Weiterentwicklung der MCDET-Methode und des zugehörigen Rechenwerkzeugs für probabilistische Dynamikanalysen

GRS - 330

Gesellschaft für Anlagen-
und Reaktorsicherheit
(GRS) gGmbH

Weiterentwicklung der MCDET-Methode und des zugehörigen Rechenwerkzeugs für probabilistische Dynamikanalysen

Martina Kloos
Joerg Peschke
Josef Scheuer

Mai 2015

Anmerkung:

Das diesem Bericht zugrunde liegende F&E-Vorhaben RS1198 „Fortschrittliche Methoden und Werkzeuge für probabilistische Sicherheitsanalysen“ wurde im Auftrag des Bundesministeriums für Wirtschaft und Energie (BMWi) durchgeführt.

Die Arbeiten wurden von der Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH ausgeführt.

Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Auftragnehmer.

GRS - 330
ISBN 978-3-944161-10-5

Deskriptoren

Aleatorische Unsicherheit, Crew-Modul, dynamische PSA, epistemische Unsicherheit,
integrale deterministisch-probabilistische Sicherheitsanalyse, probabilistische Dynamik, MCDET

Kurzfassung

Mit dem Analysewerkzeug MCDET (*M*onte *C*arlo *D*ynamic *E*vent *T*ree) kann das Verhalten eines komplexen dynamischen Systems, bestehend aus technischen Komponenten und Systemen sowie einem physikalisch-chemischen Prozess, unter Berücksichtigung menschlicher Eingriffe realitätsnah modelliert und bewertet werden. Die implementierte Methode ist eine Kombination aus Monte-Carlo-Simulation und dynamischer Ereignisbaum-Methode.

MCDET kann prinzipiell in Verbindung mit jedem deterministischen Rechenprogramm zur Simulation eines dynamischen Ablaufs eingesetzt werden. In Reaktorsicherheitsanalysen wird MCDET z. B. mit den Rechenprogrammen ATHLET oder MELCOR angewendet, um Stör- bzw. Unfallabläufe unter Berücksichtigung des zufälligen Verhaltens technischer Komponenten und Systeme berechnen und bewerten zu können. Eine Quantifizierung des Einflusses epistemischer Unsicherheiten ist ebenfalls möglich.

Ziel der im Forschungs-und Entwicklungsvorhaben RS1198 durchgeführten Arbeiten war es, das Analysewerkzeug MCDET so weiterzuentwickeln, dass es zukünftig zu einem festen Bestandteil der verfügbaren Analysewerkzeuge für Reaktorsicherheitsanalysen wird. Durch den Einsatz von MCDET sollen sowohl Fragestellungen im Rahmen von probabilistischen (PSA) und deterministischen Sicherheitsanalysen (DSA) beantwortet als auch neue Nachweisverfahren für die Beherrschbarkeit von Störfall- und Unfallabläufen entwickelt werden können.

Die Arbeiten in diesem Vorhaben hatten zum Ziel, die praktische Durchführbarkeit und Effizienz von MCDET-Analysen zu verbessern. So wurden Benutzereingaben vereinfacht und standardisiert sowie ein Konzept für eine plattform-übergreifenden Benutzeroberfläche entwickelt. Zur Reduktion des Rechenaufwands wurde ein neuer Algorithmus für die Konstruktion dynamischer Ereignisbäume sowie ein neuer Ansatz für die Steuerung von Simulationsprozessen entwickelt.

Abstract

The software tool MCDET (*M*onte *C*arlo *D*ynamic *E*vent *T*ree) can be applied for an improved modeling and probabilistic assessment of complex systems with significant process/hardware/ software/firmware/human interaction. The implemented method is a combination of Monte Carlo simulation and the Dynamic Event Tree approach.

MCDET can be principally combined with any deterministic code for simulating dynamic processes. In reactor safety analyses, MCDET is e.g. coupled with ATHLET or MELCOR to consider the stochastic behaviour of safety systems and components during the simulation of incidents or accidents. The influence of epistemic uncertainties can be considered as well.

The enhancements performed within the research and development project RS1198 aimed at making MCDET an important part of the tools for reactor safety analyses. MCDET is to be applied to answer questions within deterministic as well as probabilistic safety analyses or to develop approaches for safety demonstrations with respect to the control of incidents and accidents.

The research activities in this project aimed at improving the practicality and efficiency of MCDET analyses by a further standardization of user input and a further reduction of the computational effort. Furthermore, the concept of a platform independent graphical user interface (GUI) was developed. This user interface will essentially contribute to quality assurance of future MCDET analyses.

Inhaltsverzeichnis

1 Einführung 1

2 Neuer Ansatz zur Steuerung des Ablaufs von MCDET-Simulationen 5

2.1 Generelle Anforderungen 5

2.2 Anforderungen an zu koppelnde Rechenprogramme 6

2.3 Anforderungen an den Scheduling-Algorithmus 7

2.4 Realisierung des Steuerungsansatzes 9

2.5 Hintergründe zur Implementierung des Steuerungsansatzes 12

2.6 Datenverwaltung durch ‘Hashtabellen‘ 13

2.7 Erzielte Ergebnisse zum neuen Steuerungsansatz 16

3 Neue Version des Probabilistik-Moduls 19

3.1 Neue Datenstrukturen 19

3.1.1 Simulator-Variable 20

3.1.2 Zeit-Zustandsvariable 21

3.1.3 Epistemische Variable 22

3.1.4 Aleatorische Variable für die Berücksichtigung mittels Monte-Carlo-Simulation 23

3.2 Neue und abgeänderte Funktionalitäten 25

3.2.1 Angaben zu Zeit-Zustandsfenstern 25

3.2.2 Neues Zeit-Zustandsfenster zur Definition von Funktionen 29

3.2.3 Neue Option für die ausschließliche Durchführung von Monte-Carlo-Simulationen 30

3.2.4 Neue Option für Verzweigungen am Anfang eines dynamischen Ereignisbaums (DET) 33

3.3 Neue Programmroutinen MCDETinit und MCDETexe 34

4 Methoden zur Reduktion des Rechenaufwands bei Analysen mit MCDET 37

4.1 Simulation eines dynamischen Mega-Ereignisbaums 37

4.2 Definition von absorbierenden Zuständen als zusätzliche Option in der MCDET-Eingabedatei 40

4.3 Überwachung und Klassifizierung von Simulationsläufen während der Laufzeit 41

4.3.1 Hidden-Markov-Modell 42

4.3.2 Fuzzy Clustering 43

5 Benutzeroberfläche des Analysewerkzeugs MCDET 45

5.1 Menü ‘Project‘ 45

5.2 Menü ‘Variables‘ 46

5.3 Menü ‘Windows‘ 48

5.4 Menü ‘Crew‘ 52

5.4.1 Dialog zur Erstellung des ‘Key-Vektors‘ 57

5.4.2 Dialog zur Erstellung der Eingabeinformationen für relevante Alarme und Anzeigen in der Warte 64

5.4.3 Eingabeinformationen für Basishandlungen 74

5.4.4 Dialog zur Erstellung von Handlungssequenzen 88

5.4.5 Dialog zur Spezifikation von Zustandsänderungen durch Basishandlungen 96

5.5 Menü ‘Simulator Runs‘ 100

5.6 Menü ‘Post-Processing’ 101

5.7 Menü ‘Epistemic Uncertainty‘ 108

6 Zusammenfassung 111

Literaturverzeichnis 115

Abbildungsverzeichnis 121

Glossar **127**

A **Anhang: Tank-System** **129**

1 Einführung

Die probabilistische Dynamikanalyse, die mittlerweile sehr treffend als integrale deterministische probabilistische Sicherheitsanalyse (IDPSA) bezeichnet wird /ADO 11/, /ADO 12/, hat sich zu einem bedeutsamen Forschungsschwerpunkt entwickelt, auf dem international mittlerweile viele Forschungseinrichtungen tätig sind (vgl. z. B. /ALD 13/). Methoden zur Durchführung probabilistischer Dynamikanalysen lassen sich nicht nur in der Kerntechnik sondern auch in anderen Risikotechnologiebereichen (beispielsweise in der chemischen, petrochemischen und pharmazeutischen Industrie, der Luft- und Raumfahrt, im Schienen- und Straßenverkehr sowie in Einrichtungen für medizinische Behandlungen) anwenden, um Sicherheitsbeurteilungen durchzuführen.

Grundsätzlich erlauben die Methoden der probabilistischen Dynamikanalyse, für beliebige Prozessabläufe sowohl die Unsicherheiten aufgrund eines ungenauen Kenntnisstands (epistemische Unsicherheiten) als auch die Unsicherheiten aufgrund zufälliger Ereignisse (aleatorische Unsicherheiten, stochastische Unsicherheiten) zu berücksichtigen. Unsicherheiten aufgrund zufälliger Ereignisse beziehen sich z. B. auf den möglichen Ausfall einer Komponente oder eines Systems und den Zeitpunkt dieses Ausfalls. Methoden der probabilistischen Dynamikanalyse ermöglichen eine umfassende und realitätsnahe Berücksichtigung der Wechselwirkungen zwischen den unsicheren Faktoren und der Prozessdynamik. Der wesentliche Vorteil ihrer Anwendung in bei der Bewertung der Reaktorsicherheit besteht darin, dass sie es erlauben, die sich aus den Unsicherheiten ergebende Vielfalt möglicher Verläufe sicherheitstechnisch wichtiger Prozesse zu repräsentieren, zu analysieren und angemessen zu bewerten.

Die probabilistische Dynamikanalyse bzw. – treffender ausgedrückt – die IDPSA ist wie keine andere Analyse in der Lage, die zufälligen Schwankungen der Zeitpunkte, zu denen Ereignisse eintreten können, angemessen zu berücksichtigen und die entsprechenden Folgen für den Prozessablauf zu bewerten. Eine solche zeitabhängige Analyse sprengt die Möglichkeiten einer klassischen probabilistischen Sicherheitsanalyse (PSA), die aufgrund ihres statischen Charakters und der nur begrenzten Möglichkeiten der Berücksichtigung von Ergebnissen aus deterministischen Rechnungen mit vereinfachenden Modellannahmen und groben Abschätzungen arbeiten muss.

Aufgrund des zu erwartenden Nutzens einer IDPSA für Analysen Rahmen der Bewertung der Sicherheit von Reaktoren hat die GRS im Rahmen des Vorhabens RS1111 damit begonnen, die MCDET-Methode zu entwickeln /HOF 01/, /KLO 06/. MCDET

steht dabei für Monte Carlo Dynamic Event Tree und ist, wie der Name bereits ausdrückt, eine Kombination aus Monte-Carlo-Simulation und dynamischer Ereignisbaum-Methode (englisch: *d*ynamic *e*vent *t*ree, DET). Die Methode ist im gleichnamigen Analysewerkzeug MCDET implementiert, das prinzipiell in Verbindung mit jedem deterministischen Rechenprogramm zur Simulation eines dynamischen Ablaufs eingesetzt werden kann. In komplexen Analysebeispielen wurde MCDET z. B. mit den Rechenprogrammen ATHLET und MELCOR angewendet, um Abläufe ausgewählter Unfallszenarien in einem Kernkraftwerk genauer analysieren und bewerten zu können. Generell ermöglicht die Kombination von MCDET mit einem Rechenprogramm die integrale Modellierung des Anlagenverhaltens, bei dem physikalischer Prozess, technische Systeme, Personalhandlungen und zufällige Ereignisse im zeitlichen Verlauf zusammenwirken.

Veröffentlichungen zu den umfangreichen Analysebeispielen haben MCDET mittlerweile zu einem international anerkannten Analysewerkzeug gemacht. Ein zu MCDET vergleichbares Analysewerkzeug ist ADAPT /CAT 10/, das an der Ohio State University entwickelt wurde und mittlerweile in den USA bereits schon häufiger angewendet wurde /DEN 13/, /DEN 13a/, /BRU 13/. In Spanien wird für Fragestellungen der IDPSA die Methode ISA (*I*ntegrated *S*afety *A*ssessment) angewendet, die von dem Programmsystem SCAIS-TSD unterstützt wird /IZQ 94/, /MUN 99/.

Das Rechenwerkzeug MCDET besteht aus verschiedenen Modulen für unterschiedliche Funktionen. Das Probabilistik-Modul überprüft den aktuellen Zustand einer Simulationslaufs und liefert Werte für diejenigen Größen des angewendeten Rechenprogramms, die bei der Monte-Carlo-Simulation oder der Ereignisbaum-Simulation berücksichtigt werden sollen. Außerdem berechnet es die Wahrscheinlichkeiten für die Ereignisabläufe eines dynamischen Ereignisbaums. Mit dem Crew-Modul von MCDET können Personalhandlungen als dynamischer Prozess simuliert und in der Analyse berücksichtigt werden. Das Steuerungsmodul (Scheduler) kontrolliert den Ablauf von Simulationsprozessen des eingesetzten deterministischen Rechenprogramms. Das Treiber-Modul ist zuständig für die Kommunikation mit den Simulationsprozessen.

Übergeordnetes Ziel der im Rahmen des Forschungs- und Entwicklungsvorhabens RS1198 des Bundesministeriums für Wirtschaft und Energie (BMWi) durchgeführten Arbeiten war es, das Analysewerkzeug MCDET so weiterzuentwickeln, dass es zukünftig ein fester Bestandteil der verfügbaren Analysewerkzeuge für die Reaktorsicherheitsforschung wird und damit sowohl Fragestellungen im Rahmen von probabilistischen

(PSA) und deterministischen Sicherheitsanalysen (DSA) beantwortet als auch neue Nachweisverfahren für die Beherrschbarkeit von Störfall- und Unfallabläufen entwickelt werden können.

Die Arbeiten in diesem Vorhaben hatten zum Ziel, die praktische Durchführbarkeit und Effizienz von MCDET-Analysen zu verbessern. Künftige MCDET- Analysen sollen für den Benutzer möglichst einfach gestaltet und mit vertretbarem Rechenaufwand durchgeführt werden können. Deshalb wurden erforderliche Benutzereingaben weiter reduziert, vereinfacht und standardisiert sowie ein Konzept für eine plattformübergreifenden Benutzeroberfläche entwickelt. Zur Reduktion des Rechenaufwands wurde ein neuer Algorithmus für die Konstruktion dynamischer Ereignisbäume sowie ein neuer Ansatz für die Steuerung von Simulationsprozessen entwickelt.

2 Neuer Ansatz zur Steuerung des Ablaufs von MCDET-Simulationen

Simulationen mit MCDET in Verbindung mit einem deterministischen Rechenprogramm erzeugen eine Stichprobe dynamischer Ereignisbäume mit einer Vielzahl von unterschiedlichen Störfall- bzw. Unfallabläufen. Abhängig von Art und Anzahl der (aufgrund von Unsicherheiten) zu variierenden Eingabegrößen können sich eine sehr große Zahl von Verzweigungspunkten in einem dynamischen Ereignisbaum und dementsprechend viele Simulationsläufe ergeben. Daraus entsteht die Notwendigkeit, für einen möglichst effizienten Ablauf zum Berechnen der Sequenzen aller dynamischen Ereignisbäume zu sorgen.

Eine Aufgabe der Forschungs- und Entwicklungsarbeiten war die Erarbeitung eines neuen Konzepts zur effizienten Kopplung von MCDET mit deterministischen Rechenprogrammen und die Implementierung eines Prototyps, der die Arbeitsweise einer Kopplung beispielhaft realisiert. Die Abschnitte 2.1 – 2.7 beschreiben die dafür zu Grunde gelegten Anforderungen und Lösungsansätze.

2.1 Generelle Anforderungen

Für Modularisierung und Ablauf ergaben sich die nachfolgenden Anforderungen, die auf bereits durchgeführte Implementierungen sowie Realisierbarkeit und Widerspruchsfreiheit untersucht wurden:

- Parallele Berechnung von Ereignispfaden:
  Die Berechnung voneinander unabhängiger Ereignispfade kann von gleichzeitig gestarteten Simulationsprozessen durchgeführt werden. Diese Prozesse können auf die verfügbaren Recheneinheiten verteilt werden, um die Rechenkapazität von Mehrkern-Systemen zu nutzen und so die Gesamtrechenzeit drastisch zu reduzieren.

- Dynamische Ermittlung von alternativen Ereignispfaden:
  Die Notwendigkeit der Berechnung von alternativen Ereignispfaden wird bereits rein regelbasiert (durch die implementierten Zeit-Zustandsfenster, vgl. Abschnitt 3.2) bestimmt. Dadurch können die von MCDET durchzuführenden Aktionen direkt anhand aktuell im simulierten Ereignisablauf vorherrschender Bedingungen getroffen werden.

- Kopplung von MCDET mit verschiedenen Rechenprogrammen:
  Ein Steuerungssystem, welches die Verwaltung der Simulationsprozesse übernimmt, muss eine Reihe verschiedener Rechenprogramme ansteuern können. Hierzu ist eine einheitliche, sogenannte Simulator-Schnittstelle für Steuerung und Datenaustausch notwendig. Alle von MCDET benötigten Größen und Signale eines Simulationslaufs müssen darüber zugänglich sein. Die Anforderungen an diese Schnittstelle werden im Abschnitt 2.2 beschrieben.

- Datenaustausch mit den Prozessen eines Rechenprogramms über Inter-Prozess-Kommunikation (IPC):
  Die zentrale Steuerung der Prozesse eines Rechenprogramms sollte nicht über einen dateibasierten Datenaustausch sondern über synchronisierbare Netzwerkverbindungen erfolgen. Dadurch wird sowohl die I/O-Last (Eingabe/Ausgabe-Last) durch dafür notwendige Festplattenzugriffe reduziert als auch die Verwaltung entsprechender Steuerdateien vermieden.

- Vermeidung von zwischengespeicherten Restartpunkten:
  Die Verwaltung der Ereignispfade erfolgt durch den Scheduler, der die Rechenreihenfolge der Ereignispfade automatisch anhand der Prozess-Hierarchie bestimmt. Dadurch wird die Stapel-basierte Speicherung von Restartpunkten unnötig, wodurch die verursachte I/O-Last entsprechend abnimmt.

- Nutzen von gewöhnlichen Eingabespezifikationen:
  Die Eingabedateien des verwendeten Rechenprogramms müssen nicht speziell für die Kopplung mit MCDET angepasst werden. Dadurch erleichtert sich die Pflege der Eingabedateien erheblich, Unstimmigkeiten zwischen verschiedenen Versionen werden vermieden.

- Plattformunabhängigkeit:
  MCDET in Verbindung mit einem deterministischen Rechenprogramm sollte praktisch ohne Anpassungen auf verschiedenen Rechnersystemen lauffähig sein. Hierzu zählen vor allem Desktop- und Cluster-Systeme unter Linux als auch unter Windows.

2.2 Anforderungen an zu koppelnde Rechenprogramme

In den generellen Anforderungen an das Steuerungssystem wurde bereits die Notwendigkeit einer einheitlichen Simulator-Schnittstelle für koppelbare Rechenprogramme

festgestellt. Allgemein entsteht eine erfolgreiche Kopplung von Programmteilen erst durch ein koordiniertes Zusammenspiel von gegenseitiger Steuerung und Datenaustausch. Um dieses zu ermöglichen, muss ein Rechenprogramm für die Kopplung mit MCDET einige Voraussetzungen erfüllen:

- Taktsteuerung:
  Das Rechenprogramm teilt die simulierte Zeit in diskrete Schritte ein und berechnet die einzelnen Zeitschritte entsprechend eines von außen vorgegebenen Takts (tick-Befehl).

- Statusangabe:
  Der Status eines Simulationslaufs mit dem Rechenprogramm kann von außen abgefragt werden. Dieser gibt Auskunft darüber, ob das Rechenprogramm gerade einen Zeitschritt berechnet (busy) oder auf einen neuen Befehl wartet (idle / ready).

- Datenzugriff:
  Das Rechenprogramm stellt den Zustand eines Simulationslaufs, d. h. sämtliche für die Kopplung notwendigen Signale und Größen, bereit und erlaubt auch, diese für die weiteren Berechnungen zu verändern.

- Klonfähigkeit:
  Das Rechenprogramm bietet für die Behandlung von Verzweigungspunkten eine Möglichkeit, eine laufende Simulation zu duplizieren (fork). Eine Methode hierfür ist die Speicherung des aktuellen Systemzustandes (Restartpunkt) und die Erstellung eines neuen Simulationsprozesses, der die Berechnung an diesem Punkt fortführt.

- Terminierung:
  Das Rechenprogramm bietet für den Fall, dass die Eintrittswahrscheinlichkeit einer Ereignissequenz unter einen vom Benutzer definierten Schwellenwert fällt, die Möglichkeit, eine laufende Simulation explizit zu beenden. Diese Möglichkeit der Terminierung kann vom Benutzer optional eingesetzt werden.

2.3 Anforderungen an den Scheduling-Algorithmus

Der Begriff ‘Scheduling‘ bezeichnet das Vorgehen bei der zeitlichen Einteilung von Simulations-Rechenschritten. Hierbei muss sowohl die Reihenfolge als auch die optionale Verteilung auf mehrere Recheneinheiten festgelegt werden, um eine reibungslose Berechnung zu ermöglichen. Besonders das Ziel einer performanten (d. h. leistungsfä-

higen) Parallelisierung von Simulationspfaden (Ereignispfaden) stellt einige Anforderungen an den Scheduling-Algorithmus:

- Kein ‘Verhungern‘ von Simulationsprozessen:
  Alle Simulationspfade müssen verfolgt werden, bis das von MCDET vorgegebene Abbruchkriterium erfüllt ist. Daher muss für alle Simulationsprozesse gewährleistet werden, dass kein Prozess unendlich auf den Takt (‘tick-Befehl‘) warten muss, weil dieser vom Scheduler ‘vergessen‘ wurde.

- Bestmögliche Nutzung verfügbarer Recheneinheiten:
  Simulationsprozesse, die bereit sind, den jeweilig nächsten Zeitschritt zu rechnen, sollten bei verfügbaren Recheneinheiten nicht unnötig auf den Takt (tick-Befehl) warten müssen.

- Konfigurierbarer Parallelisierungsgrad:
  Je nach Nutzung des Rechnersystems ist eine Auslastung aller verfügbaren Recheneinheiten nicht unbedingt gewünscht. Um eine gewisse Reaktionszeit zu gewährleisten, ist es sinnvoll, das System nicht voll auszulasten, indem die Anzahl der gleichzeitig rechnenden Simulationen begrenzt wird.

- Minimierung der Anzahl inaktiver Prozesse:
  Auch der Verbrauch des Arbeitsspeichers ist eine wichtige Größe, die im Auge behalten werden sollte, um die Systemlast zu begrenzen und die Zuverlässigkeit bei der Berechnung von dynamischen Ereignisbäumen zu erhöhen. Besonders bei Systemen für mehrere Nutzer ist dies relevant, da Prozesse, für die nicht mehr ausreichend Speicher zur Verfügung steht, vom Betriebssystem automatisch beendet werden. Deshalb ist es für den Scheduling-Algorithmus sehr wichtig, die Anzahl an inaktiven Simulationen durch ein kontrolliertes Erstellen von Rechenprozessen zu minimieren.

- Priorisierung der Berechnung lokaler Teilbäume:
  Gerade im Hinblick auf die Speicherkapazität ist es nicht unerheblich, welche Simulationsprozesse parallel rechnen. Besonders Prozesse, die an Verzweigungspunkten auf die Erstellung ihrer sogenannten ‘Kindprozesse‘ warten müssen, können die Anzahl an inaktiven Simulationsprozessen im Speicher anwachsen lassen. Für die Zuteilung von Rechentakten erweisen sich dabei bekannte Scheduling-Strategien, wie ‘First In-First Out‘ oder ‘Round Robin‘ /SIL 03/, als wenig geeignet. Vielmehr ist eine dynamische Priorisierung sinnvoll, welche Prozesse lokale Teilbäume bevorzugt behandelt.

