNTHESE UND ANALYSE

Der Entwurf fehlertoleranter Flugzeug–Systemarchitekturen mit diesem Software–Tool stützt sich auf die Synthese und Analyse des Systems im fehlerfreien Nominalzustand und in degradierten Systemzuständen. Beispielhaft wird hierzu die Modellierung und Analyse der Höhenrudersteuerung eines Verkehrsflugzeugs auf RBD– und HCFSM–Ebene vorgestellt. Bei diesem System handelt es sich um ein sicherheitskritisches System der primären Flugsteuerung; ein unsymmetrischer Ausschlag großer Amplitude von Aktuatoren der linken und rechten Ruderseite ist nach JAR 25 als „Catastrophic“ zu klassifizieren und der Fehler mit der entsprechenden Eintrittswahrscheinlichkeit von maximal $F = 10^{-9}$ pro Flugstunde verbunden [3]. Eine unsymmetrische Ansteuerung der Aktuatoren führt zu einem Torsionsmoment auf die Flugzeugzelle, was in Konsequenz fatale Folgen für das Flugzeug und dessen Insassen haben kann.

4.1 Synthese und Analyse des RBD–Modells

Ausgangspunkt des Systementwurfs bildet das TOP EVENT auf dessen Basis die Modellierung und Analyse erfolgt. Für die Höhenrudersteuerung lautet es:
„*Symmetrische Funktionsfähigkeit der fehlertoleranten Höhenrudersteuerung eines Verkehrsflugzeugs*“.
Bild 6 zeigt das RBD bezüglich der Definition des TOP EVENTS, in dem voneinander abhängige Komponenten des Systems durch logische Verknüpfung RBD–Blöcke abgebildet sind. Jedem RBD–Block wird in diesem Zusammenhang eine Ausfallrate λ zugeordnet. Ausgenommen davon sind jedoch die Software–Blöcke $\{K_5,\ldots,K_8\}$, denen nur die Wahrscheinlichkeiten *funktionsfähig* $(F = 0)$ und *ausgefallen* $(F = 1)$ zugeordnet werden können. An die Funktionalität des Systems ist bei diesem Beispiel der Symmetriebetrieb der linken und rechten Ruderseite über eine sogenannte *n*–von–*m* Bedingung geknüpft [1]. Die Eingabe erfolgt über logische Nebenbedingungen (22) in der RBD–Modellierung und wird in der Analyse entsprechend berücksichtigt [6]

$$(22)\quad \Gamma = (K_{20} \wedge K_{22}) \vee (K_{20} \wedge K_{23}) \vee \ldots \\ \ldots \vee (K_{21} \wedge K_{22}) \wedge (K_{21} \vee K_{23})\,.$$

Zur Zuverlässigkeitsanalyse des fehlerfreien Nominalsystems wird der sogenannte CAOS Orthogonalisierungs–Algorithmus auf das RBD–Modell des Systems angewandt [6]. Die Wahrscheinlichkeit, daß in diesem Initialzustand ein Systemausfall in einer Flugstunde eintritt beträgt $F(1h) = 6,60 \cdot 10^{-11}$ und erfüllt damit die Zuverlässigkeitsanforderungen. Er handelt sich bei der Ausfallwahrscheinlichkeit um eine *konservative* Berechnung, da die redundanten Komponenten ausschließlich als „aktiv–heiss“ betrachtet werden.

4.2 Synthese und Analyse des HCFSMs–Modells

Aufbauend auf der in verschiedenen Industrieanwendungen schon bewährte zuverlässigkeitstechnische Modellierungs– und Analyseumgebung von SYRELAN^{TM}, wird die neuartige Erweiterung des Software–Tools hinsichtlich der Visualisierung des Redundanzmanagements am Beispiel Höhenrudersteuerung aus Bild 6 vorgestellt.
Ausgangpunkt bildet das RBD–Modell, in dem jeder einzelnen Komponente ein bzw. mehrere HCFSMs zugeordnet werden. Die Mehrfachzuordnung von HCFSMs erfolgt in diesem Beispiel bei den multifunktionalen Software–Blöcken. In Bild 8 ist der BLOCK EDITOR der erweiterten SYRELAN^{TM} Version abgebildet, in dem über die TYPE–Option die Eigenschaft eines Software–Blocks K_7 ausgewählt wurde.