2.4 Realisierung des Steuerungsansatzes

Um MCDET an verschiedene Rechenprogramme anbinden zu können, wurde an der bisherigen modularen Aufteilung der MCDET-Programmteile mit einigen Änderungen festgehalten. Das Scheduler-Modul übernimmt dabei die Steuerung des Rechenprogramms sowie die Verwaltung der Simulationsprozesse und die Organisation des Datenaustauschs.

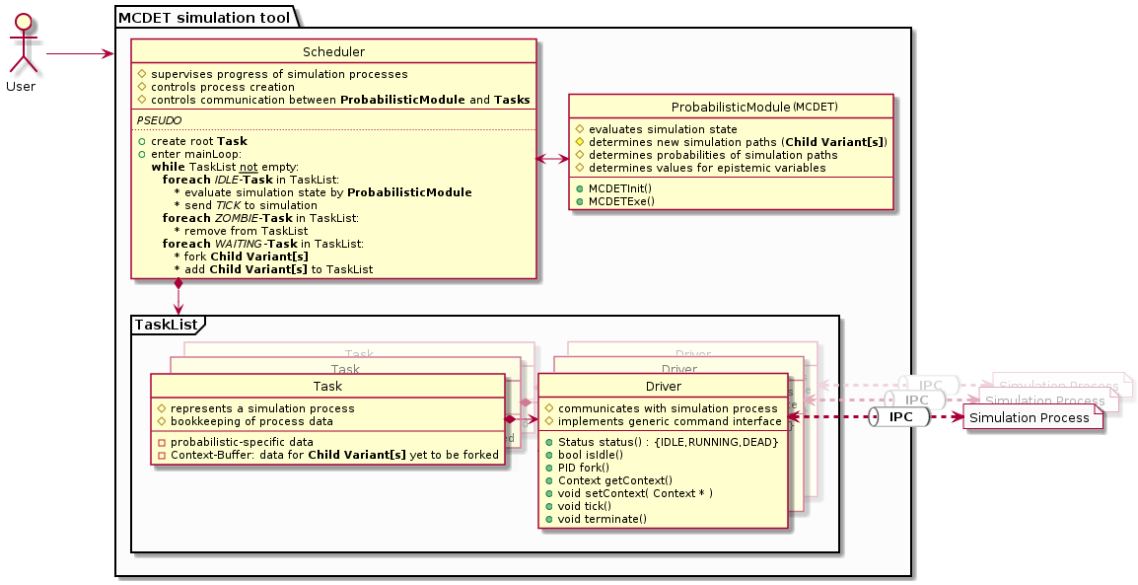


Abb. 2.1 MCDET-Module für die Kopplung mit einem deterministischen Rechenprogramm und deren Zusammenspiel

Abb. 2.1 gibt einen Überblick über die MCDET Module, die für die Kopplung mit einem deterministischen Rechenprogramm benötigt werden. Die einzelnen Module sind für folgende Aufgaben zuständig:

- Steuerungsmodul (Scheduler):
  Dieser Programmteil startet und taktet alle Simulationen, kontrolliert den Datenaustausch und synchronisiert alle Rechenläufe mit dem Probabilistik-Modul.

- Probabilistik-Modul:
  Das im Probabilistik-Modul implementierte Verfahren beurteilt nach jedem Zeitschritt den Zustand eines Rechenlaufs, entscheidet über den weiteren Verlauf, d. h. ob und wenn ja welche Aktionen zur Generierung eines dynamischen Ereignisbaums durchgeführt werden sollen.

- Treiber-Modul:
  Dieser Programmteil abstrahiert durch generische Kommando-Schnittstellen von den konkret eingesetzten Rechenprogrammen und vereinfacht den Scheduler durch eine einheitliche Ansteuerung der Simulationsprozesse des jeweiligen Rechenprogramms.

Der grundsätzliche Ablauf für die Berechnung von dynamischen Ereignisbäumen wird durch Abb. 2.2 verdeutlicht und lässt sich durch folgende Schritte beschreiben:

1. Der Benutzer erstellt die für die Simulationen notwendigen Eingabedateien von MCDET und dem Rechenprogramm und startet den Scheduler (a).

2. Der Scheduler startet die Berechnung eines Ereignispfades durch die Erstellung eines Simulationsprozesses (in Abb. 2.2 als ‘Simulator‘ bezeichnet (b), der als Task verwaltet und anschließend der Taskliste hinzugefügt (c) wird. Die Berechnung wird vom Rechenprogramm durch Zeitschritte (ticks) getaktet diskretisiert. Dieses ist in Abb. 2.2 dadurch erkennbar, dass das Rechenprogramm nach asynchron durchgeführtem Start und Initialisierung auf einen Befehl wartet (d). In einem weiteren Schritt lädt und initialisiert der Scheduler das Probabilistik-Modul, welches mit Hilfe eines sogenannten ‘Estimators‘ verwaltet wird (e). Anschließend tritt der Scheduler in die Hauptschleife (f) ein, die solange durchlaufen wird, bis alle Tasks abgearbeitet sind und die Taskliste leer ist.

3. Die Hauptschleife beginnt mit dem Durchsuchen der Taskliste nach Tasks, die bereit (ready-Zustand) sind einen Zeitschritt zu rechnen. Wird keine entsprechende Task gefunden, z. B. weil alle zugehörigen Simulationsprozesse mit der Berechnung eines Zeitschrittes beschäftigt sind, wartet die Hauptschleife solange, bis eine Task diesen Zustand erreicht (g).

4. Die ausgewählte und rechenbereite Task wird durch einen Befehl dazu aufgefordert ihren Zustand (Context) mitzuteilen, welcher alle von MCDET benötigten Prozess- und Systemgrößen (z. B. in Form von Signalen des sogenannten ‘*G*eneral *C*ontrol *S*imulation‘ *M*oduls, GCSM) enthält (h). Der aktuelle Zustand dieser Größen, die in der MCDET-Eingabedatei als Simulator-Variable bezeichnet werden, wird direkt an das Probabilistik-Modul zur Auswertung weitergeleitet.

5. Das Probabilistik-Modul wertet den aktuellen Zustand der Simulator-Variablen und anderer MCDET spezifischer Größen aus (i) und teilt dem Scheduler mit, wie mit der zugehörigen Task weiter verfahren werden soll: Terminierung (z. B. wegen Un-

terschreitung eines Schwellwertes), Fortsetzung (d. h. keine Veränderung und Ausführen des nächsten Zeitschritts) oder Verzweigung (d. h. Erzeugung eines oder mehrerer alternativer Rechenpfade (Variants)).

6. Der Scheduler setzt die vom Probabilistik-Modul erhaltene Auswertung der Task um:

   - Terminierung:
     Die Task wird durch einen ‘terminate-Befehl‘ zum Beenden der Simulation aufgefordert und anschließend aus der Taskliste entfernt (j).

   - Fortsetzung:
     Die Task wird durch einen tick-Befehl zum Berechnen des nächsten Zeitschritts aufgefordert (k). Der zugehörige Simulationsprozess setzt daraufhin die asynchrone Berechnung der Simulation fort und wartet nach Beendigung des Zeitschritts wiederum auf einen neuen Befehl.

   - Verzweigung:
     Für jeden alternativen Ereignispfad (Variante) teilt das Probabilistik-Modul die zu verändernden Größen dem Scheduler in Form eines sogenannten ‘ChangeSets‘ mit. Alle Varianten werden durch eigenständige Simulationsprozesse berechnet, welche durch Klonen der ursprünglichen Simulation erzeugt werden. Hierzu fordert der Scheduler die Task jeweils durch einen ‘fork-Befehl‘ zum Duplizieren ihrer Simulation auf, teilt der dadurch erstellten Task die zu verändernden Größen mit und fügt sie zur Taskliste hinzu (l).

7. Die Ausführung des Schedulers wird am Beginn der Hauptschleife fortgesetzt.

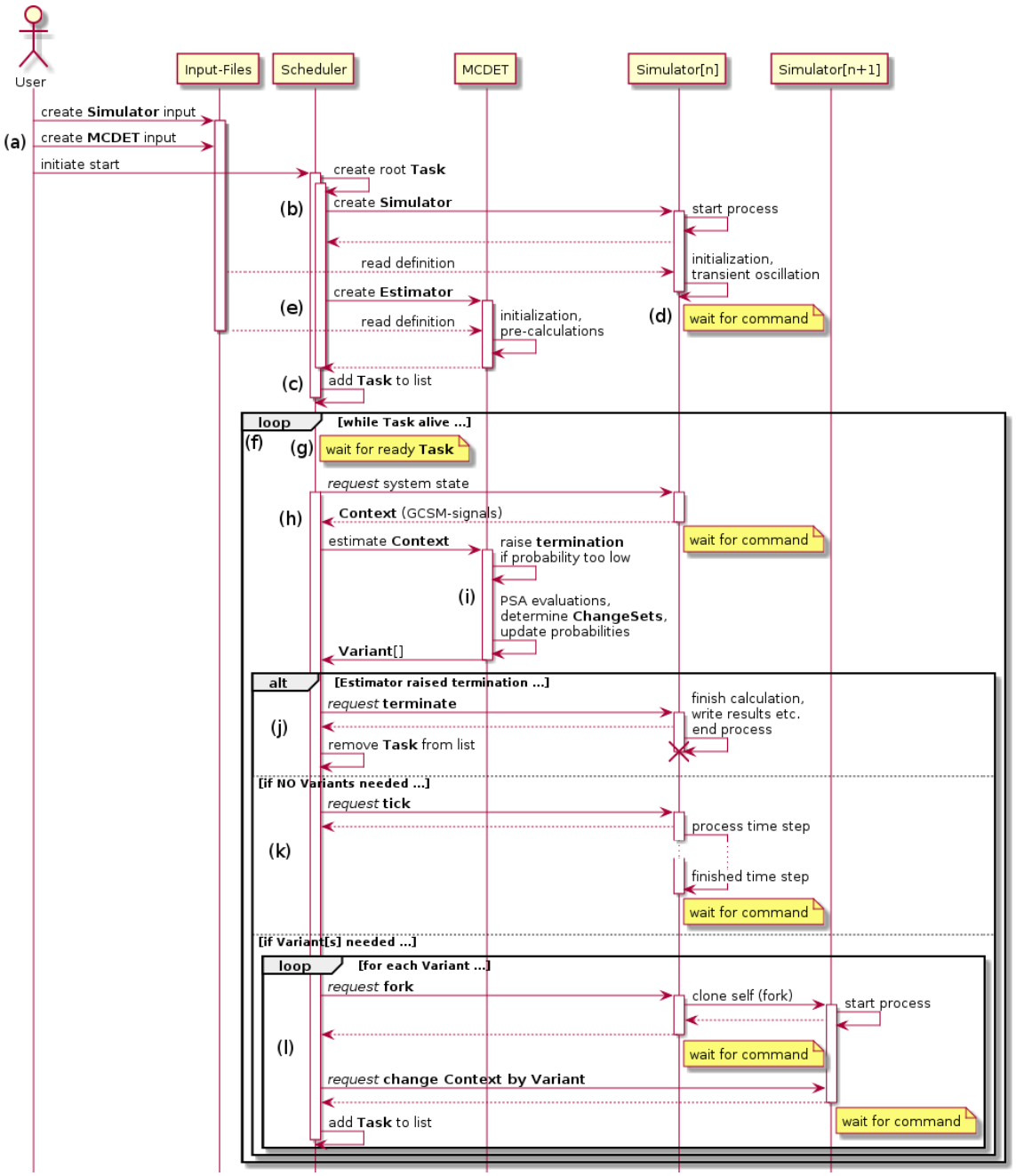


Abb. 2.2 Ablauf der Berechnung eines dynamischen Ereignisbaumes

2.5 Hintergründe zur Implementierung des Steuerungsansatzes

Die Implementierung des Scheduling-Systems erforderte die Nutzung diverser Techniken zur Prozess-Verwaltung und -Kommunikation. Hieraus resultierte die Entscheidung, Python als Programmiersprache für die Realisierung des Scheduling-Moduls zu verwenden. Als universell einsetzbare Sprache, die durch ihre hohe Verfügbarkeit mittlerweile sehr weit verbreitet ist, unterstützt Python bereits viele dieser Techniken, die in

anderen Sprachen, z. B. FORTRAN, nur sehr mühsam umgesetzt werden könnten. Auch die hervorragende Unterstützung zur Einbindung von Code-Modulen, die in anderen Sprachen entwickelt wurden, war ein wichtiges Argument dafür, das zentrale Steuersystem in Python umzusetzen.

Der Algorithmus der MCDET-Methode wurde als eigenständiges Probabilistik-Modul in FORTRAN implementiert und wird auch weiterhin als solches gewartet und weiterentwickelt. Für die Einbindung dieses Moduls in das Scheduling-System ist eine Kommunikationsschnittstelle notwendig, die die Berechnungsfunktionen des Probabilistik-Moduls steuern kann und einen effizienten Datenaustausch ermöglicht. Hierfür wurde eine Kommunikationsbibliothek entwickelt, die es erlaubt, aus beiden Sprachen, FORTRAN und Python, auf dieselben Daten lesend und schreibend zuzugreifen und diese zu verwalten. Mittels der hierbei eingesetzten Datenstrukturen können sämtliche Variable, die die Grundlage für die Berechnungen von MCDET bilden, bereit gehalten und bei Bedarf verändert werden.

Die Berechnung von Ereignispfaden wird von unabhängig ausgeführten Simulationsprozessen (z. B. des deterministischen Rechenprogramms ATHLET) übernommen. Damit das Scheduling-System diese eigenständigen Prozesse steuern und beeinflussen kann, ist es notwendig, mit diesen in Verbindung zu bleiben und Daten austauschen zu können. Hierzu wurde eine Analyse der in ATHLET bestehenden Mechanismen zur Steuerung und zum Datenaustausch mit externen Anwendungen durchgeführt.

Am Beispiel der Online-Visualisierung in ATLAS /VOG 06/ konnten die Voraussetzungen für eine Laufzeitkopplung an ATHLET verifiziert werden. Durch diese Untersuchung wurde klar, dass alle für eine Kopplung mit MCDET notwendigen Steuerungsmechanismen in ATHLET grundsätzlich vorhanden sind. Allerdings müssen einige Teile dieser Schnittstelle überarbeitet werden, um eine Kopplung auch skalierbar, d. h. für eine große Anzahl an Simulationsprozessen, betreiben zu können. Für die Änderungen, die hierfür an ATHLET vorgenommen werden müssen, wurde ein Konzept erarbeitet und in Form eines Prototyps implementiert.

2.6 Datenverwaltung durch ‘Hashtabellen‘

Das Probabilistik-Modul von MCDET kennt weder die Anzahl der während der Laufzeit entstehenden Simulationsprozesse noch deren Reihenfolge, in der es deren Zustände

zu bewerten hat. Während der Abarbeitung darf es natürlich trotzdem nicht zu einer Vermischung von Daten der einzelnen Simulationsprozesse kommen. Daher ist es sehr wichtig, dass das Probabilistik-Modul keine dieser Daten intern zwischenspeichert sondern Werte mit eindeutigem Bezug zu einem Simulationsprozess geeignet ablegen kann. Eine entsprechend leistungsfähige Verwaltung solcher Daten innerhalb des Probabilistik-Moduls zu realisieren, würde einen erheblichen Aufwand bedeuten und die spezifischen Algorithmen unnötig verkomplizieren.

Eine ähnliche Problemstellung ergibt sich für den Scheduler: Dieser muss für eine Zustandsbewertung durch das Probabilistik-Modul eine aktuelle Momentaufnahme aller Prozess- und Systemgrößen auf einmal zur Verfügung stellen. Die Anzahl dieser Größen kann für realistische Simulationen schnell sehr groß werden, was es erforderlich macht, die Daten in einer geeignet organisierten Struktur, dem sogenannten Context (siehe Abschnitt 2.4), bereitzustellen.

Der Zugriff auf einzelne Werte im Context muss einfach, eindeutig und effizient sein. Das Probabilistik-Modul kennt auch hier weder die Anzahl noch die Reihenfolge der Einträge, wodurch ein indexbasiertes Zugriffsverfahren ausscheidet (Näheres zu indexbasierten Zugriffsverfahren in /OTT 02/). Vielmehr müssen Prozess- und Systemgrößen mit symbolischem Namen angesprochen werden können. Erfüllt die für den Context gewählte Datenstruktur alle diese Anforderungen, kann diese zusätzlich zur Ablage der spezifischen Daten für einen Simulationsprozess vom Probabilistik-Modul genutzt werden. Damit lässt sich auch das Problem der Zuordnung dieser Daten zum entsprechenden Simulationsprozess lösen.

Um all diesen Ansprüchen gerecht zu werden, wurde der Kontext als sogenannte ‘Hashtabelle‘ (oder ‘Hashmap‘) realisiert /OTT 02/. Die auch als ‘Assoziative Arrays‘ bekannten Hashtabellen sind besonders geeignet, um große Datenmengen zu verwalten, und zeichnen sich besonders durch eine normalerweise konstante Zugriffszeit aus. Dadurch ist der Zeitaufwand, um Werte zu lesen oder zu schreiben, praktisch unabhängig von der Anzahl der verwalteten Elemente.

Die in Hashtabellen abgelegten Elemente werden durch einen eindeutigen Schlüssel identifiziert. In den meisten Fällen wird hierfür ein String mit einem symbolischen Namen verwendet. Der Ablagealgorithmus basiert auf der Idee, den Schlüssel eines Elements durch eine Hashfunktion in einen eindeutigen, möglichst kollisionsfreien Hashwert umzurechnen. Dieser Hashwert kann dann auf den Indexbereich einer Ablageta-

belle (Indextabelle) abgebildet werden und identifiziert damit die Position, den sogenannten ‘Bucket‘ der Tabelle, an dem das entsprechende Element abgelegt wird.

Idealerweise würde also jeder mögliche Schlüssel eindeutig (injektiv) auf einen Bucket abgebildet. Die Indextabelle müsste hierfür eine Größe aufweisen, die der Anzahl aller möglichen Schlüssel entspricht, was aufgrund begrenzten Speicherplatzes praktisch nicht möglich ist. Der berechnete Hashwert eines Schlüssels wird deshalb erst durch Anwendung des Modulo (Division mit Rest) auf die Größe der Indextabelle begrenzt.

In der Praxis wird die Indextabelle meist anwendungsabhängig oder in einer benutzerdefinierten Größe, oft in der Größenordnung von 100, angelegt. Hier lassen sich also Situationen, sogenannte Kollisionen, in denen mehrere unterschiedliche Schlüssel demselben Bucket zugeordnet werden, nicht vermeiden. Darum muss eine Hashtabelle auf Kollisionen vorbereitet sein. Es gibt verschiedene Algorithmen, wie lineares / quadratisches Sondieren, Doppel-Hashing oder Verkettung, um solche Kollisionen aufzulösen.

Weitere Details zu Hashtabellen findet man in /OTT 02/.

Die als Context eingesetzte Hashtabelle wurde mit den folgenden Eigenschaften implementiert:

- Zugriff auf abgelegte Elemente über symbolischen Namen,
- direkter Zugriff auf die Daten aus Python und FORTRAN,
- leichte Duplizierbarkeit,
- Indextabelle mit benutzerdefinierter Größe oder automatischer Größenanpassung für optimierten Zugriff,
- Kollisionsbehandlung durch Verkettung (‘separate chaining‘).

Alle im Rechenprogramm und im Probabilistik-Modul vorkommenden Datentypen können als Werte abgelegt werden. Unter anderem werden hierfür Integer-, Real-, Logical-Werte, Zeichenketten und Arrays unterschiedlicher Dimension unterstützt.

2.7 Erzielte Ergebnisse zum neuen Steuerungsansatz

Die implementierten Programmteile erlauben bereits die parallele Ausführung einer Mockup-Version des Rechenprogramms bzw. des Probabilistik-Moduls. Die Simulationsläufe können auf verschiedene Weise visualisiert. Die für einzelne Ereignispfade verwendete Rechenzeit kann so abgeschätzt werden. Auch können abnormal beendete Rechnungen, z. B. durch Absturz der Simulationsrechnung, erkannt werden. Dies ermöglicht die Erkennung von Problemen und erlaubt, je nach Fall zu entscheiden, ob ein abgebrochener Teilbaum als eigene Berechnung neu gestartet werden soll.

Eine Variante für die Visualisierung von Ereignispfaden ist das UML-Sequenzdiagramm in Abb. 2.3, das einen beispielhaften Ablauf von Simulationsprozessen darstellt. Hier geben die Nummern in den gelben Kästchen (a) die Prozess-IDs der Simulationsprozesse an. Der Lebenszyklus eines Simulationsprozesses beginnt mit einem fork-Signalpfeil auf seine Prozess-ID, verläuft entlang der gestrichelten Lebenslinie und endet mit seiner Zerstörung, die durch ein rotes Kreuz symbolisiert wird (b). Die aktive Rechenzeit eines Simulationsprozesses, d. h. die Zeit, in der ein Zeitschritt berechnet wird, zeigt das Diagramm jeweils durch einen vertikalen Balken (c) und ein tick-Signal auf dessen Lebenslinie an.

Für dieses Beispiel wurde die parallele Abarbeitung auf vier Prozesse beschränkt. Des Weiteren wurden zwei Simulationsprozesse per Task-Manager beendet um Programmabstürze zu simulieren. Dieses wurde vom Scheduler erkannt und im Diagramm mit der Markierung ‘sudden death‘ (d) gekennzeichnet.

- Bei der Darstellung in Abb. 2.3 ist zu beachten, dass die vertikale Achse durch das Layout zufällig gedehnt ist, wodurch leider nur ein qualitativer Vergleich der Lebenszyklen und Rechenzeiten zwischen den Simulationsprozessen möglich ist und keine Schlüsse über die absolute Dauer einzelner Phasen gezogen werden können. Dennoch wird aus der Darstellung bereits ersichtlich, dass:
- alle Simulationspfade (ausgenommen, die aktiv beendeten) bis zum Ende gerechnet werden: Alle Lebenslinien enden mit einem roten Kreuz;
- eine gute parallele Nutzung von Recheneinheiten erreicht wird: Es sind immer wieder mehrere Prozesse gleichzeitig (auf einer Höhe) durch einen vertikalen Balken als aktiv markiert, z. B. (e);

- die Beschränkung für das parallele Ausführen der Simulationsprozesse beachtet wird: Es sind maximal vier Prozesse gleichzeitig durch einen vertikalen Balken als aktiv markiert, siehe (e).

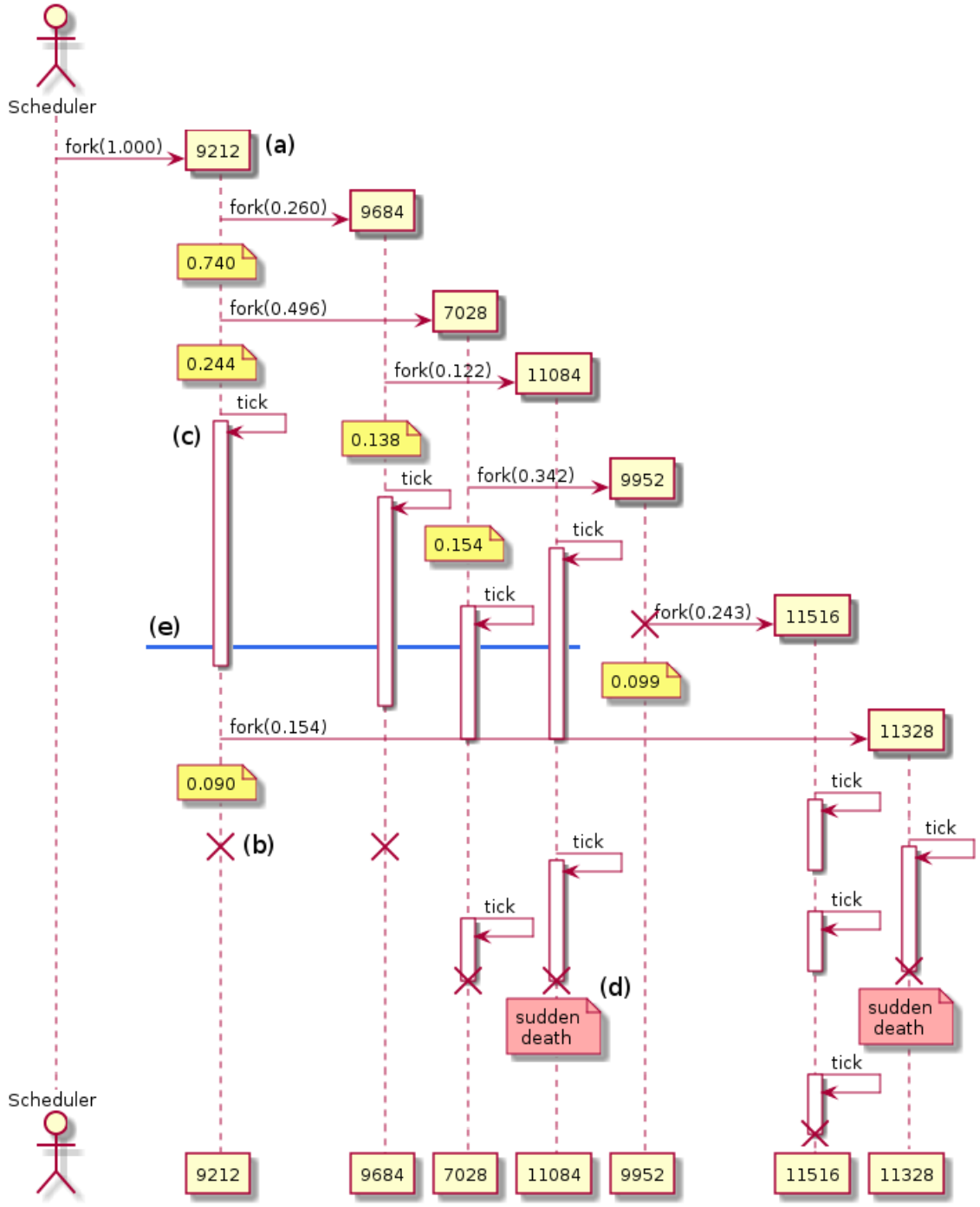


Abb. 2.3 Prozesssequenz bei der Berechnung eines dynamischen Ereignisbaums

3 Neue Version des Probabilistik-Moduls

Das Probabilistik-Modul des MCDET-Analysewerkzeugs überprüft den aktuellen Zustand eines vom Steuerungsmodul ausgewählten Simulationsprozesses und wertet diesen daraufhin aus, ob er die Bedingungen für weitere Programmschritte des Probabilistik-Moduls erfüllt oder nicht. Wesentliche Aufgaben des Probabilistik-Moduls sind:

- Ermitteln und Bereitstellen der Anfangszustände von neuen Simulationspfaden, die ab einem Verzweigungspunkt im dynamischen Ereignisbaum als Alternative zum laufenden Simulationspfad berechnet werden sollen;
- berechnen und Bereitstellen der Eintrittswahrscheinlichkeiten von neuen Simulationspfaden und Ermittlung der Wahrscheinlichkeiten vollständiger Simulationssequenzen;
- ausspielen von Werten sowohl für aleatorische als auch für epistemische Variable und Bereitstellen der ausgespielten Werte zur Berücksichtigung in der Monte-Carlo-Simulation. (Aleatorische Variable sind Parameter, deren Werte aufgrund stochastischer Einflussfaktoren variieren und die deshalb mit einer aleatorischen Unsicherheit verbunden sind. Epistemische Variable sind Parameter, deren Werte fest aber nur ungenau bekannt sind und die deshalb mit einer epistemischen Unsicherheit verbunden sind.).

3.1 Neue Datenstrukturen

Durch die Veränderung des Ablaufkonzepts für MCDET-Simulationen (siehe Kapitel 2) musste das in RS1111 entwickelte Probabilistik-Modul /HOF 01/, dessen Algorithmus auf einer sequentiellen Berechnung der Sequenzen eines dynamischen Ereignisbaums (DET) nach dem ‘last-in-first-out‘ (LIFO) Prinzip basierte, für die in diesem Vorhaben möglich gemachte parallele Berechnung der Sequenzen entsprechend angepasst werden. U. a. mussten Datenstrukturen (z. B. skalare Variable und Datenfelder) implementiert werden, die eine Kommunikation mit dem neuen Scheduler-Modul und das Durchführen paralleler Simulationsläufe ohne Konflikte bei der Datenübertragung erlauben. Diese Datenstrukturen beziehen sich insbesondere auf Größen, die für den Datentransfer zwischen Probabilistik-Modul und den einzelnen Simulationsprozessen des eingesetzten Rechenprogramms benötigt werden. Zu diesen Größen zählen die Prozessgrößen des Rechenprogramms sowie MCDET-spezifische Größen, wie z. B. die Zeit-Zustandsvariablen (siehe Abschnitt 3.1.2), die Nummer des Ereignisbaums, zu

dem eine Simulationssequenz gehört, oder die Wahrscheinlichkeit einer Simulationssequenz. Durch Implementierung sogenannter Hashtabellen wurde eine einfache und effiziente (Rechenzeit sparende) Datenverwaltung zur Kommunikation zwischen Probabilistik-Modul und den Simulationsprozessen ermöglicht (siehe Kapitel 2).

3.1.1 Simulator-Variable

Für den Datenaustausch zwischen Probabilistik-Modul und den Simulationsprozessen des verwendeten Rechenprogramms stehen sogenannte Simulator-Variablen zur Verfügung. Diese werden vor Beginn der Simulationsläufe in der MCDET-Eingabedatei jeweils mit ihrem Namen und einem Anfangswert initialisiert. Die jeweiligen Namen sollten dabei mit den im Rechenprogramm verwendeten Namen für die betreffenden Größen übereinstimmen. Die Werte aller Simulator-Variablen eines Simulationsprozesses werden in einer prozessspezifischen Hashtabelle abgelegt. Jede abgelegte Simulator-Variable wird durch ihren Namen eindeutig identifiziert. Damit sind alle Variablenwerte eines Simulationsprozesses unmittelbar verfügbar, ohne dass sie zu Lasten der Rechenzeit auf entsprechende Dateien geschrieben und von dort wieder gelesen werden müssen.

```
@*****************************************************************************
Simulator variables
@*****************************************************************************
@
@ Number of simulator variables
    5
@ List of simulator variables
@
@ Name          Initial Value
    Time                0
    FillLevel           5
    Valve               0
    Pump1               1
    Pump2               0
```

Abb. 3.1 MCDET-Eingabedatei: Eingabe von Simulator-Variablen

Wie Simulator-Variablen in der MCDET Eingabedatei spezifiziert werden, zeigt Abb. 3.1. Bei dem in der Abbildung behandelten Beispiel geht es um das im Anhang detailliert beschriebene Tanksystem. Der Zulauf des Tanks erfolgt über zwei Pumpen (0: Pumpe steht, 1: Pumpe läuft); der Ablauf wird über ein Ventil (0: Ventil offen, 1: Ventil geschlossen) sichergestellt. Zu Beginn der Simulation läuft Pumpe 1 (Pump1),

Pumpe 2 (Pump2) steht und das Ventil (Valve) ist offen. Der Füllstand (FillLevel) im Tank ist 5 m. Die Zeit (Time) beginnt bei 0 Sekunden.

3.1.2 Zeit-Zustandsvariable

Neben den Simulator-Variablen können optional sogenannte Zeit-Zustandsvariable definiert werden. Das sind solche Variablen, die im Rechenprogramm nicht verwendet werden, die aber für die Modellierung von zufälligen Ereignissen mittels der Zeit-Zustandsfenster in der MCDET-Eingabedatei (siehe Abschnitt 3.2.1) hilfreich sein können. Zeit-Zustandsvariable können sich in Abhängigkeit vom Simulationsprozess oder von zufälligen Einflussfaktoren ändern. Beispiele sind:

- die Zahl der Anforderungszyklen (Öffnen bzw. Schließen) eines Druckhalter-Ablaseventils bzw. Sicherheitsventils während der automatischen Druckentlastung, wobei das Ventil bei einem seiner Anforderungszyklen zufällig versagen kann,
- die Indikatorvariable für Spannung vorhanden/nicht vorhanden in einem Szenario mit Ausfall der Spannungsversorgung und Spannungswiederkehr zu einem zufälligen Zeitpunkt.

Die Spezifikation der Zeit-Zustandsvariablen in der MCDET-Eingabedatei ist vergleichbar mit der Spezifikation der Simulator-Variablen. Bei dem in Abb. 3.2 aufgeführten Beispiel bezeichnet *‘nValveOpen‘* die Zahl der Anforderungszyklen (Öffnen) des Ventils aus dem Tankbeispiel. Zu Beginn der Simulation ist die Zahl gleich 0. Jedes Mal, wenn das Ventil im Verlauf eines Simulationslaufs angefordert wird zu ‘Öffnen‘, wird ‘nValveOpen‘ um 1 hochgezählt. Die entsprechende Spezifikation dafür erfolgt in sogenannten ‘Function Windows‘ (siehe Abschnitt 3.2.2). Mit Hilfe der Zeit-Zustandsvariable ‘nValveOpen‘ kann z. B. modelliert werden, dass das Ventil nicht öffnet, wenn ‘nValveOpen‘ = nv, wobei ‘nv‘ eine aleatorische Variable ist, die von MCDET ausgespielt wird. Der Zugriff auf die prozessspezifischen Werte der Zeit-Zustandsvariablen erfolgt mittels Hashtabellen.

```
@*****************************************************************************
Time/state variables (additional variables used for MCDET simulations)
@*****************************************************************************
@ Number of time/state variables
    1
@ List of time/state variables
@
@ Name          Initial Value
   nValveOpen            0
```

Abb. 3.2 MCDET-Eingabedatei: Eingabe von Zeit-Zustandsvariablen

3.1.3 Epistemische Variable

Epistemische Variable sind Parameter, deren Werte fest aber nur ungenau bekannt sind und die deshalb mit einer epistemischen (d. h. Kenntnisstand-) Unsicherheit verbunden sind. Alle epistemischen Variablen einer MCDET-Analyse werden mittels Monte-Carlo-Simulation berücksichtigt. Ihre Werte werden vor Beginn der Simulationsläufe aus einer Datei eingelesen und stehen dann sämtlichen Simulationsprozessen zur Verfügung. Die Werte der epistemischen Variablen werden nicht in Hashtabellen abgelegt.

Eine beispielhafte Spezifikation von epistemischen Variablen in der MCDET-Eingabedatei ist in Abb. 3.3 zu sehen. Die aufgelisteten Größen sind die Ausfallraten der Komponenten aus dem Tankbeispiel im Anhang. Ausfallraten sind (epistemische Unsicherheit) wegen des geringen Datenmaterials, das für ihre Schätzung zur Verfügung steht, oft nur ungenau bekannt.

```
@*****************************************************************************
@ Epistemic variables sampled externally (e.g. by SUSA) (optional)
@*****************************************************************************
@
@ Number of epistemic variables
    3
@
@ Number of values (sample size) to be considered for epistemic variables
    100
@
@ Name of epistemic variables
    FailRateValve
    FailRatePump1
    FailRate Pump1
@
@ Name of file with epistemic variables
    susa.dsn
```

Abb. 3.3 MCDET-Eingabedatei: Eingabe von epistemischen Variablen

3.1.4 Aleatorische Variable für die Berücksichtigung mittels Monte-Carlo-Simulation

Aleatorische Variable sind Parameter, deren Werte aufgrund stochastischer Einflussfaktoren variieren und die deshalb mit einer aleatorischen (d.h. stochastischen) Unsicherheit verbunden sind. Diskrete aleatorische Variable (z. B. Systeme und Komponenten verfügbar/nicht verfügbar) werden in MCDET im Allgemeinen durch die Verzweigungen in einem dynamischen Ereignisbaum berücksichtigt. Informationen zu diesen Verzweigungen werden in den Verzweigungsfenstern spezifiziert (siehe Abschnitt 3.2.1). Diskrete aleatorische Variable brauchen nicht gesondert spezifiziert zu werden. Dafür kommen sowohl Simulator-Variablen als auch Zeit-Zustandsvariablen in Frage.

Stetige aleatorische Variable (z. B. Ausfallzeitpunkte von Systemen und Komponenten, Zeitpunkt der Spannungswiederkehr in einem Station Black-out Szenario) werden durch Monte-Carlo-Simulation berücksichtigt. Sie werden im Gegensatz zu den diskreten aleatorischen Variablen, die in den Verzweigungsfenstern mit ausgewählten Simulator- oder Zeit-Zustandsvariablen gleichgesetzt werden, immer separat spezifiziert. Stetige aleatorische Variable sind nicht immer gleichzusetzen mit Simulator- oder Zeit-Zustandsvariablen. Mit ihrer Hilfe werden häufig Bedingungen in Bezug auf Simulator- oder Zeit-Zustandsvariablen definiert. Diese Bedingungen sind wichtiger Bestandteil von Zeit-Zustandsfenstern. Ein Beispiel für eine stetige aleatorische Variable, die zur Definition einer Bedingung herangezogen werden kann, ist die Ausfallzeit einer Komponente. Wenn in einem Verzweigungsfenster die Bedingung definiert ist, dass die Prozesszeit gleich der Ausfallzeit ist, so wird MCDET eine Verzweigung generieren, sobald diese Bedingung erfüllt ist, d. h. sobald die Prozesszeit dem ausgespielten Wert für die Ausfallzeit entspricht. Die Verzweigung, die erzeugt wird, kann z. B. aus zwei Simulationspfaden bestehen. Im ersten Simulationspfad ist die Komponente weiterhin verfügbar wie (deterministisch) vorgesehen, in der Variante zu diesem Simulationspfad ist die Komponente ab dem aktuellen Zeitpunkt (Prozesszeit = Ausfallzeit) nicht mehr verfügbar.

Sind stetige aleatorische Variable gleichzusetzen mit Simulator- oder Zeit-Zustandsvariablen, dann kann die Zuordnung entweder bei der Initialisierung (vgl. Abb. 3.9 und Abb. 3.10) oder in einem Verzweigungsfenster erfolgen.

Die ausgespielten Werte für die stetigen aleatorischen Variablen sind im Allgemeinen prozessspezifisch. Deshalb werden sie in prozessspezifischen Hashtabellen abgelegt.

Statt die Werte von aleatorischen Variablen durch MCDET-eigene Routinen ausspielen zu lassen, können sie auch über eine externe Datei eingelesen werden. Dafür eignen sich solche aleatorischen Variablen, die unabhängig vom Prozess sind (z. B. Zeitpunkt der Spannungswiederkehr in einem Szenario mit Totalausfall der Spannungsversorgung).

```
@*****************************************************************************
MC simulation for aleatory variables
@*****************************************************************************
@
@ Number of MC simulation runs (DET generations) to be performed for aleatory variables
    50
@ Number of runs / DETs to be printed in 'sequence.prn' (Maximum number!)
    1
@ Aleatory MC variables sampled outside of MCDET (e.g. by SUSA)
@=============================================================================
@
@ Number of aleatory variables
    0
@ File of aleatory variables
@ (with different samples for each set of epistemic variables)
@
@ Name of aleatory variables
@
@ Aleatory MC variables sampled by MCDET
@=============================================================================
@
@ Number of aleatory variables
    3
@ List of aleatory variables
@ Name
    tClose_Valve
    tStop_Pump1
    tRun_Pump2
@ SEED value of random number generator
    123457
```

Abb. 3.4 MCDET-Eingabedatei: Eingabe von aleatorischen Variablen für die Berücksichtigung mittels Monte-Carlo-Simulation (Beispiel 1)

Wie die aleatorischen Variablen für die Berücksichtigung mittels Monte-Carlo-Simulation spezifiziert werden, ist beispielhaft in Abb. 3.4 zu sehen. Die aufgelisteten aleatorischen Variablen beziehen sich auf das Tankbeispiel und haben folgende Bedeutung: ‘tClose_Valve‘ bezeichnet den zufälligen Zeitpunkt, wann das Ventil fälschlicherweise schließt, ‘tStop_Pump1‘ ist der zufällige Zeitpunkt, wann Pumpe 1 fälschlicherweise aufhört zu laufen und ‘tRun_Pump2‘ ist der zufällige Zeitpunkt, wann Pumpe 2 fälschlicherweise beginnt zu laufen.

Für aleatorische Variable, deren Werte innerhalb von MCDET ausgespielt werden, müssen zusätzlich sogenannte ‘Sampling Windows‘ definiert werden, die angeben, unter welchen Bedingungen welche Verteilung zugrunde zu legen ist, um für eine aleatorische Variable einen Wert auszuspielen (siehe Abschnitt 3.2.1).

3.2 Neue und abgeänderte Funktionalitäten

Die erforderlichen Spezifikationen zu den Zeit-Zustandsfenstern einer MCDET-Anwendung mussten aufgrund von Anwendererfahrung überarbeitet werden. Entsprechende Anweisungen im Probabilistik-Modul wurden angepasst. Außerdem wurde das Spektrum der Analysemöglichkeiten mit dem Probabilistik-Modul um zusätzliche Funktionalitäten zur Berücksichtigung probabilistischer Fragestellungen erweitert. Diese neuen Funktionalitäten ermöglichen die Herleitung zusätzlicher Zeit-Zustandsvariablen als Funktion von Prozessgrößen, die exklusive Durchführung einer Monte-Carlo-Simulation ohne die Simulation eines dynamischen Ereignisbaums sowie die Berücksichtigung einer Verzweigung (d. h. alternativer Systemzustände inklusive probabilistischer Bewertung) bereits zu Beginn einer MCDET-Simulation.

Die Arbeiten zur Entwicklung neuer und Überarbeitung vorhandener Funktionalitäten waren zu Beginn des Vorhabens nicht explizit vorgesehen, wurden aber im Hinblick auf die Einsatzmöglichkeiten des MCDET-Analysewerkzeugs sowie aufgrund der Auswertung von Anwendererfahrung als wichtig erachtet und deshalb durchgeführt.

3.2.1 Angaben zu Zeit-Zustandsfenstern

Das neue Probabilistik-Modul kann vier verschiedene Typen von Zeit-Zustandsfenstern berücksichtigen und zwar das ‘Absorbierende Fenster‘ (Absorbing Window), das ‘Funktionsfenster‘ (Function Window), das ‘Verzweigungsfenster‘ (Transition Window) und das ‘Auswahlfenster‘ (Sampling Window). Diese Fenster werden definiert durch die Bedingungen für ihre Aktivierung sowie durch Aktionen, die bei Zutreffen der Bedingungen für die Aktivierung durchzuführen sind. Beim ‘Absorbierenden Fenster‘ entspricht die Aktion dem Beenden eines Simulationslaufs (siehe Abschnitt 4.2). Beim ‘Funktionsfenster‘ werden die Werte von Variablen als Funktion von anderen Variablen ermittelt. Beim ‘Verzweigungsfenster‘ werden neue Verzweigungen eines dynamischen Ereignisbaums generiert und beim ‘Auswahlfenster‘ werden neue Werte für aleatorische Variable ausgespielt.

Die erforderlichen Angaben für die Fenster wurden im Vergleich zu den ursprünglichen Versionen mehr oder weniger stark abgeändert. Für alle Fenster bis auf das ‘Absorbierende Fenster‘ wurden die Angaben zur Definition einer Bedingung erweitert. Jede Bedingung in einem Zeit-Zustandsfenster wird zunächst definiert durch Angabe einer Simulator-Variablen bzw. Zeit-Zustandsvariablen in Verbindung mit dem Zustand, in dem sich diese Variable befinden muss. Zusätzlich zu diesen Informationen erfordert die Definition einer Bedingung eine Angabe darüber, ob der Zustand der Variablen einen Stimulus (Stimulus = 1) darstellt oder nicht (Stimulus = 0).

Stellt der Zustand einer Variablen einen Stimulus dar (Stimulus = 1), so bedeutet dies, dass sich die Variable vom letzten gerechneten Zeitpunkt in den aktuellen Zeitpunkt des Simulationslaufs geändert haben muss und dass der geänderte Zustand erfüllt sein muss, damit die Aktionen im Fenster durchgeführt werden. Stellt der Zustand keinen Stimulus dar (Stimulus = 0), so bedeutet dies, dass sich die Variable vom letzten in den aktuellen Zeitpunkt des Simulationslaufs nicht unbedingt geändert haben muss, dass aber der angegebene Zustand erfüllt sein muss, damit die Aktionen im Fenster durchgeführt werden.

‘Funktionsfenster‘, ‘Verzweigungsfenster‘ und ‘Auswahlfenster‘ benötigen mindestens einen Zustand, der einen Stimulus darstellt. Durch die Angabe von Stimuli ist festgelegt, welche Veränderungen während eines Simulationslaufs zur Aktivierung eines Zeit-Zustandsfensters und seiner Aktionen führen.

Als Beispiel soll ein Ventil betrachtet werden, das bei Anforderung erfolgreich öffnet, das aber aufgrund von fehlerhafter Steuerung nach einer zufälligen Zeit fälschlicherweise wieder schließen kann. In der Analyse muss also für das Ventil eine Ausfallzeit (fehlerhaftes Schließen) ausgespielt werden. Die Ausfallzeit soll dabei abhängig von der Temperatur sein, die bei Anforderung des Ventils herrscht. Befindet sich die Temperatur auf einem hohen Niveau, soll ein Ausfall mit einer höheren Ausfallrate eintreten als wenn Normalbedingungen für die Temperatur herrschen. Stimulus für das Ausspielen eines Wertes für die Ausfallzeit ist in diesem Fall die Anforderung des Ventils. D. h. das Ventil wird während des Prozessverlaufs angefordert zu öffnen. Die zusätzliche Angabe für den Temperaturbereich legt fest, unter welcher Bedingung die Ausfallzeit des Ventils mit der jeweiligen Ausfallrate auszuspielen ist. Es gibt in diesem Fall also zwei Auswahlfenster. In dem einen Fenster (vgl. Abb. 5.8) ist die Bedingung durch das geöffnete Ventil und die Temperatur auf normalem Level (z. B. Temperatur ≤ 35 °C) gegeben. In dem anderen Fenster (vgl. Abb. 5.9) ist neben dem geöffneten

Ventil die Temperatur auf hohem Level (z. B. Temperatur > 35 °C) als Bedingung definiert. Hätten die beiden Fenster keinen Stimulus, so wären ihre Bedingungen z. B. auch dann erfüllt, wenn das Ventil bereits erfolgreich geöffnet wurde und die Temperatur im Prozessverlauf den einen oder anderen angegebenen Level erreicht.