Aus der Struktur des RBD–Modells in Bild 6 ist ersichtlich, daß die Rechnermodule CPIOM 1 und CPIOM 2 (*engl.* **Com**puting **I**nput **O**utput **M**odule) auf Basis partitionierter Software–Applikationen die Aktuatoren LOA und ROA regeln können; CPIOM 3 und CPIOM 4 entsprechend die Aktuatoren LIA und RIA. Die Trennung der HCFSMs innerhalb der multifunktionalen RBD–Blöcke erfolgt durch die Auswahl der HCFSMs unter DIAGRAM. Entsprechend wird der HCFSM eine Kennung zur Software–Applikation zugeordnet, die in Bild 6 und 8 bei Komponente K_5 mit „LIA“ spezifiziert ist.
Um für das System den fehlerfreien Vektor der Anfangszuständ $\hat{\mathbf{Z}}(t_0)$ (12) festzulegen, wird im BLOCK EDITOR der INITIAL STATE aus der „drop–down list“ ausgewählt (Bild 8). Nicht erreichbare Zustände im STATE DIAGRAM von Bild 8 sind, ebenso wie der Zustand „isoliert“, von der Liste ausgeschlossen.
Was im Bereich der Zuverlässigkeistanalyse über die logische Nebenbedingung (22) festgelegt wurde, muß auch für das Redundanzmanagement gelten. Im Betrieb des Höhenruders aus Bild 6 darf nur ein Aktuator je Stellfläche „aktiv“ arbeiten. Im fehlerfreien Systemzustand bedeutet dies, daß sich zwei Aktuatoren im Zustand „passiv“, sowie deren Software–Applikationen in den Schaltzuständen „aktiv–heiss“, „passiv–warm“ oder „passiv–kalt“ befinden. Die Unterteilung der Aktuator–Schaltzustände in den Software–Applikationen ist die Folge eines Priorisierungsprozesses, der die Schaltreihenfolgen von Aktuatoren im Fehlerfall vorgibt. Diese Priorisierung muß in den logischen Gleichungen der Transitionen im Vektor **T** (17) berücksichtigt werden. Festgelegt wird dies durch sich gegenseitig ausschließende BOOLSCHE Ausdrücke. Beispielhaft wird für die Komponente K_7 die Transitionsbedingung $T_{7,1,6}$ definiert, die den Zustandsübergang im Diagramm 1 („LIA“) von „aktiv–heiss“ zu „aktiv“ beschreibt. Verbal läßt sich diese Bedingung folgendermaßen beschreiben:
„*Ist kein anderes CPIOM für die über das CPIOM 3 zu regelnde linke Stellfäche im Zustand „aktiv“ und ist kein höher priorisiertes CPIOM für die über das CPIOM 3 zu regelnde linke Stellfäche im Zustand „aktiv–heiss“, so ist der Aktuator „LIA“ über das CPIOM 3 in den Zustand „aktiv“ zuversetzen.*“
Dazu lautet die logische Transitionsbedingung in der SYRELAN™ Syntax

$$
\begin{aligned}
(23)\quad T_{7,1,6} \;=\; & \big[(-5\_\mathrm{LOA}\_\mathrm{a} \;\&\; -20\_\mathrm{a}) \;\&\ldots \\
\ldots \;\&\; & (-6\_\mathrm{LOA}\_\mathrm{a} \;\&\; -20\_\mathrm{a}) \;\&\ldots \\
\ldots \;\&\; & (-8\_\mathrm{LIA}\_\mathrm{a} \;\&\; -21\_\mathrm{a})\big] \;\&\ldots \\
\ldots \;\&\; & \underbrace{\big[(-5\_\mathrm{LOA}\_\mathrm{h} \;\&\; -20\_\mathrm{p})\big]}_{\text{Priorisierung}}.
\end{aligned}
$$