Die folgenden Abbildungen in diesem Abschnitt zeigen je ein Beispiel für ein ‘Auswahlfenster‘ und ein ‘Verzweigungsfenster‘. Ein Beispiel für ein Funktionsfenster ist in Abb. 3.8 in Abschnitt 3.2.2 zu sehen. Beispiele für ‘Absorbierende Fenster‘ sind in Abb. 4.1 aufgeführt. Alle Beispiele beziehen sich auf das Tankbeispiel.

```
@*******************************************************************************
Sampling Windows
@*******************************************************************************
@.....
Window 1
@.....
@
--Condition
@ Variable  Stimulus  Condition
     Init
--Distribution
@                          Add              Distribution
@ Variable                 Variable         type      npar      parameter
  tClose_Valve             Time             11        4         0.0001 1.  0  1000
  tStop_Pump1              Time             11        4         0.0005 1. 0 1000
  tRun_Pump2               Time             11        4         0.0005 1. 0 1000
```

Abb. 3.5 MCDET-Eingabedatei: Spezifikation eines ‘Auswahlfensters‘ (Sampling Window), Beispiel 1 (vgl. Abb. 5.6)

Im Auswahlfenster in Abb. 3.5 stehen die erforderlichen Angaben für das Ausspielen von Werten für die aleatorischen Variablen ‘tClose_Valve‘, ‘tStop_Pump1‘ und ‘tRun_Pump2‘ des Tankbeispiels (siehe Abschnitt 3.1.4). Bedingung für das Ausspielen der Werte ist der Anfangszeitpunkt und -zustand der Simulationsläufe (Angabe: ‘Init‘). Anfangszeitpunkt und -zustand werden in der MCDET-Eingabedatei durch die Spezifikation der Anfangswerte (Initial Values) für die Simulator- und Zeit-Zustandsvariablen definiert. Die Verteilung, aus der ausgespielt werden soll, ist für alle drei Variablen vom Typ = 11, was einer Gamma-Verteilung entspricht. Insgesamt werden vier Parameter für die Verteilung angegeben, nämlich 0.0001 (bzw. 0.0005) 1, 0 und 1000. Die letzten beiden Parameter (0 und 1000) sind Minimum und Maximum der Gamma-Verteilung.

Erfolgt im Auswahlfenster eine Angabe zu ‘Add Variable‘, so hat das zur Folge, dass auf die ausgespielten Wert der betreffenden Variablen (z. B. Ausfallzeitpunkt ‘tClose_Valve‘ des Ventils) der Wert der unter ‘Add Variable‘ angegebenen Variablen ad-

diert wird. Im vorliegenden Fall wird z. B. auf den ausgespielten Wert für den Versagenszeitpunkt des Ventils (‘tClose_Valve‘) der Anfangswert der Prozesszeit (Time) addiert, was in diesem Fall keine Auswirkungen hat, da der Anfangswert gleich 0 ist.

Im Verzweigungsfenster in Abb. 3.6 stehen die erforderlichen Angaben für das Erzeugen einer Verzweigung im Tankbeispiel. Bedingung für diese Verzweigung ist, dass die Prozesszeit Time zum ersten Mal größer oder gleich der aleatorischen Variablen ‘tClose_Valve‘ ist (Stimulus = 1) und das Ventil offen ist. Die Verzweigung, die erzeugt wird, besteht aus zwei Simulationspfaden. Im ersten Simulationspfad bleibt das Ventil offen (Valve = 0) wie (deterministisch) vorgesehen, in der Varianten zu diesem Simulationspfad ist das Ventil ab dem aktuellen Zeitpunkt geschlossen (Valve = 1). Die Wahrscheinlichkeit dafür wird durch die Größe ‘pClose_Valve‘ angegeben. ‘pClose_Valve‘ wird aufgrund der Anweisungen unter dem Schlüsselwort ‚‘Branching probabilities* (vgl. Abb. 3.7) von MCDET ermittelt. Statt 'pClose_Valve‘ kann auch der Wert selbst angegeben werden.

```
@*******************************************************************************
Transition Windows
@*******************************************************************************
@.....................................................................
Window 1
@.....................................................................
--Condition
@ Variable  Stimulus  Condition
    Time        1     >= tClose_Valve
    Valve       0      = 0
--Transition
@                Number of        State of
@ Variable       Variants         Variant  Probability
    Valve        1                1        pClose_Valve
```

Abb. 3.6 MCDET-Eingabedatei: Spezifikation eines ‘Verzweigungsfensters‘ (Transition Window)

Die Angaben in Abb. 3.7 führen dazu, dass MCDET für das angegebene Argument (z. B. 1000) den Wert der Verteilung (z. B. Gamma Verteilung mit den Parametern 0.0001 und 1), d. h. die kumulierte Wahrscheinlichkeit, berechnet. Diese Wahrscheinlichkeit kann dann als Verzweigungswahrscheinlichkeit in einem Verzweigungsfenster verwendet werden (siehe oben).

```
@*****************************************************************************
Branching probabilities to be calculated (optional)
@*****************************************************************************
@
@ Number of branching probabilities
    1
@ Branching probabilities to be calculated for argument x
@
@                                        Distribution
@ name                  Argument         type    npar    parameter
    pClose_Valve        1000             11      2       0.0001 1
```

Abb. 3.7 MCDET-Eingabedatei: Angaben zur Berechnung einer Verzweigungswahrscheinlichkeit

3.2.2 Neues Zeit-Zustandsfenster zur Definition von Funktionen

Zeit-Zustandsvariable können als Funktionen von Größen des Rechenprogramms definiert werden (siehe Abschnitt 3.1.2). Die entsprechende Spezifikation dafür erfolgt in sogenannten Funktionsfenstern (Function Windows). Diese Fenster enthalten die Bedingungen für ihre Aktivierung, die Namen der Zeit-Zustandsvariablen, die Funktionen von Prozessgrößen sind, sowie die Funktionen selbst. Die Funktionen werden dabei als FORTRAN-Formel ausgedrückt. Jedes Mal, wenn die Bedingungen eines Funktionsfensters erfüllt sind, werden die spezifizierten Funktionen MCDET intern kompiliert und ausgeführt.

```
@*****************************************************************************
Function Windows
@*****************************************************************************
@
@.....................................................................................
Window 1
@.....................................................................................
@
@--Condition
@ Variable  Stimulus  Condition
    Valve           1         =0
@
@--Function
@
@ Variable                  Function
    nValveOpen              nValveOpen+1
```

Abb. 3.8 MCDET-Eingabedatei: Spezifikation eines Funktionsfensters (Function Window)

Abb. 3.8 gibt ein Beispiel für ein Funktionsfenster. Im Beispiel ist ‘nValveOpen‘ die Anzahl der Anforderungszyklen (Öffnen) des Ventils aus dem Tankbeispiel. Zu Beginn der Simulation ist die Anzahl gleich 0. Jedes Mal, wenn das Ventil im Verlauf eines Simulationslaufs geöffnet wird (Valve = 0), wird ‘nValveOpen‘ um 1 hochgezählt.

3.2.3 Neue Option für die ausschließliche Durchführung von Monte-Carlo-Simulationen

Mit MCDET können auch ausschließlich Monte-Carlo-Simulationen zur Berücksichtigung epistemischer und aleatorischer Unsicherheiten durchgeführt werden. Wenn es keine Abhängigkeiten der Unsicherheiten von der Prozessdynamik gibt, dann können Simulator-Variablen gleich zu Beginn mit den ausgespielten Werten der epistemischen und aleatorischen Variablen initialisiert werden. Dazu werden die epistemischen und aleatorischen Variablen sowie die Simulator-Variablen wie vorher beschrieben in der MCDET-Eingabedatei spezifiziert. Bei der Initialisierung der Simulator-Variablen wird statt eines festen Wertes der Name der epistemischen bzw. aleatorischen Variablen angegeben, deren Werte für die Initialisierung herangezogen werden sollen.

In entsprechenden Auswahlfenstern (Sampling Windows) wird anschließend als Bedingung das Schlüsselwort ‘Init‘ spezifiziert. MCDET interpretiert dieses Schlüsselwort als Anfangszustand. Fenster mit der Bedingung ‘Init‘ werden noch vor den Simulationen aktiviert, um entsprechende Aktionen durchzuführen. Auswahlfenster mit dieser Bedingung führen also dazu, dass gleich zu Beginn Werte für die aleatorischen Variablen ausgespielt werden. Die Anzahl der Werte, die für jede Variable ausgespielt werden, hängt von der entsprechenden Angabe in der MCDET-Eingabedatei ab (vgl. Abb. 3.4: ‘Number of Monte Carlo simulation runs (DET generations) to be performed‘).

Sind sowohl epistemische als auch aleatorische Variable zu berücksichtigen, so werden die Werte für die Monte-Carlo-Simulationen in zwei Schleifen – einer äußeren und einer inneren Schleife – ausgespielt. In der äußeren Schleife werden die Werte für die epistemischen Variablen ausgespielt (bzw. eingelesen). Für jeden Wertesatz aus der äußeren Schleife werden anschließend in einer inneren Schleife die Werte für die aleatorischen Variablen ausgespielt.

Für jeden Simulationslauf, der durchgeführt werden muss, führt MCDET die Initialisierung der Simulator-Variablen mit den zugehörigen Werten durch. Das Steuerungspro-

gramm von MCDET verteilt anschließend die Simulationsläufe mit den unterschiedlichen Anfangszuständen auf einem Rechencluster zur parallelen Berechnung.

Wenn es Abhängigkeiten der Unsicherheiten von der Prozessdynamik gibt, dann wird einer Simulator-Variablen erst im Verlauf eines Simulationslaufes ein ausgespielter Wert einer epistemischen oder aleatorischen Variablen zugeordnet. Die Bedingung, zu der die Zuordnung erfolgen soll, muss in einem Verzweigungsfenster festgelegt werden. Wenn ausschließlich eine Monte-Carlo-Simulation durchgeführt werden soll, dann ist in dem Verzweigungsfenster für die Simulator-Variable genau eine Variante zu spezifizieren, die mit der Wahrscheinlichkeit 1 eintritt. Die Variante ist mit dem ausgespielten Wert der aleatorischen Variablen gleichzusetzten. Dadurch dass diese Variante mit Wahrscheinlichkeit 1 eintreten soll, wird die Sequenz mit dem Ausgangswert der Simulator-Variablen abgebrochen.

Die folgenden vier Abbildungen zeigen beispielhaft, welche Eingaben in der MCDET-Eingabedatei erforderlich sind, um ausschließlich Monte-Carlo-Simulation durchzuführen. Es gibt insgesamt zwei aleatorische Variable, die mittels Monte-Carlo-Simulation berücksichtigt werden: ‘LevelInit‘ und ‘ValveValue‘. Die Simulator-Variable ‘FillLevel‘ (Füllstand im Tank) wird gleich zu Beginn mit den ausgespielten Werten der aleatorischen Variablen ‘LevelInit‘ initialisiert. Das führt zu insgesamt zehn Simulationsläufen (Number of MC simulation runs = 10) mit unterschiedlichen Anfangswerten für ‘FillLevel‘. Die Simulator-Variable ‘Valve‘ wird am Anfang auf 0 gesetzt. Erst wenn ‘Valve‘ während einer Simulation auf 1 (Ventil geschlossen) gesetzt wird, wird der ausgespielte Wert von ‘ValveValue‘ berücksichtigt. Der Wert ist entweder 1 (Ventil geschlossen) oder 2 (Ventil in Offenstellung ausgefallen) mit einer Wahrscheinlichkeit von 0,95 bzw. 0,05.

```
@*****************************************************************************
MC simulation for aleatory variables (optional)
@*****************************************************************************
@
@ Number of MC simulation runs (DET generations) to be performed for aleatory variables
@
    10
@
@ Number of runs / DETs to be printed in 'sequence.prn' (Maximum number!)
    1
@
@ Aleatory MC variables sampled outside of MCDET (e.g. by SUSA)
@============================================================
@
@ Number of aleatory variables
    0
@
@ Aleatory MC variables sampled by MCDET
@ ============================================================
@
@ Number of aleatory variables
    2
@ List of aleatory variables
@
@ Name
    LevelInit
    ValveValue
@
@ SEED value of random number generator
@
    123457
```

Abb. 3.9 MCDET-Eingabedatei: Eingabe von aleatorischen Variablen für die Berücksichtigung mittels Monte-Carlo-Simulation, Beispiel 2

```
@*****************************************************************************
Simulator variables
@*****************************************************************************
@
@ Number of simulator variables
    5
@ List of simulator variables
@
@ Name          Initial Value
@
    Time                0
    FillLevel           LevelInit
    Valve               0
    Pump1               1
    Pump2               0
```

Abb. 3.10 MCDET-Eingabedatei: Initialisierung einer Simulator-Variablen mit einer aleatorischen Variablen

```
@*****************************************************************************
Sampling Windows
@*****************************************************************************
@
@.....................................................................
Window 1
@.....................................................................
@
--Condition
@ Variable  Stimulus  Condition
    Init
--Distribution
@                Add              Distribution
@ Variable       Variable         type      npar      parameter
    LevelInit    -                3          4        5   1   0  10
    ValveValue   -                1          4        1   2   0.95 0.05
```

Abb. 3.11 MCDET-Eingabedatei: Spezifikation eines ‘Auswahlfensters‘ (Sampling Window), Beispiel 2

```
@*****************************************************************************
Transition Windows
@*****************************************************************************
@.....................................................................
Window 1
@.
--Condition
@ Variable  Stimulus  Condition
    Valve          1       = 1
--Transition
@                Number of        State of
@ Variable       Variants         Variant   Probability
    Valve        1                ValveValue    1
```

Abb. 3.12 MCDET-Eingabedatei: Spezifikation eines Verzweigungsfensters für die Zuordnung einer aleatorischen Variablen zu einer Simulator-Variablen

3.2.4 Neue Option für Verzweigungen am Anfang eines dynamischen Ereignisbaums (DET)

MCDET kann jetzt auch Verzweigungen gleich am Anfang der Simulationen generieren. D. h. gleich von Beginn an können alternative Werte von entsprechenden Simulator-Variablen zusammen mit ihren Eintrittswahrscheinlichkeiten berücksichtigt werden. Dazu muss lediglich die Bedingung ‘Init‘ für ein Verzweigungsfenster spezifiziert werden.

Im Beispiel in Abb. 3.13 werden zwei Anfangswerte für die Pumpe 1 aus dem Tankbeispiel berücksichtigt. Dies ist zum einen der Wert 1 (Pumpe läuft), mit dem Pumpe 1 initialisiert wurde (vgl. Abb. 3.1), und zum anderen der Wert 0 (Pumpe steht), der durch die Verzweigung gleich zu Beginn noch zusätzlich betrachtet wird (State of Variant = 0). Beide Anfangszustände der Pumpe 1 kommen mit Wahrscheinlichkeit 0,5 vor.

```
@*******************************************************************************
Transition Windows
@*******************************************************************************
@
Window 0
@
@
--Condition
@ Variable  Stimulus  Condition
     Init
@
@
--Transition
@                  Number of        State of
@ Variable         Variants         Variant  Probability
     Pump1         1                0        0.5
```

Abb. 3.13 MCDET-Eingabedatei: Spezifikation eines Verzweigungsfensters für eine Verzweigung am Anfang eines dynamischen Ereignisbaums

3.3 Neue Programmroutinen MCDETinit und MCDETexe

Durch Zusammenfassung entsprechender Routinen der ersten Version des Probabilistik-Moduls wurde die Zahl der Programmroutinen, die das neue Probabilistik-Modul dem Steuerungsmodul bereitstellt, um MCDET relevante Aktionen aufzurufen, erheblich reduziert. Dadurch erhält das Steuerungsmodul eine übersichtlichere und nachvollziehbarere Struktur, die eventuell erforderliche Anpassungen in der Zukunft wesentlich erleichtern. Das Probabilistik-Modul stellt nur noch die beiden Routinen MCDETinit und MCDETexe bereit.

Die Routine MCDETinit war bereits Bestandteil der ersten Version des Probabilistik-Moduls. Der Aufruf dieser Routine muss im Steuerungsmodul vor dem eigentlichen Start der Simulationsläufe erfolgen. Er führt zur Auswertung der Benutzerangaben in der MCDET-Eingabedatei, der Definition und Ablage von konstanten Datenelementen und -feldern, auf die das Probabilistik-Modul bei seinen Berechnungen zugreifen muss, sowie zur Definition, Initialisierung und Ablage von variablen Datenstrukturen, die für

den Datenaustauch zwischen Probabilistik-Modul, Steuerungsprogramm und dem jeweiligen Simulationsprozess des eingesetzten Rechenprogramms erforderlich sind.

Die Routine MCDETexe überprüft den Zustand des vom Steuerungsmodul ausgewählten Simulationsprozesses daraufhin, ob eine MCDET-Aktion durchgeführt werden muss oder nicht. Wenn ja, werden die entsprechenden Programmanweisungen für die Monte-Carlo-Simulation oder die Erzeugung eines dynamischen Ereignisbaums (DET) ausgeführt. Die Routine MCDETexe ist nicht nur auf die Behandlung aleatorischer Variablen beschränkt. Sie beinhaltet auch Anweisungen für die Durchführung von Monte-Carlo-Simulation zur Berücksichtigung epistemischer Variablen. In der ersten Version des Probabilistik-Moduls gab es insgesamt sechs Routinen, die die Aufgaben von MCDETexe ausgeführt haben (SimulationInit, TSWindow, SimWindow, TransWindow, BaseProbability, GetEpistemic).

4 Methoden zur Reduktion des Rechenaufwands bei Analysen mit MCDET

Der Rechenaufwand von MCDET-Analysen kann erheblich sein, gerade dann, wenn die Anwendung des eingesetzten Rechenprogramms sehr zeitintensiv ist. Um die Effizienz von Analysen mit MCDET weiter zu verbessern, ist zusätzlich zum neuen Ansatz für die Steuerung von parallelen Simulationsprozessen auf Rechenclustern (siehe 2) der Einsatz von Methoden erforderlich, die zu einer weiteren Reduktion des Rechenaufwands führen.

4.1 Simulation eines dynamischen Mega-Ereignisbaums

Mit der MCDET-Methode wird eine Kombination aus Monte-Carlo-Simulation und dynamischer Ereignisbaum-Methode angewendet, um die Unsicherheiten einer Anwendung zu berücksichtigen. Diese Unsicherheiten beziehen sich auf aleatorische Variable, die aufgrund zufälliger Ereignisse variieren, und auf epistemische Variable, die eigentlich feste Werte haben, die aber aufgrund mangelnden Kenntnisstands nur ungenau bekannt sind und deshalb variiert werden müssen. Diskrete aleatorische Variable, die sich auf Systemzustände (z. B. an/aus, offen/geschlossen) und Zeitpunkte (Ausfallzeitpunkt bei mehreren Anforderungszyklen) beziehen, werden durch die dynamische Ereignisbaum Methode berücksichtigt. Stetige aleatorische Variable und epistemische Variable werden durch Monte-Carlo-Simulation berücksichtigt.

Das Ergebnis von MCDET-Simulationen ist eine Stichprobe von mehreren dynamischen Ereignisbäumen. Jeder Ereignisbaum aus dieser Stichprobe wird mit einem unterschiedlichen Satz von Werten für die stetigen aleatorischen Größen konstruiert. Mit der alten, im Rahmen des Vorhabens RS1111 entwickelten MCDET-Version erfolgt die Konstruktion dynamischer Ereignisbäume separat einer nach dem anderen oder separat und parallel auf verschiedenen Rechenknoten.

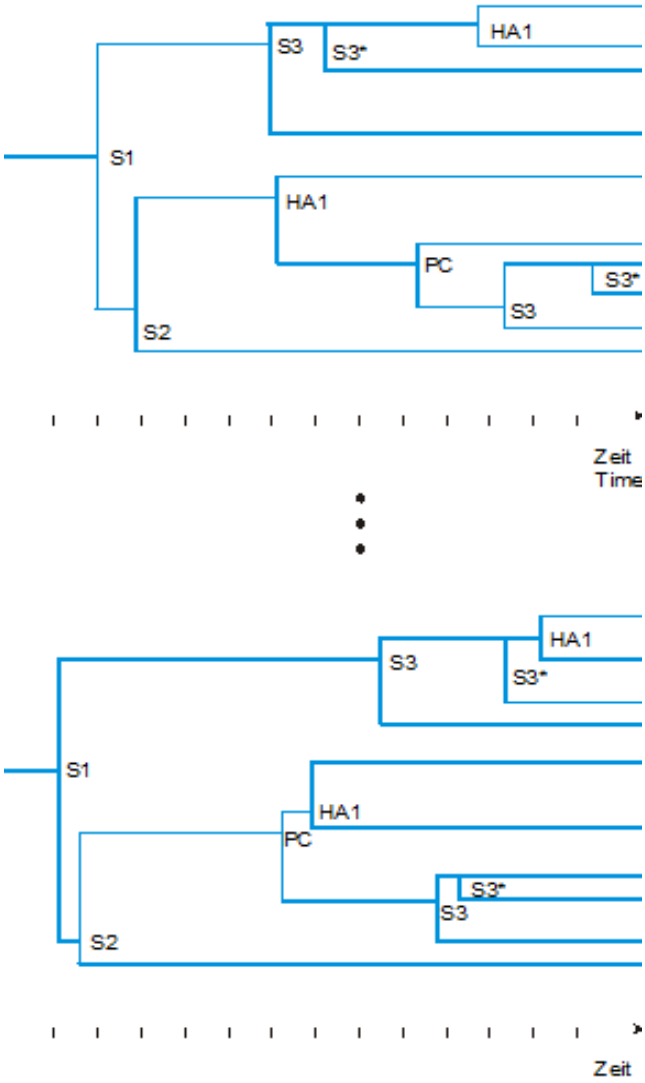


Abb. 4.1 Stichprobe dynamischer Ereignisbäume

Der Konstruktionsalgorithmus für eine Stichprobe von dynamischen Ereignisbäumen wurde im Rahmen des Vorhabens so optimiert, dass nun die Sequenzen aller Ereignisbäume als Bestandteil eines einzigen dynamischen Mega-Ereignisbaums generiert werden. Dadurch wird ein effizienter Ablauf der Simulationsläufe erreicht und erhebliche Rechenzeit eingespart, indem identische Sequenzen in den einzelnen Ereignisbäumen nur einmal gerechnet werden. Die Zahl identischer Sequenzen in den Ereignisbäumen variiert von Anwendungsfall zu Anwendungsfall und kann zum Teil sehr groß sein. In beispielhaften Anwendungen bewegte sich diese Zahl zwischen 0 und 240.

Um bei der Auswertung die Simulationsergebnisse den einzelnen Ereignisbäumen zuordnen zu können, wurde die Indikatorvariable DETIndex eingeführt. DETIndex = 0

bedeutet, dass die berechneten Sequenzen jedem der Bäume zuzuordnen sind. Wenn DETIndex = i, gehören die Sequenzen zum Baum i, i = 1,…, nDET, wobei nDET die Gesamtzahl aller Ereignisbäume ist. Der Wert von nDET wird in der Eingabedatei von MCDET angegeben (vgl. Abb. 3.4).

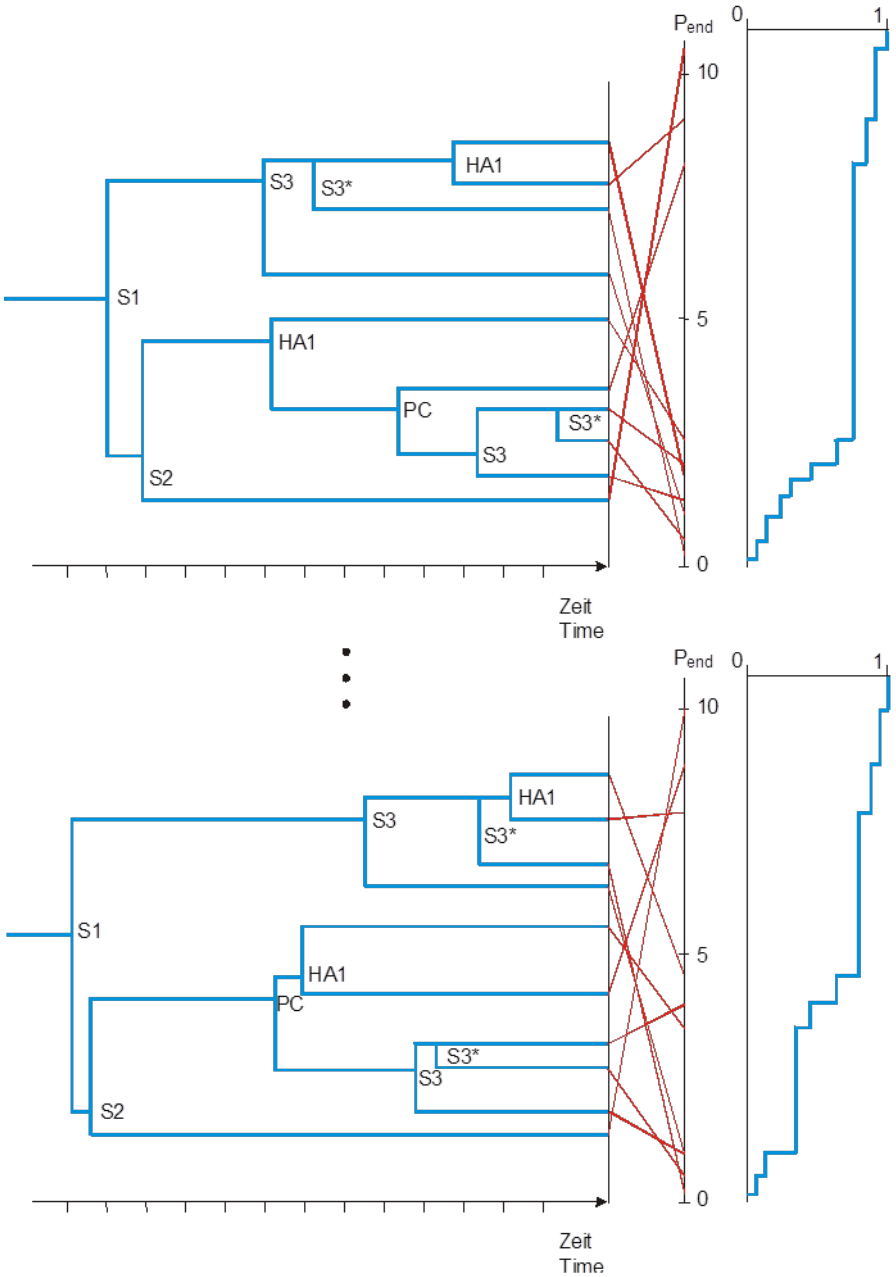


Abb. 4.2 Stichprobe dynamischer Ereignisbäume mit baumspezifischen Verteilungen für eine Prozessgröße

Beim neuen Konstruktionsalgorithmus wird berücksichtigt, dass Sequenzen, die jedem der Bäume zuzuordnen sind, unterschiedliche Eintrittswahrscheinlichkeiten in den einzelnen Bäumen haben können. Deshalb wird jeder Sequenz, die jedem der Bäume

zuzuordnen und deshalb durch DETIndex = 0 gekennzeichnet ist, ein Vektor von Eintrittswahrscheinlichkeiten zugeordnet. Die Länge des Vektors entspricht der Gesamtzahl der Ereignisbäume. Das i-te Element des Vektors ist die Eintrittswahrscheinlichkeit der jeweiligen Sequenz in Baum i, i = 1,…, nDET. Wenn eine Sequenz mit DETIndex = i, i = 1,…, nDET gekennzeichnet ist, wird dieser Sequenz nur ein einziger Wert für die Eintrittswahrscheinlichkeit zugeordnet.

Durch die Zuordnung von Eintrittswahrscheinlichkeiten zu jeder Sequenz eines dynamischen Ereignisbaums kann für jede berechnete Prozessgröße eine baumspezifische Verteilung ermittelt werden (siehe Abb. 4.2). Aus den baumspezifischen Verteilungen wird durch Anwendung statistischer Methoden schließlich die anwendungsspezifische Verteilung ermittelt.

4.2 Definition von absorbierenden Zuständen als zusätzliche Option in der MCDET-Eingabedatei

Absorbierende Zustände sind sichere, Gefährdungs- oder Schadenszustände, in denen das System, das Gegenstand der Simulation ist, verbleiben wird, wenn es diese erreicht hat, weil es keine Einflussfaktoren mehr gibt, die eine relevante Zustandsänderung bewirken können. In der MCDET-Eingabedatei können absorbierende Zustände als zusätzliche Option für die Durchführung der Simulationsläufe vom Benutzer festgelegt werden. Wenn dann der Systemzustand während eines Simulationslaufs mit einem absorbierenden Zustand übereinstimmt, wird der Simulationslauf vorzeitig abgebrochen. Wenn z. B. bei einem Unfallablauf in einem Kernkraftwerk die Kühlung des Kerns durch die erfolgreiche Einspeisung des Notkühlsystems über zwei Stränge sichergestellt ist, so kann der Benutzer die Verfügbarkeit von zwei Notkühlsträngen als absorbierenden Zustand definieren. In der Folge werden dann alle Simulationsläufe vorzeitig beendet, und zwar zu dem Zeitpunkt, wenn über mindestens zwei Notkühlstränge die Einspeisung beginnt.

```
@*****************************************************************************
Absorbing Windows (Conditions for stopping sequence calculation)
@*****************************************************************************
@.....
Window 1
@.....
--Condition
@ Variable  Condition
   Time      >= 1000
--End
@.....
Window 2
@.....
@
--Condition
@ Variable  Condition
   FillLevel     <= 0
--End
@.....
Window 3
@.....
--Condition
@ Variable  Condition
   FillLevel     >= 10
--End
@
```

Abb. 4.3 MCDET-Eingabedatei: Spezifikation von absorbierenden Fenstern (Absorbing Windows)

Wie absorbierende Zustände in der Eingabedatei definiert werden, ist in Abb. 4.3 festgehalten. Bei dem in der Abbildung behandelten Beispiel geht es um das Tanksystem, dessen Zulauf über zwei Pumpen und dessen Ablauf über ein Ventil sichergestellt wird. Wenn diese Komponenten nicht mehr funktionieren, kommt es entweder zum Austrocknen (FillLevel ≤ 0 m) oder zum Überlaufen (FillLevel ≥ 10 m) des Tanks. Beide Zustände werden als absorbierend definiert. Ein weiterer absorbierender Zustand bzw. Zeitpunkt ist das Ende der Problemzeit (Time ≥ 1000 s).

4.3 Überwachung und Klassifizierung von Simulationsläufen während der Laufzeit

Die hier beschriebenen Methoden zielen darauf ab, jeden Simulationslauf von Beginn an zu überwachen und während der Laufzeit daraufhin zu klassifizieren, ob er weitergerechnet werden soll, weil er höchstwahrscheinlich in einen unerwünschten Zustand verläuft, oder nicht. Simulationsläufe, die nicht weitergerechnet werden brauchen, sind solche, in denen sich das System bzw. die Anlage in einem kontrollierbaren Zustand befindet und kein Risiko für weitere Schäden besteht. Die Methoden basieren auf Prozessgrößen, die dazu geeignet sind, die Entwicklung eines Stör- bzw. Unfallablaufs vorherzusagen (z. B. Hüllrohrtemperatur, H_2-Masse, UO_2-Schmelzmasse).

Bevor ein Simulationslauf vorzeitig abgebrochen werden kann, muss darauf geachtet werden, dass im weiteren Verlauf keine Verzweigungen mehr stattfinden können. Eine Verzweigung erfordert, dass ein Simulationslauf zumindest bis zum Zeitpunkt der Verzweigung gerechnet wird. Am Verzweigungspunkt wird dann der aktuelle Zustand des Simulationslaufs geklont. Der geklonte Zustand wird dann entsprechend den Angaben für den neuen Zweig geändert und ist dann Anfangszustand eines neuen Simulationsprozesses.

Nach Auswertung der Fachliteratur kommen als geeignete Methoden insbesondere ein auf Hidden-Markov-Modellen (HMM) basierendes Verfahren sowie ein probabilistisches Clusteranalyse-Verfahren in Frage. Beide Verfahren wurden bereits erfolgreich im Zusammenhang mit der Berechnung eines dynamischen Ereignisbaums bzw. mit der Durchführung von Monte-Carlo-Simulationen angewendet.

4.3.1 Hidden-Markov-Modell

Ein Hidden-Markov-Modell (HMM) ist ein stochastisches Modell, das sich durch zwei Zufallsprozesse beschreiben lässt. Der erste Zufallsprozess entspricht einer Markov-Kette. Die Zustände dieser Markov-Kette sind verborgen und deshalb unbekannt. Stattdessen liefert ein zweiter Zufallsprozess beobachtbare Zustände gemäß einer Wahrscheinlichkeitsverteilung, die von den verborgenen Zuständen der ersten Markov-Kette abhängt.

Anwendung finden HMMs in der Mustererkennung bei der Verarbeitung sequentieller Daten, z. B. einer physikalischen Messreihe. Dazu werden die Modelle so konstruiert, dass die verborgenen Zustände Merkmalen wie z. B. bestimmten System- bzw. Prozesszuständen, entsprechen, die nur in Form der sequentiellen Daten (z. B. Temperaturverläufe) beobachtet werden können.

Eine weitere Anwendung besteht darin, für ein gegebenes HMM solche Sequenzen zu identifizieren, die sehr wahrscheinlich von diesem HMM erzeugt wurden. Beispielsweise kann ein HMM, das mit Temperaturverläufen aus kontrollierbaren Ereignisabläufen trainiert wurde, dazu eingesetzt werden, um weitere kontrollierbare Ereignisabläufe zu identifizieren /RAB 89/, /ZAM 13/, /ZHO 14/.

4.3.2 Fuzzy Clustering

Bei der Clusteranalyse wird eine Menge von Zuständen in homogene Gruppen bzw. Klassen unterteilt. Die Zuordnung der Zustände zu den Clustern kann deterministisch oder probabilistisch erfolgen. Bei deterministischen Verfahren werden die einzelnen Zustände immer nur einem Cluster zugeordnet. Bei probabilistischen Verfahren (Fuzzy Clustering) werden die Daten mit einer zwischen 0 und 1 liegenden Vertrauenswahrscheinlichkeit den Clustern zugeordnet.

Eine Anwendung des probabilistischen Clusteranalyseverfahrens im Rahmen von MCDET-Analysen kann z. B. darin bestehen, dass die Endzustände aller gerechneten Ereignisabläufe zu einem Störfall- bzw. Unfallszenario entsprechend ihrer Ähnlichkeit zum aktuell sich entwickelnden Ereignisablauf gewichtet werden. Der Endzustand mit dem größten Gewicht wird dann dem sich gerade entwickelnden Ereignisablauf zugeordnet /ZIO 12/.

5 Benutzeroberfläche des Analysewerkzeugs MCDET

Das Konzept der grafischen Benutzeroberfläche sieht eine Menü- und Dialog-gesteuerte Kommunikation vor (ähnlich wie beim Analysewerkzeug SUSA für Unsicherheits- und Sensitivitätsanalysen /KLO 15/). Die Programmierung der Oberfläche wird mit der sogenannten Qt-Bibliothek erfolgen. Qt ist eine C^{++}-Klassenbibliothek für die plattformübergreifende Programmierung grafischer Benutzeroberflächen. Im Folgenden werden wichtige Bestandteile der grafischen Benutzeroberfläche näher beschrieben.

Die Gestaltung der abgebildeten Dialogfenster erfolgte mit Visual Basic (VB) .Net.

Abb. 5.1 zeigt den Entwurf des Hauptfensters des MCDET-Analysewerkzeugs. Die Menüleiste gibt die aufeinanderfolgenden Schritte einer MCDET-Analyse wieder. Eine Analyse beginnt mit der Definition eines neuen bzw. der Auswahl eines bestehenden Projekts. MCDET-Projekte sind durch die Erweiterung ‘md‘ gekennzeichnet.

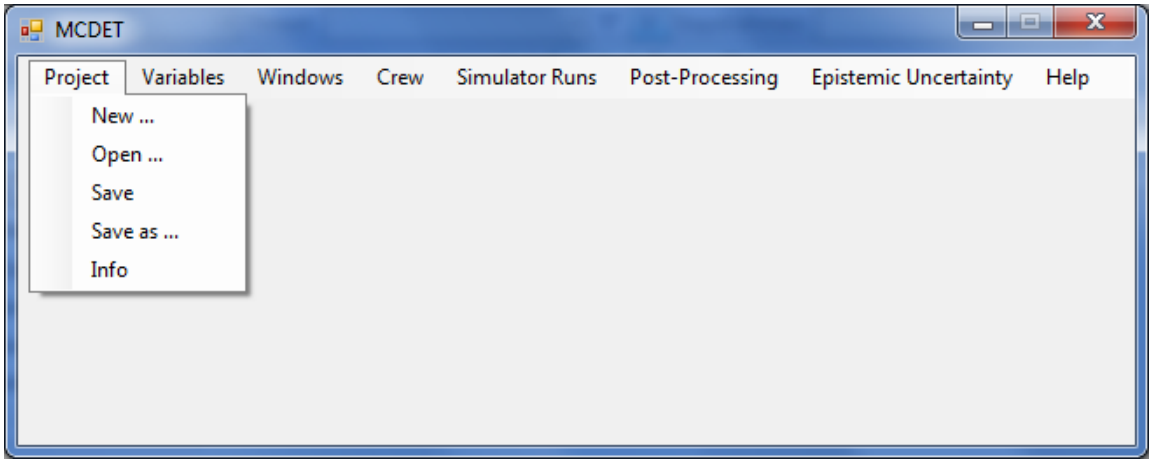


Abb. 5.1 Menü ‘Project‘ der MCDET-Menüleiste

5.1 Menü ‘Project‘

Jede MCDET-Analyse beginnt mit der Auswahl des Menüs ‘Project‘ in Verbindung mit dem Menüpunkt ‘New…‘ oder ‘Open…‘. Sobald ‘New…‘ gewählt wird, ist ein Verzeichnis auszuwählen und ein neuer Projektname mit der Erweiterung ‘md‘ zu spezifizieren, der in dem ausgewählten Verzeichnis gespeichert werden soll. Wenn ‘Open…‘ gewählt wird, ist ein bestehendes Projekt mit der Erweiterung 'md‘ aus einem Verzeichnis auszuwählen.

Abgesehen vom Menü ‘Project‘, das immer zuerst ausgewählt werden muss, hängt die Auswahl der weiteren Menüs vom Analysefortschritt des ausgewählten MCDET-Projektes ab.

5.2 Menü ‘Variables‘

Am Anfang einer Analyse erfolgt zunächst die Spezifikation der zu betrachtenden Variablen eines Projekts bzw. einer Anwendung. Dazu wird das Menü ‘Variables‘ (siehe Abb. 5.2) ausgewählt. Unterschieden wird zwischen Simulator-Variablen, Crew-Variablen, Zeitzustands-Variablen (‘Auxiliary‘) sowie aleatorischen und epistemischen Variablen (siehe Abschnitt 3.1).

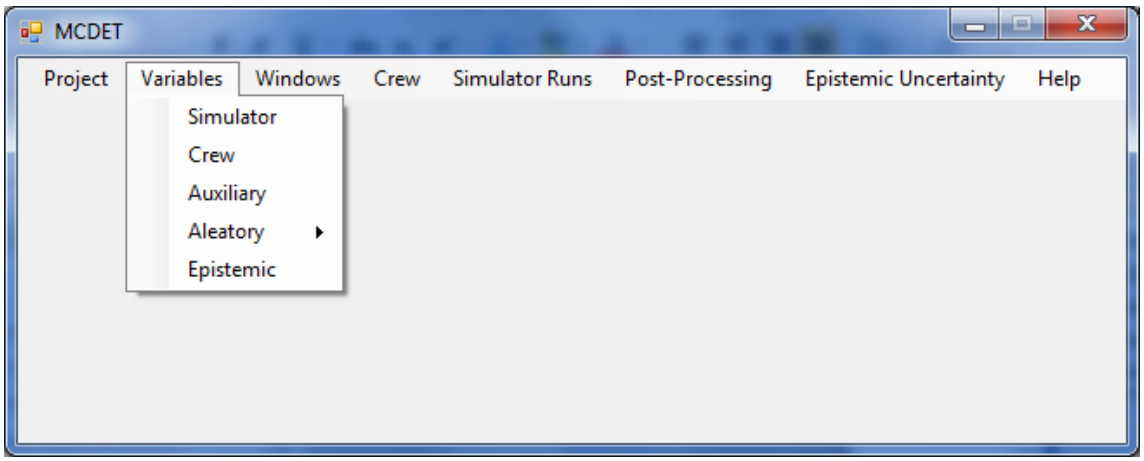


Abb. 5.2 Menü ‘Variables‘ der MCDET-Menüleiste

Wenn im Menü ‘Variables‘ der Menüpunkt ‘Simulator‘ ausgewählt wird, wird anschließend das Dialogfenster in Abb. 5.3 angezeigt. In diesem Fenster sollen die Simulator-Variablen eines Anwendungsfalles spezifiziert werden. Die einzige erforderliche Angabe in dem Fenster ist die Angabe der ID (kurze Kennzeichnung) der jeweiligen Simulator-Variablen. Alle anderen Angaben (zu Name, Einheit, Referenzwert, Dokumente und Bemerkungen) sind optional. Mit den Angaben auf der rechten Seite des Fensters können ganze Blöcke von Simulator-Variablen eingegeben werden (z. B. SimVar1, SimVar2, …, SimVar10). Dazu werden einfach die Anzahl der Variablen, die zu einem Block gehören (z. B. 10), der Prefix (z. B. SimVar) und die Nummer der ersten Variablen im Block (z. B. 1) spezifiziert.

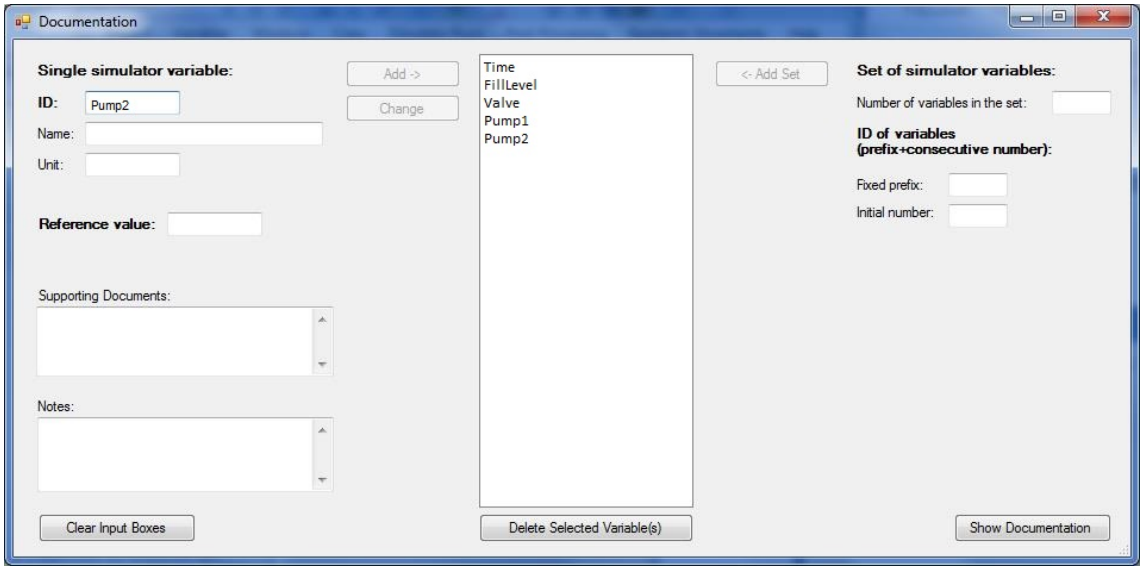


Abb. 5.3 Dialog zur Eingabe von Simulator-Variablen (vgl. Abb. 3.1)

Nach Auswahl des Menüpunkts ‘Epistemic‘ im Menü ‘Variables‘ wird das Dialogfenster in Abb. 5.4 angezeigt. Hier sollen die epistemischen Variablen eines Anwendungsfalles spezifiziert werden. Obligatorische und optionale Angaben sind dieselben wie im Fenster für die Simulator-Variablen.

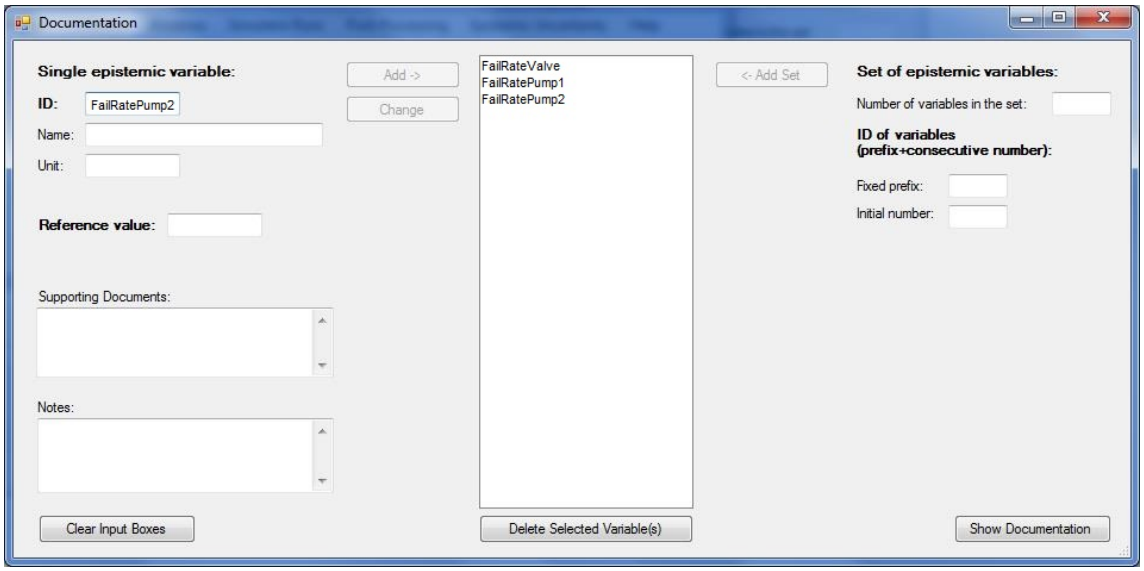


Abb. 5.4 Dialog zur Eingabe von epistemischen Variablen (vgl. Abb. 3.3)

Wenn im Menü ‘Variables‘ der Menüpunkt ‘Aleatory‘ ausgewählt wird, erscheint ein ähnliches Fenster wie in Abb. 5.4, um die aleatorischen Variablen eines Anwendungsfalles zu spezifizieren.