Sind für sämtliche HCFSMs die Transitionen definiert und der Initialzustand gespeichert, kann die Simulation des Redundanzmangements ausgeführt werden, indem über INJECT FAILURE den Systemkomponenten Fehler injiziert werden (Bild 8). Auf das HCFSM–Modell wird der Redundanzmanagement–Algorithmus angewandt und bei Erreichen eines stabilen Zustands wird an die RBD–Ebene der aktuelle Zustand sämtlicher HCFSMs weitergeleitet, sowie die Ausfallwahrscheinlichkeit angezeigt, die sich für den degradierten Systemzustand einstellt (Bild 9).
Der Anwender kann auf Basis seiner Fehlersimulationen analysieren, ob das System wie gewünscht bei Fehlerinjektionen rekonfiguriert und gegebenenfalls Korrekturen vornehmen. Dies ist ein iterativer Prozeß, bei dem das System in den Initialzustand oder einen gespeicherten Zwischenzustand zurückversetzt wird. Dem schließen sich Korrekturen in den Transitionen an, bis das gewünschte Rekonfigurationsverhalten erzielt ist.
Auf Basis dieses interaktiven und anwenderfreundlichen Software–Tools kann auch für komplexe Flugzeugsysteme das Systemverhalten modelliert und analysiert werden. Dies schließt ein, daß neben den zuverlässigkeitstechnischen Aussagen auch Umschaltstrategien für Fehlerfälle definiert und bewertet werden, welche anschließend auf Applikationsebene des realen Systems ohne Risiko übertragen werden können.