5.3 Menü ‘Windows‘

Unter dem Menü ‘Windows‘ erfolgt die Spezifikation der einzelnen Zeit-Zustandsfenster eines Anwendungsfalles. Vier verschiedene Typen von Zeit-Zustandsfenstern können spezifiziert werden und zwar das ‚Auswahlfenster‘ (Sampling), das ‚Verzweigungsfenster‘ (Transition) das ‘Funktionsfenster‘ (Function) und das ‘Absorbierende Fenster‘ (Absorbing). Diese Fenster enthalten die Bedingung für ihre Aktivierung sowie Angaben zu den Aktionen, die bei Zutreffen der Bedingungen durchzuführen sind (vgl. Abschnitt 3.2.1).

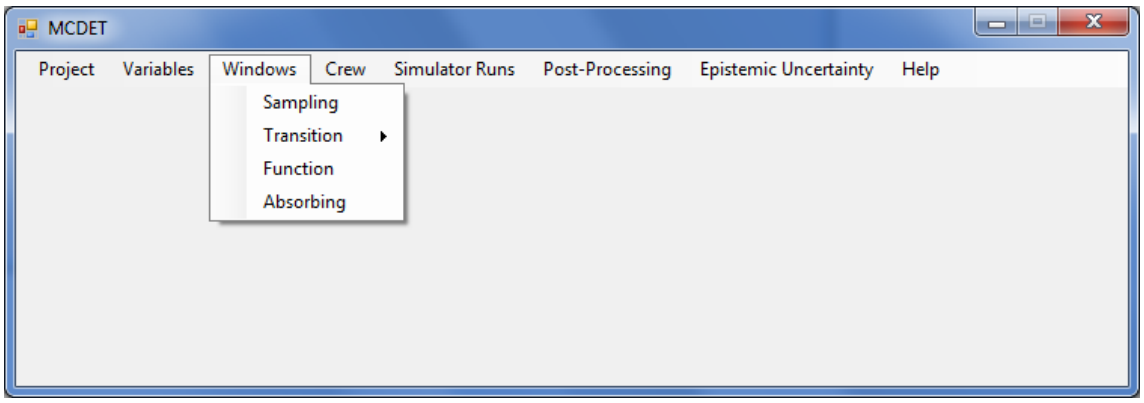


Abb. 5.5 Menü ‘Windows‘ der MCDET-Menüleiste

Ein Dialogfenster zur Definition eines Auswahlfensters fragt die in den Abbildungen Abb. 5.6 und Abb. 5.9 beispielhaft dargestellten Informationen ab. Neben der Bedingung für die Aktivierung des Fensters sind Angaben darüber zu machen, für welche aleatorischen Variablen unter der gegebenen Bedingung jeweils ein Wert ausgespielt werden soll und aus welchen Verteilungen (Verteilungstyp und Verteilungsparameter) die jeweiligen Werte ausgespielt werden sollen.

Wird die Bedingung ‘Init‘ wie in den Abbildungen Abb. 5.6 und Abb. 5.7 spezifiziert, so erfolgt die Auswahl von Werten für die aleatorischen Variablen gleich zu Beginn noch vor dem Start der Simulationsläufe. Wird eine Bedingung wie z. B. in den Abbildungen Abb. 5.8 und Abb. 5.9 definiert, so erfolgt die Werteauswahl erst dann, wenn der Prozesszustand mit dieser Bedingung übereinstimmt.

Die Parameter der Verteilungen der aleatorischen Variablen können als feste Werte eingegeben werden wie in Abb. 5.6 (z. B. Gamma-Verteilung mit den Parametern

0.0001 und 1 über dem Intervall zwischen 0 und 1000) oder als Namen von epistemischen Variablen wie in Abb. 5.7.

--Condition

Variable	Stimulus	Condition
Init		

--Distribution

Variable	Add Variable	Distribution	Parameters		Min	Max
tClose_Valve	Time	Gamma	0.0001	1.	0	1000
tStop_Pump1	Time	Gamma	0.0005	1.	0	1000
tRun_Pump2	Time	Gamma	0.0005	1.	0	1000

Abb. 5.6 Auswahlfenster mit der Bedingung ‘Init‘ und der Berücksichtigung von Verteilungsparametern als feste Werte (vgl. Abb. 3.5)

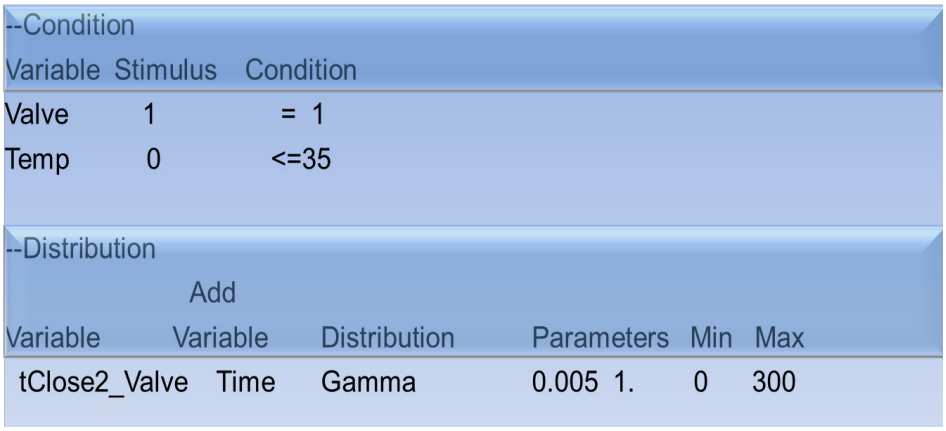
--Condition

Variable	Stimulus	Condition
Init		

--Distribution

Variable	Add Variable	Distribution	Parameters		Min	Max
tClose_Valve	Time	Gamma	FailRateValve	1.	0	1000
tStop_Pump1	Time	Gamma	FailRatePump1	1.	0	1000
tRun_Pump2	Time	Gamma	FailRatePump2	1.	0	1000

Abb. 5.7 Auswahlfenster mit der Bedingung ‘Init‘ und der Berücksichtigung von Verteilungsparametern als epistemische Variable



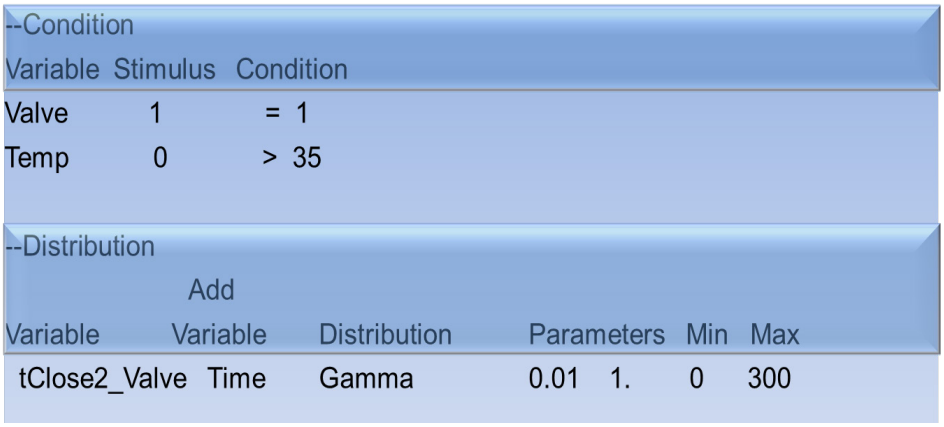
--Condition

Variable	Stimulus	Condition
Valve	1	= 1
Temp	0	<=35

--Distribution

Variable	Add Variable	Distribution	Parameters	Min	Max
tClose2_Valve	Time	Gamma	0.005 1.	0	300

Abb. 5.8 Auswahlfenster für den zufälligen Ausfallzeitpunkt einer Komponente (Beispiel 1)



--Condition

Variable	Stimulus	Condition
Valve	1	= 1
Temp	0	> 35

--Distribution

Variable	Add Variable	Distribution	Parameters	Min	Max
tClose2_Valve	Time	Gamma	0.01 1.	0	300

Abb. 5.9 Auswahlfenster für den zufälligen Ausfallzeitpunkt einer Komponente (Beispiel 2)

Ein Dialogfenster zur Definition eines Verzweigungsfensters fragt die in Abb. 5.10 und Abb. 5.11 beispielhaft dargestellten Informationen ab. Neben den Angaben zur Bedingung für die durchzuführende Verzweigung müssen Informationen darüber geliefert werden, für welche aleatorischen Variablen unter der gegebenen Bedingung jeweils alternative Werte betrachtet werden sollen und mit welcher Wahrscheinlichkeit die alternativen Werte eintreten. Im Verzweigungsfenster in Abb. 5.10 wird neben dem anforderungsgemäßen Schließen des Ventils (Valve = 1) die Variante berechnet, dass das Ventil bei Anforderung mit einer Wahrscheinlichkeit von 0.08 offen ausfällt (Valve = 3).

--Condition

Variable	Stimulus	Condition
Valve	1	= 1

--Transition

Variable	Number of Variants	State of Variant	Probability
Valve	1	3	0.08

Abb. 5.10 Verzweigungsfenster für Ausfall einer Komponente bei Anforderung

--Condition

Variable	Stimulus	Condition
Valve	0	= 1
t	1	>= tClose_Valve

--Transition

Variable	Number of Variants	State of Variant	Probability
Valve	1	0	pClose_Valve

Abb. 5.11 Verzweigungsfenster für Ausfall einer Komponente zu einem zufälligen Zeitpunkt

Ein Dialogfenster zur Definition eines Funktionsfensters fragt die in Abb. 5.12 beispielhaft dargestellten Informationen ab. Neben der Definition einer Bedingung für die Bestimmung des Funktionswertes erfolgt die Festlegung, welche aleatorische Variablen unter der gegebenen Bedingung als Funktionen von anderen Variablen betrachtet werden und wie die Funktionen genau definiert sind.

--Condition

Variable	Stimulus	Condition
Valve	1	= 0

--Function

Variable	Function
nValveOpen	nValveOpen + 1

Abb. 5.12 Funktionsfenster für Anzahl der Anforderungszyklen (vgl. Abschnitt 3.2.2)

Ein Dialogfenster zur Definition eines Absorbierenden Fensters fragt die in Abb. 5.13 dargestellten Informationen ab. Es erfordert lediglich die Definition einer Bedingung für die Beendigung eines Simulationslaufs.

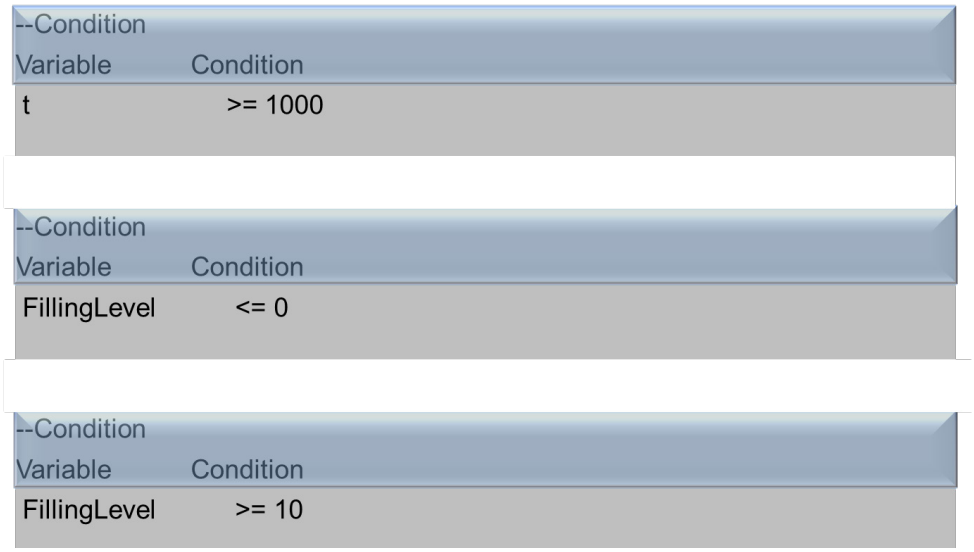


Abb. 5.13 Absorbierende Fenster mit Bedingungen für die Beendigung eines Simulationslaufs

5.4 Menü ‘Crew‘

Bei der Auswahl des Menüs ‘Crew‘ wird das Eingangsfenster zur Erstellung der Eingabedatensätze für das Crew-Modul zur Simulation von Handlungsabläufen angezeigt. Für die Anwendung des Crew-Moduls müssen verschiedene Eingabedatensätze erstellt werden, die die notwendigen Informationen zur Simulation der komplexen dynamischen Handlungsabläufe des Personals beinhalten.

Das Crew-Modul liegt bisher nur in einer Prototyp-Version vor, bei der die notwendigen Informationen in codierter Form direkt in die entsprechenden Eingabedatensätze eingetragen werden müssen. Eine Hilfestellung der relativ umfangreichen und zum Teil unübersichtlichen Eingabe, existierte bisher nicht, wodurch die Dateneingabe sehr fehleranfällig war.

Zur Vereinfachung der Dateneingabe und Verringerung der Fehleranfälligkeit bei der Erstellung der notwendigen Eingabeinformationen ist im vorliegenden Projekt eine me-

nügesteuerte Benutzeroberfläche in VB.NET entwickelt worden. Durch die Oberfläche wird dem Benutzer eine möglichst selbsterklärende Hilfestellung zur Verfügung gestellt.

Die nachfolgende Beschreibung der Benutzeroberfläche dient gleichzeitig dazu, die Informationen zu erläutern, die zur Simulation eines Handlungsablaufs mit dem Crew-Modul notwendig sind.

Die Dateneingabe für das Crew-Modul setzt generell voraus, dass möglichst detaillierte Beschreibungen für die zu analysierenden Handlungsabläufe sowie deren Abhängigkeiten vom Prozesszustand und der zu berücksichtigenden stochastischen Einflussfaktoren vorliegen. Speziell dazu sind in bisherigen Anwendungen Schwierigkeiten aufgetreten, die zeigen, dass bei der Modellerstellung und Beschreibung der dynamischen Handlungsabläufe ein erheblicher Weiterentwicklungsbedarf besteht.

Wenn in der MCDET-Menüleiste (vgl. Abb. 5.1) das Menü ‘Crew‘ ausgewählt wird, erscheint das in Abb. 5.14 dargestellte Hauptfenster für das Crew-Modul.

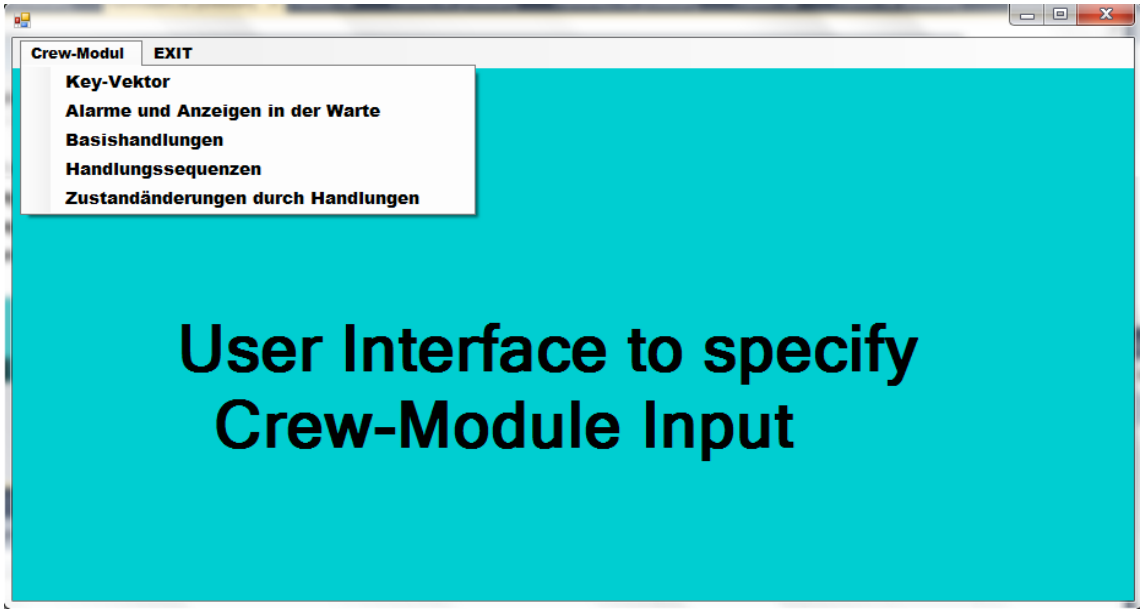


Abb. 5.14 Hauptdialogfenster des Crew-Moduls

Folgende Informationen werden über die Benutzeroberfläche abgefragt:

- Key-Vektor:

  Hier sind alle Parameter zu spezifizieren, die für die durchzuführende Analyse und insbesondere für die Kommunikation zwischen Crew-Modul, dem Probabilistik-

Modul von MCDET und dem zu verwendenden deterministischen Rechencode benötigt werden. Solche Parameter sind z. B.

- Prozessgrößen (z. B. Druck im RDB, Hüllrohrtemperatur, Einspeiserate von Pumpen etc.),
- Zustände von Komponenten (z. B. Pumpe an/aus, Ventil offen/zu, Öffnungsquerschnitt eines geöffneten Ventils etc.),
- Zustände bzw. Indikatoren von Verzweigungen stochastischer Ereignisse (z. B. menschliche oder technische Fehler), die unterschiedliche Handlungsabläufe zur Folge haben.

Der Key-Vektor wird nach jedem Berechnungszeitpunkt aktualisiert und beschreibt den aktuellen Zustand der im Key-Vektor enthaltenen Parameter. Der Key-Vektor steht allen Modulen zur Verfügung und dient somit der Kommunikation zwischen den Modulen.

- Alarme und Anzeigen in der Warte:

  Die Informationen betreffen die für die Analyse relevanten Alarme und Anzeigen, die das Personal in der Warte erhält und aus denen die notwendigen Entscheidungen zur Durchführung der Handlungen abgeleitet werden. Die Alarme und Anzeigen, die in der Warte anfallen, werden von bestimmten Größen und Zuständen des dynamischen Prozesses bestimmt. Deshalb müssen die Prozesszustände spezifiziert werden, die zu den jeweiligen Alarmen und Anzeigen führen. Zu jedem Zeitpunkt wird anhand der spezifizierten Informationen überprüft, ob Alarm- oder Anzeigesituationen vorliegen. Zusätzlich können auch Bedingungen spezifiziert werden, die eine Änderung von Parametern des Key-Vektors im zeitlichen Ablauf bewirken. Beispielsweise kann hier der Ausfall von mehreren Komponenten spezifiziert werden. Diese Bedingung ändert dann z. B. den Stresslevel der beteiligten Personen von ‚normal' auf ‚hoch'.

- Basishandlungen:

  Basishandlungen bzw. Einzelhandlungen sind elementare Handlungen und beschreiben einfache vom Operateur durchzuführende Tätigkeiten. Jeder Basishandlung sind Attribute zugeordnet, die Informationen darüber geben, wer die Basishandlung durchführt, um welche Basishandlung es sich handelt, welche Person

oder welche Komponente durch die Handlung beeinflusst wird und wieviel Zeit zur Ausführung der Basishandlung benötigt wird.

Neben den reinen Basishandlungen, die tatsächliche Aktionen von Personen beschreiben, können auch andere Parameter bzw. Indikatoren definiert werden, die z. B. anzeigen, an welchen Stellen des Handlungsablaufs Zeitzähler, Zeitvergleiche und Verzweigungen auftreten. Diese Parameter müssen jeweils gesondert gekennzeichnet werden. Wenn z. B. eine Basishandlung eine mögliche Fehlerquelle enthält, die in der Analyse berücksichtigt werden soll, führt dies in der Analyse zu einer Verzweigung im Handlungsablauf. Ein Verzweigungspunkt im Handlungsablauf, der sich entweder durch einen menschlichen Fehler oder durch ein beliebiges anderes stochastisches Ereignis ergeben kann, muss gesondert gekennzeichnet werden. Es ist zu beachten, dass die zu einem Verzweigungspunkt gehörigen Fehler- bzw. Verzweigungswahrscheinlichkeiten nicht im Crew-Modul, sondern im Probabilistik-Modul von MCDET spezifiziert werden müssen.

Zu jeder Basishandlung, die eine Aktion einer Person beschreibt, muss eine Schätzung der Zeitdauer angegeben werden, die für die Ausführung dieser Basishandlung benötigt wird. Dabei wird davon ausgegangen, dass die Ausführungszeit einer Basishandlung einer zufallsbedingten Variation unterliegen kann. In diesem Fall wird die aleatorische Unsicherheit der Ausführungszeit durch eine Wahrscheinlichkeitsverteilung angegeben.

Die mit einer Basishandlung ggf. verbundene Fehlerwahrscheinlichkeit ist im Allgemeinen nicht genau bekannt. Die epistemischen Unsicherheit (Kenntnisstandunsicherheit) wird ebenfalls durch eine Wahrscheinlichkeitsverteilung ausgedrückt. Alle zu berücksichtigenden aleatorischen und epistemischen Unsicherheiten bzgl. der Ausführungszeiten und Fehlerwahrscheinlichkeiten von Basishandlungen werden im Probabilistik-Modul von MCDET spezifiziert.

- Handlungssequenzen:

  Die einzelnen Handlungssequenzen werden aus den jeweiligen Basishandlungen zusammengesetzt, die für den Handlungsablauf definiert werden müssen. Zu jeder Handlungssequenz muss eine bestimmte eindeutige Bedingung definiert werden, durch die die jeweilige Handlungssequenz aktiviert wird. Die Handlungssequenzen (Handlungslisten) sind somit eine sequentielle Abfolge von Basishandlungen, die zur Ausführung kommen, wenn eine zur Handlungssequenz zugehörige Bedingung

erfüllt ist. Die Bedingung kann sowohl aus System- und Prozessgrößen, bisher durchgeführten Handlungen oder auch aus Zuständen stochastischer Ereignisse zusammengesetzt sein.

Durch die Zuordnung einer speziellen Bedingung zu einer Handlungssequenz ist es beispielsweise möglich, Handlungsabläufe in Abhängigkeit vom Erfolg bereits durchgeführter Maßnahmen oder auch in Abhängigkeit von kognitiven und ergonomischen Faktoren zu berücksichtigen. Die Flexibilität des Crew-Moduls erlaubt es dem Benutzer, die Handlungsabläufe in einem beliebigen Detaillierungsgrad zu modellieren. Das wesentliche Merkmal des Konzepts ist, dass auch Wechselwirkungen zwischen menschlichen Handlungen, physikalischen Prozess und stochastischen Einflüssen modelliert werden können.

- Zustandsänderungen durch menschliche Handlungen:

  Durch die hier einzugebenden Informationen werden die Zustandsänderungen im System spezifiziert, die durch bestimmte Basishandlungen verursacht werden. Durch die Attribute der Basishandlungen wird in codierter Form angegeben, welche Basishandlung die Änderung eines Komponentenzustands oder ein sonstiges relevantes Ereignis zur Folge hat.

  Wenn die Basishandlung eine Codierung aufweist, bei der eine Änderung des Zustands einer Komponente erfolgen soll, so wird durch die zu spezifizierende Information eine Zustandsänderung der Komponente zu dem Zeitpunkt vorgenommen, an dem die entsprechende Basishandlung ausgeführt wird.

Bei der Erstellung der Eingabeinformationen für das Crew-Modul ist es sinnvoll, sie nach der in Abb. 5.14 aufgeführten Reihenfolge zu erstellen. Der Grund dafür ist, dass Informationen verschiedener Menüpunkte zur Erstellung der Informationen anderer Menüpunkte verwendet werden. So wird z. B. der Key-Vektor zur Erstellung der Informationen für die Alarm-und Anzeigesituation in der Warte, für die Erstellung der Handlungssequenzen und für die Definition von Zustandsänderungen benötigt. Die Informationen der Basishandlungen werden zur Erstellung der Handlungssequenzen verwendet.

Die ersten Informationen sollten sich also auf die Spezifikation der Parameter für den Key-Vektor beziehen. Wurde im aktuellen Projekt noch kein Key-Vektor angelegt, so wird dies erkannt, und es erfolgt eine automatische Voreinstellung des Key-Vektors.

In den Abschnitten 5.4.1 – 5.4.5 erfolgt die Beschreibung der jeweiligen Eingabeinformationen entsprechend der Reihenfolge, wie sie im Hauptmenü des Crew-Moduls in Abb. 5.14 aufgeführt sind.

5.4.1 Dialog zur Erstellung des ‘Key-Vektors‘

Um die für die Analyse relevanten Parameter einzugeben, ist in dem in Abb. 5.14 dargestellten Fenster der Menüpunkt ‘**Key-Vektor‘** auszuwählen. Durch die unter diesem Menüpunkt zu spezifizierenden Parameter wird der Key-Vektor angelegt, der zu jedem Zeitpunkt den aktualisierten Zustand aller wichtigen Größen angibt, die in der Analyse benötigt werden. Dadurch kann die Kommunikation zwischen Crew-Modul, Probabilistik-Modul und Dynamik-Code gewährleistet werden.

In der ursprünglichen Version mussten bei der Anwendung des Crew-Moduls Restart-Informationen definiert werden, was sich als relativ umständlich und aufwändig erwiesen hat. Die Vorgehensweise zur Definition solcher Restart-Informationen bzgl. des Crew-Moduls ist in /PES 06/ beschrieben. Durch das entwickelte Konzept des Key-Vektors kann nun auf die aufwändige Definition von Restart-Informationen bei der Anwendung des Crew-Moduls verzichtet werden. Dabei ist darauf zu achten, dass die Parameter des Key-Vektors als Zeit-Zustandsvariable im Probabilistik-Modul definiert werden.

Wenn der Menüpunkt **Key-Vektor** angeklickt wird, erscheint das Dialogfenster zur Eingabe der für die Analyse relevanten Parameter. Wurde für die Analyse noch kein Key-Vektor spezifiziert, so werden zunächst einige Elemente als Voreinstellung angelegt. Dazu erscheint zunächst das in Abb. 5.15 dargestellte Dialogfenster, in dem die Eingabe für die Anzahl und die Kurzbezeichnungen der Personen, die an einem Handlungsablauf beteiligt sind, erfolgen soll.

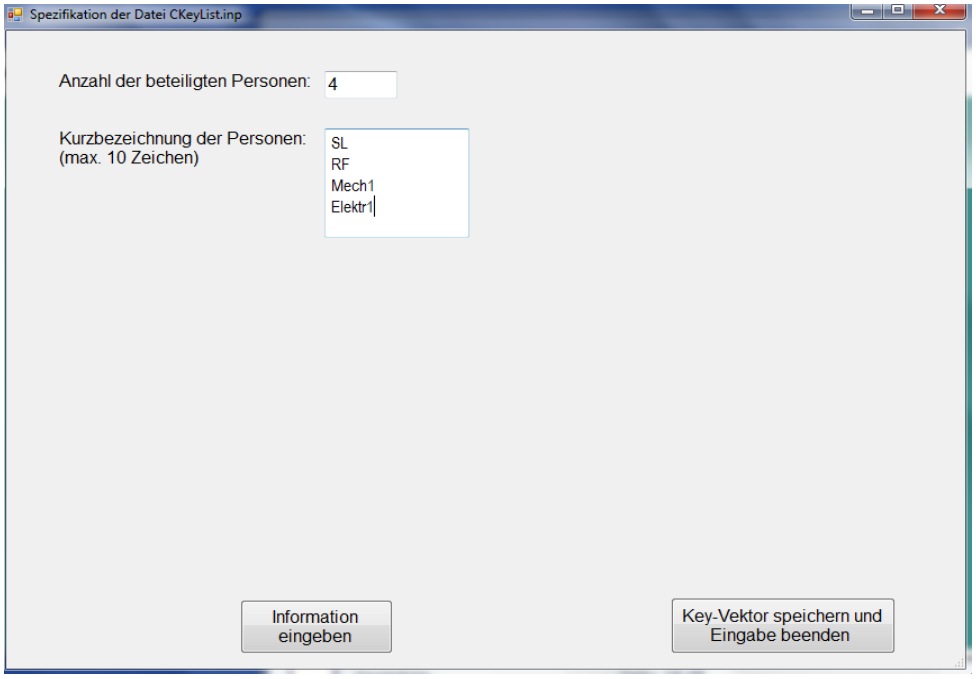


Abb. 5.15 Dialog zur Spezifikation der an einer Handlung beteiligten Personen

Wenn der Benutzer die Eingabe zur Anzahl der Personen vergisst oder einen Wert < 1 eingibt, erscheint die Mitteilung:

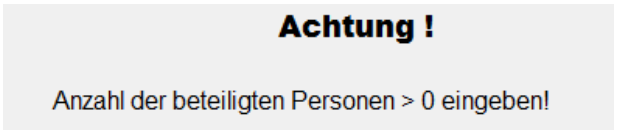


Abb. 5.16 Mitteilung zur Anzahl der beteiligten Personen

In Abhängigkeit von der Anzahl der Personen werden feste Plätze im Key-Vektor für die Parameter ‘tOP1‘, …, ‘tOPn‘ reserviert. Diese Parameter werden zur Steuerung des Ablaufs der Handlungsmaßnahmen benötigt und geben diejenigen Zeiten an wie lange die jeweiligen Personen mit einer Handlung beschäftigt sind bzw. wann sie zur Durchführung neuer Handlungen frei sind.

Wenn als Anzahl der beteiligten Personen ein Wert > 0 eingegeben wurde, können vom Benutzer optional beliebig zu wählende Kurzbezeichnungen für die Personen angegeben werden, wie z. B. SL für Schichtleiter, RF für Reaktorfahrer, Mech1 für den **ersten** Mechaniker und Elektr1 für den ersten Elektriker.

In Abb. 5.15 wurden vier Personen mit ihren Kurzbezeichnungen angegeben. Bei der Betätigung des Buttons **‘Information eingeben‘** werden für die spezifizierten vier Personen die Parameter ‘tSL‘, ‘tRF‘, ‘tMech1‘ und ‘tElektr1‘ auf den Plätzen 12 – 15 des Key-Vektors automatisch angelegt. Wenn keine Kurzbezeichnungen angegeben werden, erhalten die Personen die standardmäßig festgelegten Namen ‘tOP1‘, ‘tOP2‘, …, ‘tOPn‘.

Der Key-Vektor enthält als Default-Einstellung zunächst 20 Parameter. Die Default-Einstellung des Key-Vektors (siehe Abb. 5.17) beinhaltet Elemente, die mit ‚free‘ gekennzeichnet sind und Elemente, die bereits mit Parametern belegt sind und mit einem * gekennzeichnet sind. Die mit * gekennzeichneten Parameter dürfen innerhalb des Key-Vektors aus programmtechnischen Gründen nicht auf andere Plätze des Key-Vektors verschoben werden. Die Plätze im Key-Vektor, die mit ‚free‘ bezeichnet sind, können mit beliebigen Parametern belegt werden.

Neben der Default-Einstellung des Key-Vektors werden gleichzeitig die Eingabefelder zur Spezifikation der Parameter in den Key-Vektor aktiviert und die Felder zur Eingabe der Anzahl und Kurzbezeichnungen der beteiligten Personen deaktiviert. Diese Ansicht ist in Abb. 5.17 dargestellt.

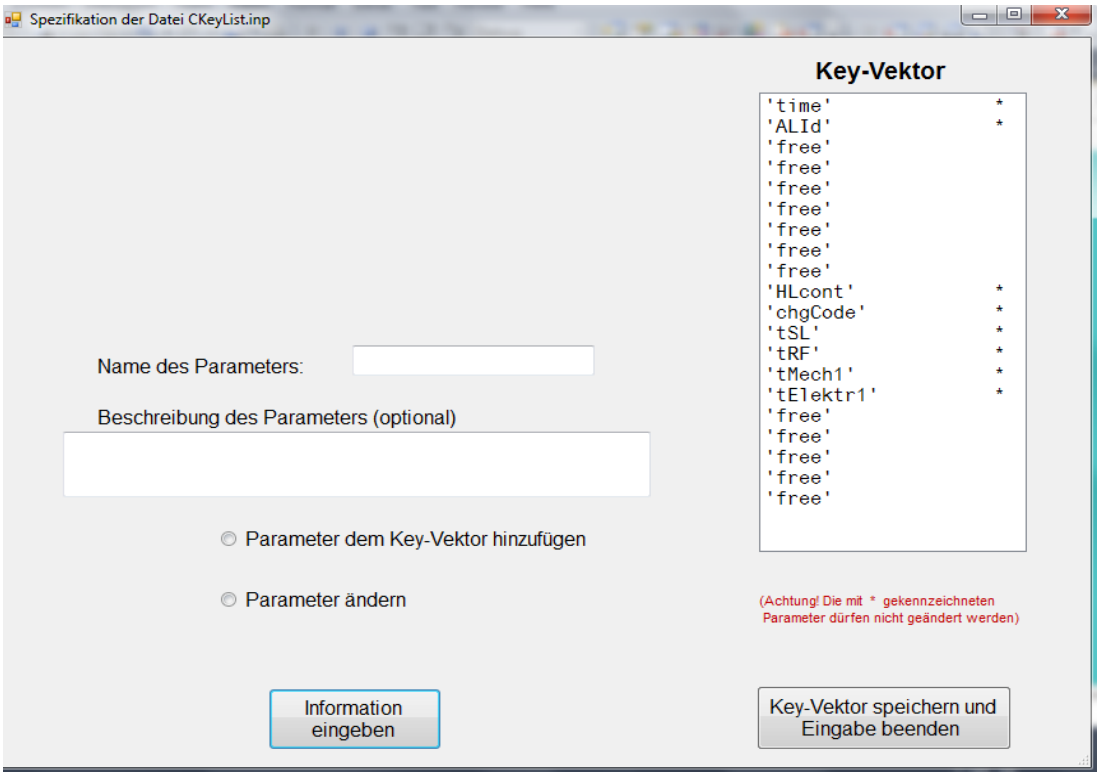


Abb. 5.17 Dialog zur Spezifikation der Parameter im Key-Vektor

Die Eingabefelder zur Spezifikation der Parameter des Key-Vektors umfassen den Namen des Parameters, eine optionale Kurzbeschreibung des Parameters zu Dokumentationszwecken und eine Auswahlmöglichkeit, ob der Parameter dem Key-Vektor hinzugefügt werden soll, oder ein bestehender Parameter des Key-Vektors geändert werden soll.

Diejenigen Größen, die in jeder Analyse eines menschlichen Handlungsablaufs mit dem Crew-Modul vorkommen, werden durch eine automatische Voreinstellung an fest vorgegebene Plätze des Key-Vektors angelegt. Dadurch soll dem Benutzer eine Hilfestellung geliefert und gleichzeitig eine häufig vorkommende Fehlerquelle beseitigt werden. Außerdem kann durch die Tatsache, dass diese Parameter bei jeder Anwendung des Crew-Moduls an der gleichen Stelle des Key-Vektors stehen, die programmtechnische Umsetzung erheblich vereinfacht werden. Bei diesen voreingestellten Größen handelt es sich um die Parameter:

- ***time**:* Zeitpunkt, an dem sich das Gesamtsystem befindet,
- ***ALId**:* Identifikationsnummer von Alarmen bzw. Anzeigen, bei denen das Crew-Modul zur Simulation der zugehörigen Handlungsabläufe aufgerufen wird. Die Spezifikation der Alarm- bzw. Anzeigebedingungen, unter denen die zugehörigen Handlungsabläufe aktiviert werden, erfolgt unter dem Menüpunkt ‚Alarme und Anzeigen in der Warte'.
- ***HLcont**:* Handlungsabläufe werden durch mehrere Teilsequenzen beschrieben, denen jeweils eine Identifikationsnummer zugeordnet wird. Der Parameter '*HLcont'* gibt an, bei welcher Teilsequenz der Handlung aufgesetzt werden muss, um den Handlungsablauf je nach Prozessbedingungen fortführen zu können. Der Parameter '*HLcont'* wird während des Programmablaufs automatisch belegt und wird ausführlicher im Abschnitt 5.4.4 erläutert.
- ***ChgCode***: Durch den aktuellen Wert dieses Indikators wird dem Programm mitgeteilt, welche Zustandsänderung im System durch eine Personalhandlung vorgenommen werden muss, bzw. welche Information vom Prozess benötigt wird, um den weiteren Handlungsablauf bestimmen zu können. Der Parameter 'ChgCode*'* wird während des Programmablaufs automatisch belegt und wird ausführlicher im Abschnitt 5.4.5 erläutert.

- ***tOP1*, …, *tOPn***: Zeiten der Operateure, bis zu welchem Zeitpunkt sie beschäftigt sind bzw. wann die jeweiligen Operateure für nachfolgende Handlungen wieder zur Verfügung stehen.

Größen, die vom Anwender in der Analyse benötigt werden und im Key-Vektor noch nicht festgelegt sind, können beliebig benannt und auf einen beliebigen freien Platz im Key-Vektor angelegt werden. Außerdem können dem Key-Vektor beliebig viele Parameter hinzugefügt werden, auch dann, wenn noch freie Plätze vorhanden sind. In den meisten Anwendungen wird der Benutzer nicht von Anfang an alle benötigten Parameter kennen. Deshalb kann er den Key-Vektor sukzessive erweitern, indem er zusätzliche Parameter dem Key-Vektor hinzufügt. Es können alle Parameter im Key-Vektor, die nicht mit einem * gekennzeichnet sind, geändert werden. Zu jedem geänderten oder hinzugefügten Parameter kann der Benutzer eine kurze Beschreibung angeben, welche Funktion der Parameter in der Analyse hat. Die Parameterbeschreibung ist optional.

Durch die Auswahloption **Parameter ändern** hat der Benutzer die Möglichkeit, die im Key-Vektor spezifizierten Parameter, nach seinem Wunsch anzuordnen oder umzubenennen, sofern es sich nicht um Parameter handelt, die mit * gekennzeichnet sind. Die Belegung der freien Plätze im Key-Vektors erfolgt, indem

- der zu belegende freie Platz in der Liste des Key-Vektors ausgewählt wird,
- der Name des Parameters in die entsprechende Maske **‘Name des Parameters‘** und optional eine Beschreibung des Parameters eingetragen wird,
- die Option **‘Parameter ändern‘** ausgewählt wird und
- der Button **‘Information eingeben‘** angeklickt wird.

Wird z. B. im Key-Vektor das dritte Element (‘free‘) ausgewählt sowie der Name des Parameters eingegeben (z. B. DE1-FH) und anschließend die Auswahloption **‘Parameter ändern‘** gewählt – diese Situation ist in Abb. 5.18 dargestellt -, dann wird durch Anklicken des Buttons **‘Information eingeben‘** für das dritte Element des Key-Vektors der Name DE1-FH angezeigt.

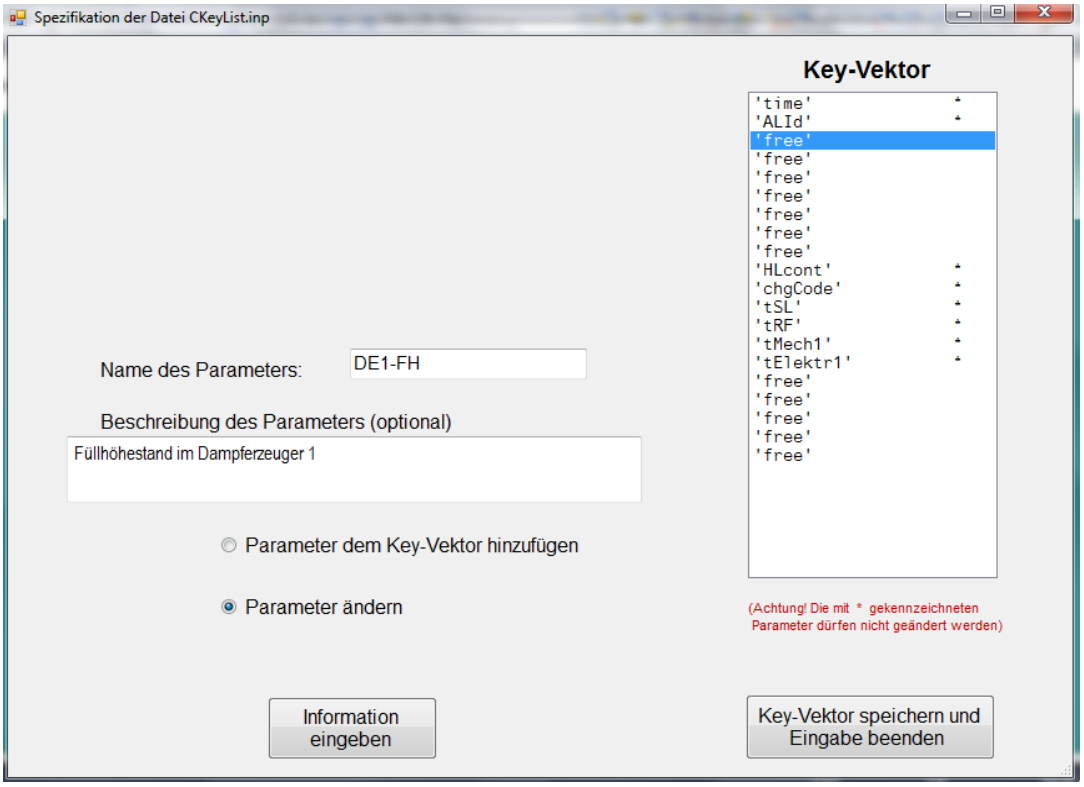


Abb. 5.18 Dialog zur Änderung eines Parameters im Key-Vektor

Unabhängig davon, ob alle freien Plätze des Key-Vektors bereits mit Parametern belegt sind oder nicht, kann der Benutzer über die Auswahloption **‘Parameter dem Key-Vektor hinzufügen‘** beliebig viele zusätzliche Parameter dem Key-Vektor hinzufügen. Dazu muss

- der Name des Parameters in die entsprechende Maske eingegeben werden (z. B. sHKP1),
- optional die Beschreibung des Parameters (z. B. Zustand der Hauptkühlmittelpumpe 1: 0-aus 1-an) in die dafür vorgesehene Maske eingetragen werden,
- die Option **‘Parameter dem Key-Vektor hinzufügen‘** ausgewählt sowie
- der Button **‘Information eingeben‘** angeklickt werden.

Diese Eingabe ist in Abb. 5.19 dargestellt. Man erkennt in dieser Darstellung, dass der vorher eingegebene Parameter DE1-FH als drittes Element im Key-Vektor enthalten ist. Der neu hinzuzufügende Parameter sHKP1 wird an das Ende des Key-Vektors angehängt.

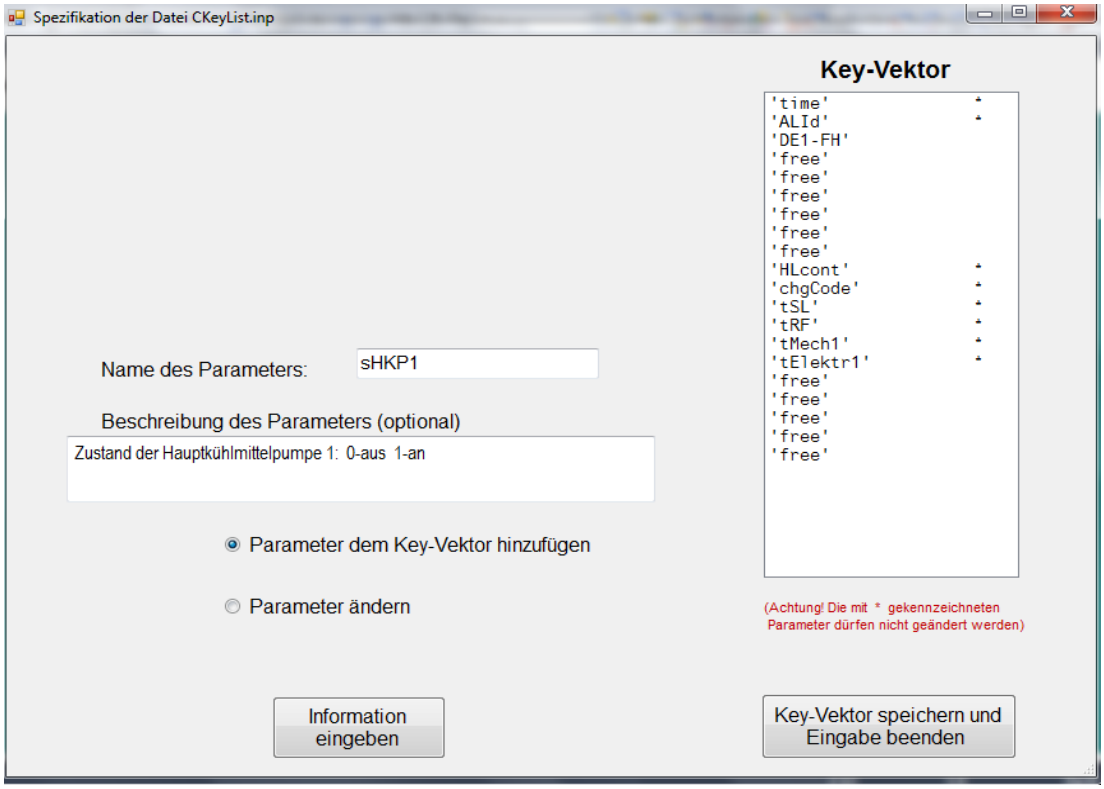


Abb. 5.19 Dialog zum Hinzufügen eines Parameters in den Key-Vektor

Durch Anklicken des Buttons **'Key-Vektor speichern und Eingabe beenden'** werden alle Parameter, die in der Oberfläche durch den Key-Vektor angezeigt werden, gespeichert. Wird das aktuelle Projekt wieder geöffnet, werden die bis dahin gespeicherten Informationen automatisch geladen, so dass der aktuelle Zustand des Key-Vektors anzeigt wird und entsprechend erweitert bzw. modifiziert werden kann.

Die Spezifikation des Key-Vektors wird in der Regel ein iterativer Prozess sein, der durch die grafische Benutzeroberfläche vereinfacht und übersichtlicher gestaltet wird.

Zur Vereinfachung der Parameterkennzeichnung wird folgende Konvention in der Namensgebung der Parameter des Key-Vektors vorgeschlagen:

- Parameter, die eine Zeitangabe enthalten, werden mit einem voranstehenden t (time) gekennzeichnet, z. B. *'tSL'.* Der Parameter *'tSL'* beschreibt beispielsweise den Zeitpunkt, bis wann der Schichtleiter mit einer Aufgabe beschäftigt ist bzw. ab wann er zur Durchführung von weiteren anstehenden Aufgaben wieder zur Verfügung steht.