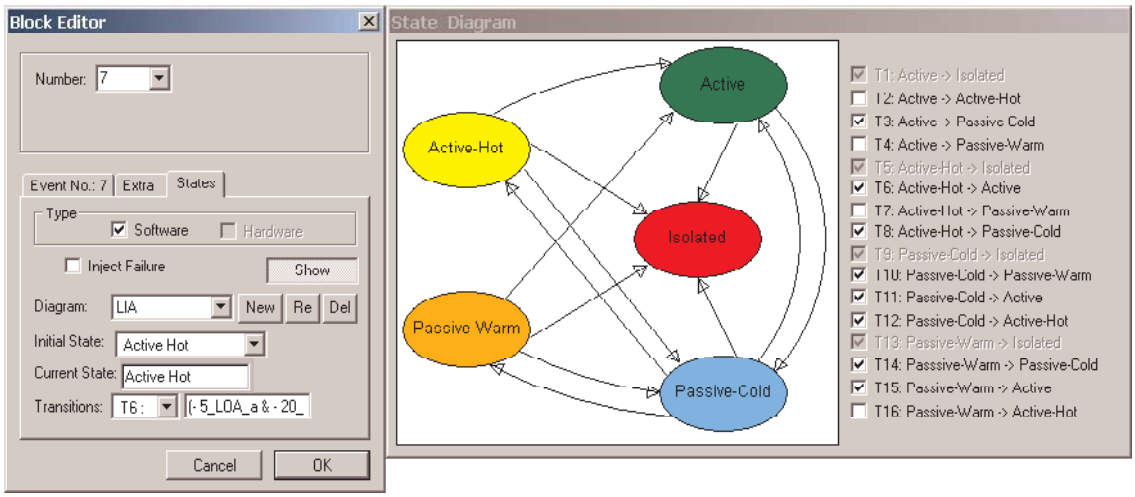


BILD 8: *Block Editor* und *State Diagram* von SYRELAN™

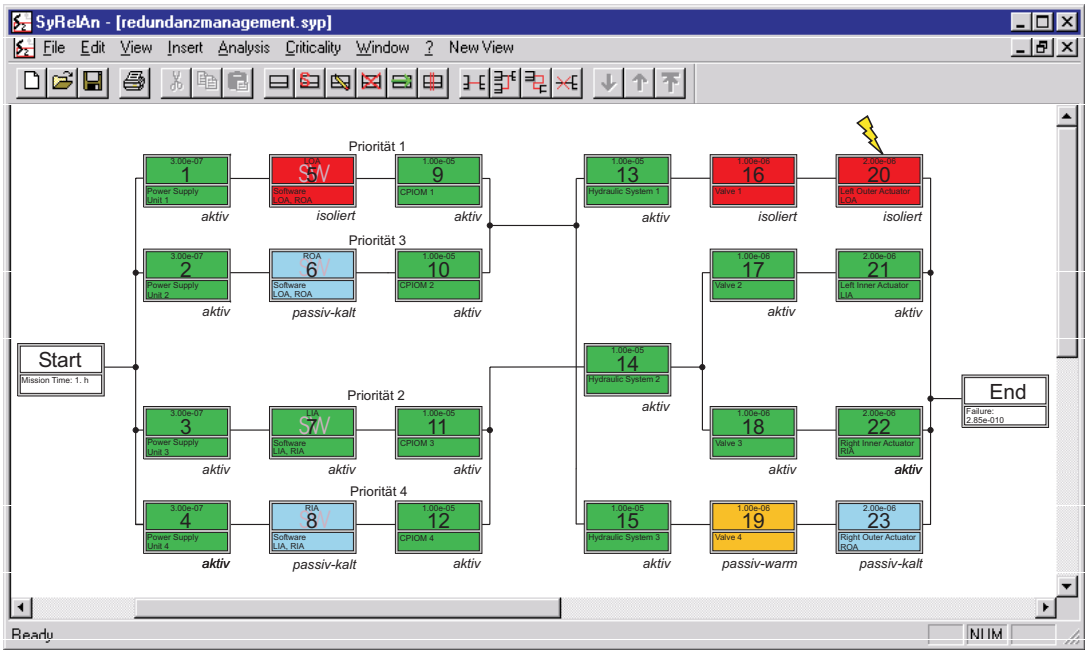


BILD 9: Rekonfigurierte Höhenrudersteuerung

5 ZUSAMMENFASSUNG

In diesem Bericht wurde ein Software–Tool vorgestellt, welches bereits in der Vorentwurfsphase von fehlertoleranten Flugzeugsystemen zur Synthese und Analyse in Bereichen der Zuverlässigkeit und des Redundanzmangements eingesetzt werden kann. Die Basis dazu bildet das hybride Systemmodell, welches sich in die obere RBD– und untere HCFSM–Modellierungsebene unterteilt. Die innovative Erweiterung des bereits im industriellen Einsatz bewährten Software–Tools wird durch die Abbildung des Redundanzmanagements auf der HCFSM–Ebene bereitgestellt, zu der eine formelle Beschreibung des zu Grunde liegenden HCSFM–Modells sowie des implementierten Algorithmus erfolgt.

Mit der Erweiterung von SyRelAn™ wird den Systemingenieuren eine Simulationsplattform zur Verfügung gestellt, die interaktiv und auf Basis einer anschaulichen Modellierung die Simulation verschiedenster Fehlerszenarien ermöglicht. Dazu zählt die Bewertung unterschiedlicher Rekonfigurationstrategien in einem iterativen Prozeß sowie die bewährte zuverlässigkeitstechnische Analyse des RBD–Modells in sämtlichen Zuständen.

Weitere Entwicklungsstufen des Tools sehen IMA–spezifische Modellierungseigenschaften der Funktionsintegration auf der Rechnerressource vor und die Berücksichtigung von Redundanzmerkmalen in der Zuverlässigkeitsanalyse.

DANKSAGUNG

Der Autor dankt Airbus Deutschland, Hamburg für die Finanzierung und freundliche Unterstützung des Forschungprojektes *Modellierung und zuverlässigkeitstechnische Analyse fehlertoleranter, vernetzter Systemarchitekturen auf Basis von IMA*.

SCHRIFTTUM

[1] Echtle, K.: *Fehlertoleranzverfahren.* Springer Verlag, Berlin Heidelberg, 1990.

[2] Gajki, D. D.; Vahid, F.; Narayan, S.; Gong, J.: *Specification and Design of Embedded Systems.* Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[3] Joint Aviation Authorities: *1 to JAR 25.1309 – Advisory Circular Joint to Aviation Requirements.* Civil Aviation Authority, London, 1989.

[4] Lunze, J.: *Künstliche Intelligenz für Ingenieure. Band 1: Methodische Grundlagen und Softwaretechnologie.* Oldenbourg Verlag, München Wien, 1994.

[5] Teich, J.: *Digitale Hardware/Software–Systeme.* Springer Verlag, Berlin Heidelberg, 1997.

[6] Vahl, A.: *Interaktive Zuverlässigkeitsanalyse von Flugzeug–Systemarchitekturen.* Dissertation, Arbeitsbereich Flugzeug–Systemtechnik, Technische Unversität Hamburg–Harburg, Fortschritt–Berichte VDI, Reihe 10, Nr. 565, Düsseldorf, 1998.

[7] Verein Deutscher Ingenieure (Hrsg.): *Mathematische Modelle für Redundanz.* VDI–Richtlinie 4008, VDI–Handbuch Technische Zuverlässigkeit, VDI–Verlag Düssledorf, 1986.