- Parameter, die den Zustand einer technischen Komponente beschreiben, der durch eine menschliche Handlung verändert werden kann, werden mit einem voranstehenden ‘s‘ (englisch: state für Zustand) gekennzeichnet. So beschreibt z. B. ‘s*HKP1‘*den Zustand der Hauptkühlmittelpumpe 1 (HKP1), wobei sHKP1 = 1 den Zustand beschreibt, dass die Hauptkühlmittelpumpe 1 in Betrieb ist, und sHKP1 = 0 den Zustand, dass die Hauptkühlmittelpumpe 1 abgestellt ist. Parameter, die als Indikatorvariable zur Kennzeichnung eines Verzweigungspunktes für die alternativen Ausprägungen diskreter Zufallsvariablen dienen, werden mit einem voranstehenden ‘b_‘ (branching point) gekennzeichnet. Der Parameter ‘b_*SV1-ÖQ‘* kann beispielsweise die zufälligen diskreten Ausprägungen des Öffnungsquerschnitts des Sicherheitsventils 1 angeben, wie z. B. b_*SV1-ÖQ = 1* –> Öffnungsquerschnitt von Sicherheitsventil 1 ist 100 %, *b_SV1-ÖQ = 2* –> Öffnungsquerschnitt von Sicherheitsventil 1 ist 80 %, b_*SV1-ÖQ = 3* –> Öffnungsquerschnitt von Sicherheitsventil 1 ist 60 % *etc.*

Die Einhaltung der Namenskonvention dient zur besseren Strukturierung und Interpretation der im Key-Vektor verwendeten Parameter. Die Einhaltung der Namenskonventionen bzgl. des Zustands von Komponenten mit einem voranstehenden s vor dem Komponentennamen und bzgl. einer Verzweigungsvariablen mit einem voranstehenden b_ ist aus programmtechnischen Gründen zwingend einzuhalten.

5.4.2 Dialog zur Erstellung der Eingabeinformationen für relevante Alarme und Anzeigen in der Warte

System- und Prozesszustände, die bestimmte Handlungsmaßnahmen zur Folge haben, werden der Schichtmannschaft in der Warte über bestimmte Alarme und Anzeigen mitgeteilt. D. h., durch die Alarme und Anzeigen in der Warte wird die Schichtmannschaft über eine Anlagensituation informiert, die bestimmte Handlungsmaßnahmen erfordern.

Die aktuellen System- und Prozesszustände, die vom angewendeten Rechenprogramm zur Verfügung gestellt werden und in die entsprechenden Parameter des Key-Vektors übertragen werden, werden zu jedem Zeitschritt daraufhin überprüft, ob Bedingungen für einen Alarm oder eine Anzeige vorliegen oder nicht. Sobald eine Bedingung für einen Alarm oder Anzeige vorliegt, wird die Identifikationsnummer des Alarms bzw. der Anzeige in der Variablen *‘ALId‘* des Key-Vektors gespeichert. Durch die Belegung

der Variablen *‘ALId‘* mit einem Wert > 0 wird das Crew-Modul angewiesen, den Handlungsablauf zu aktivieren, der mit der Identifikationsnummer *‘ALId‘* verbunden ist.

Nachdem die notwendigen Parameter im Key-Vektor spezifiziert worden sind, kann die entsprechende Eingabeinformation bzgl. der relevanten Alarme und Anzeigen erstellt werden. Dazu ist im Hauptmenü des Crew-Moduls (siehe Abb. 5.14) der Menüpunkt ‘**Alarme und Anzeigen in der Warte‘** auszuwählen. Nach der Auswahl erscheint das in Abb. 5.20 dargestellte Fenster.

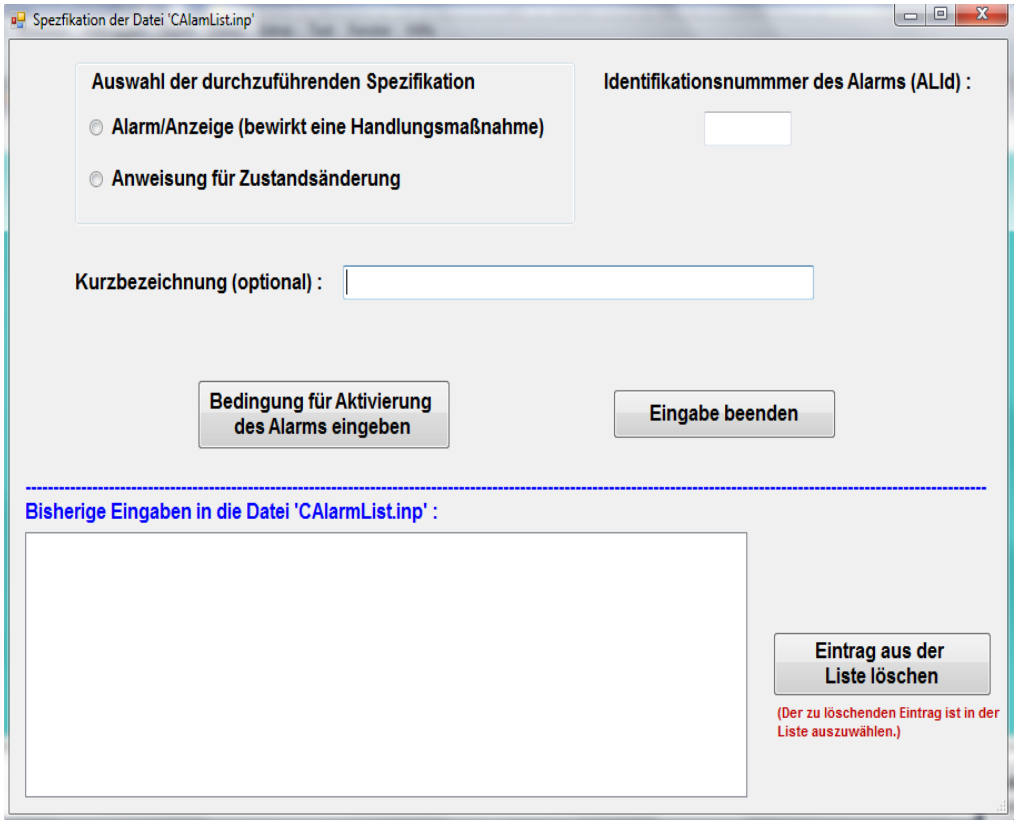


Abb. 5.20 Dialog zur Spezifikation der Alarm- bzw. Anzeigenbedingungen

Es können sowohl Alarm- und Anzeigesituationen mit nachfolgender Handlungsmaßnahme als auch Situationen ohne nachfolgende Handlungsmaßnahme aber mit nachfolgender Zustandsänderung von Systemgrößen spezifiziert werden. Deshalb ist im Dialog in Abb. 5.20 zunächst anzugeben, welche der beiden Spezifikationen erfolgen soll. Wenn die Option **‘Alarm/Anzeige‘ (bewirkt eine Handlungsmaßnahme)** ausgewählt wird, muss eine Identifikationsnummer für die entsprechende Alarm- bzw. Anzeigesituation vergeben werden. Bei der Eingabe wird automatisch überprüft, ob ein Wert für die Identifikationsnummer schon vergeben wurde. Mit jeder Identifikationsnummer ist eine bestimmte Alarm- bzw. Anzeigesituation mit einem bestimmten Handlungsab-

lauf verbunden. Optional kann eine Kurzbeschreibung zur Alarm- und Anzeigesituation eingegeben werden.

Wenn nach Auswahl der Option **‘Alarm/Anzeige‘ (bewirkt eine Handlungsmaßnahme)** die Eingabe der Identifikationsnummer vergessen wurde, erscheint eine entsprechende Meldung

Abb. 5.21 Meldung zur Eingabe der Identifikationsnummer

Zur Information und besseren Übersicht für den Benutzer werden im unteren Teil des Dialogfensters in Abb. 5.20 alle Anweisungen aufgelistet, die bisher eingegeben und gespeichert wurden. Der Benutzer hat hier die Möglichkeit, bereits eingegebene Informationen zu löschen. Dazu hat er die zu löschende Information aus der Liste auszuwählen und den Button **‘Eintrag löschen‘** zu betätigen.

Das Konzept der Benutzerführung soll im Folgenden anhand eines Beispiels erläutert werden. Angenommen, in der Analyse soll die Anzeige- bzw. Alarmierungssituation berücksichtigt werden, dass die Kriterien zur Durchführung einer Dampferzeuger-Druckentlastung (DE-DrE) im Rahmen der Notfallmaßnahme ‚Sekundärseitiges Druckentlasten und Bespeisen‘ anstehen. Nach Notfallhandbuch sind die Bedingungen zur Durchführung einer Dampferzeuger-Druckentlastung gegeben, wenn

- die Kühlmitteleintrittstemperatur in den Reaktordruckbehälter (KMT) ≥ 583 K oder
- der Druckhalter-Füllstand (DH-FH) > 9,5 m oder
- das Druckhalter-Abblaseregelventil (DH-ARV) mehrmals (z. B. mehr als dreimal) aktiviert worden ist.

Zur Spezifikation dieser Prozessbedingungen werden die Parameter ‘KMT‘, ‘DH-FH‘‘ und ‘DH-ARV‘ im Key-Vektor benötigt, auf die zu jedem Zeitpunkt die aktuellen Werte der entsprechenden Prozessgrößen aus dem deterministischen Rechencode geschrieben werden. Für das Beispiel sollen die Parameter ‘KMT‘, ‘DH-FH‘‘ und ‘DH-ARV‘ auf die freien Plätze 16, 17 und 18 des Key-Vektors geschrieben werden. Dies kann über den Dialog zur Eingabe des Key-Vektors unter Verwendung der Auswahloption **‘Parameter ändern‘** erfolgen, wie dies im vorangegangenen Abschnitt 5.4.1 erläutert wurde.

Im Beispiel wird für die zu spezifizierende Alarm- bzw. Anzeigesituation die frei wählbare Identifikationsnummer 1 eingesetzt und als optionale Kurzbezeichnung ‘Einleitung der Maßnahmen zur DE-DrE‘ eingegeben. Nach dem Klick auf den Button **‘Bedingung für Aktivierung des Alarms eingeben‘** erscheint das in Abb. 5.22 dargestellte Dialogfenster, in dem die Bedingungen und Anweisungen für die betreffende Alarm- bzw. Anzeigesituation spezifiziert werden können.

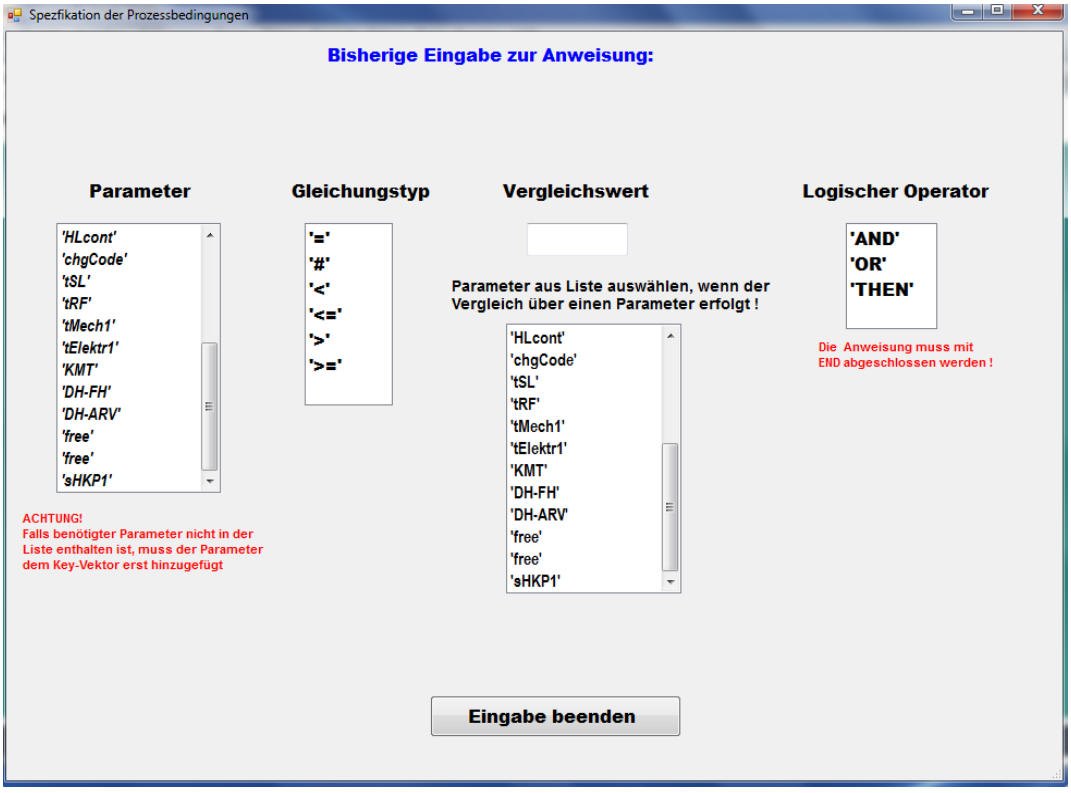


Abb. 5.22 Dialog zur Spezifikation von Alarmen, Anzeigen und Zustandsänderungen

In dem in Abb. 5.22 dargestellten Dialog werden verschiedene Auswahllisten zur Spezifikation der gewünschten Bedingungen und Anweisungen aufgeführt. Die Auswahlliste unter der Überschrift **‘Parameter‘** beinhaltet alle im Key-Vektor spezifizierten Pa-

rameter. Falls ein für die einzugebende Bedingung benötigter Parameter noch nicht im Key-Vektor enthalten ist, muss das aktuelle Fenster über **‘Eingabe beenden‘** verlassen werden und der fehlende Parameter im Key-Vektor nach dem im Abschnitt 5.4.1 beschriebenen Vorgehen eingegeben werden. Erst dann kann der entsprechende Parameter für die Spezifikation der Bedingung verwendet werden.

Die Auswahlliste unter der Überschrift ‘**Gleichungstyp‘** liefert eine Liste von Vergleichsoperatoren, aus denen die entsprechenden Operatoren ausgewählt werden können, die für die einzugebende Anweisung benötigt werden. Die Syntax der zu spezifizierenden Anweisungen lautet:

IF Bedingungsteil **THEN** Anweisungsteil **END.**

Der Bedingungsteil kann aus mehreren Bedingungsblöcken bestehen, die durch die logischen Operatoren ‘AND‘ oder ‘OR‘ verknüpft sein können. Der Bedingungsteil wird durch den logischen Operator ‘THEN‘ abgeschlossen, wobei gleichzeitig der Anweisungsteil beginnt. Der Anweisungsteil kann wiederum aus mehreren Anweisungsblöcken bestehen, die jedoch nur mit dem logischen Operator ‘AND‘ verknüpft sein können. Durch ‚END‘ wird die Spezifikation der Anweisung abgeschlossen.

In Abhängigkeit davon, an welcher Stelle der Anweisung man sich befindet, werden unterschiedliche Auswahlmöglichkeiten von Vergleichsoperatoren angeboten. Im Bedingungsteil werden alle Vergleichsoperatoren zur Auswahl angeboten. Im Anweisungsteil wird nur das Gleichheitszeichen ‘ = ‘ zur Auswahl angeboten, da im Anweisungsteil nur konkrete Zuordnungen erfolgen können.

Unter der Überschrift ‘**Vergleichswert‘** hat der Benutzer die Möglichkeit, einen konkreten Zahlenwert für die zu spezifizierende Gleichung bzw. Ungleichung in das dafür vorgesehene Eingabefeld einzugeben oder einen Parameter aus der darunter stehenden Liste auszuwählen, der in der Gleichung bzw. Ungleichung zu verwenden ist. Die Angabe eines konkreten Zahlenwertes und gleichzeitige Auswahl eines Parameters aus der Liste führt zu der Meldung, dass entweder nur ein Zahlenwert oder nur ein Parameter angegeben werden darf.

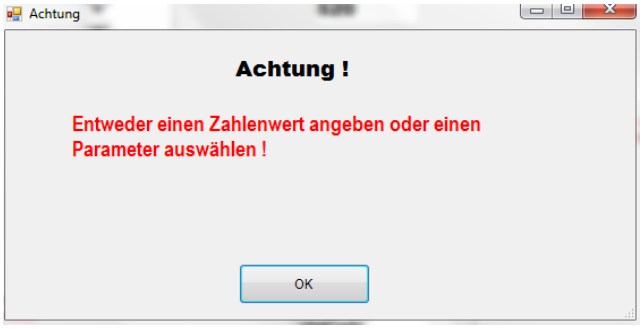


Abb. 5.23 Meldung zur Eingabe von Zahlenwert oder Parameter

Die Liste unter der Überschrift **'Logischer Operator'** liefert eine Auswahl von logischen Operatoren, mit der die Bedingungsblöcke bzw. Anweisungsblöcke verknüpft sind. In Abhängigkeit davon, an welcher Stelle der Anweisung man sich befindet, werden unterschiedliche Auswahlmöglichkeiten der logischen Operatoren angeboten. D. h., die jeweiligen unterschiedlichen Auswahllisten geben nur diejenigen logischen Operatoren an, die an der betreffenden Stelle sinnvoll sind. Befindet man sich z. B. im Bedingungsteil der Anweisung, so werden als logische Operatoren 'AND', 'OR' und 'THEN' zur Auswahl angeboten. Im Anweisungsteil werden dagegen nur die logischen Operatoren 'AND' und 'END' zur Auswahl angeboten, da nur diese Operatoren im Anweisungsteil sinnvoll eingesetzt werden können.

Im Folgenden soll beispielhaft die Anweisung für die Situation eingegeben werden, dass das Personal die Maßnahmen zur Dampferzeuger-Druckentlastung (DE-DrE) einleitet. Dafür muss die Prozessbedingung vorliegen, dass KMT ≥ 583 K oder DH-FH > 9,5 m oder DH-ARV ≥ 4 ist. Wenn die Variable 'ALId' gleich 1 ist (s. o.), soll eine dieser drei Bedingungen durch den Prozess gegeben sein.

Über das in Abb. 5.22 dargestellte Fenster muss folgende Anweisung eingegeben werden:

IF KMT ≥ 583 OR DH-FH > 9.5 OR DH-ARV ≥ 4 THEN ALId = 1 END

Der Bedingungsteil besteht aus drei Bedingungsblöcken:

1. Bedingungsblock: KMT ≥ 583 OR
2. Bedingungsblock: DH-FH > 9.5 OR
3. Bedingungsblock: DH-ARV ≥ 4 THEN

Der Anweisungsteil besteht aus einem Anweisungsblock: ALId = 1 END.

Über den Dialog werden die einzelnen Bedingungs- und Anweisungsblöcke sequentiell eingegeben, wie nachfolgend demonstriert wird. In dem in Abb. 5.22 dargestellten Fenster wird in der Liste **‘Parameter‘** das Element ‘KMT‘ ausgewählt, in der Liste **‘Gleichungstyp‘** der Operator ‘> = ‘ und als **'Vergleichswert‘** gibt man in das entsprechende Textfeld den Wert 583 ein. In der Liste **‘logischer Operator‘** wird das Element ‘OR‘ ausgewählt. Sobald in der Liste **‘logischer Operator‘** das Element ‘OR‘ angeklickt wird, werden die angegebenen Informationen des ersten Bedingungsblocks gespeichert und es erscheint automatisch das in Abb. 5.24 dargestellte Fenster zur Eingabe des zweiten Bedingungsblocks. Dabei befinden sich alle Listen und Eingabefelder wieder auf ihrem Ausgangszustand. Gleichzeitig wird der Benutzer im oberen Teil des Fensters über die bisherige Eingabe der Anweisung (hier die Angaben des ersten Bedingungsblocks) informiert. Nachdem in analoger Weise der zweite und dritte Bedingungsblock eingegeben wurde, muss noch der Anweisungsteil der Anweisung spezifiziert werden.

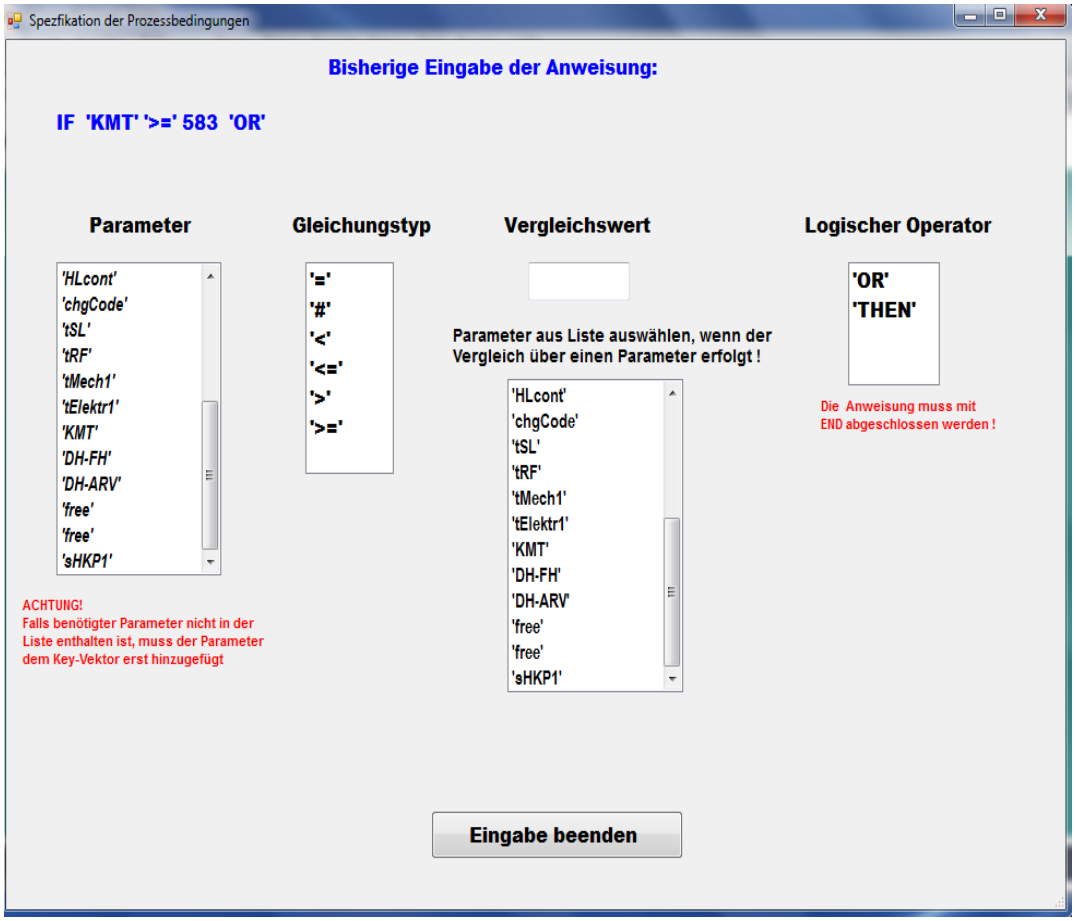


Abb. 5.24 Dialog nach Eingabe des ersten Bedingungsblocks

Das Ende des Bedingungsteils und der Anfang des Anweisungsteils der Anweisung wird durch den logischen Operator ‘THEN‘ gekennzeichnet. D. h., sobald im dritten Bedingungsblock der logische Operator ‘THEN‘ ausgewählt wurde, erscheint die bisherige Eingabe des Bedingungsteils im oberen Teil des Fensters.

Zur Eingabe des Anweisungsteils werden die entsprechenden Angaben aus den Listen ausgewählt und der Vergleichswert in das entsprechende Textfeld eingegeben. Da der Anweisungsteil nur aus einem Anweisungsblock besteht, wird in der Liste ‘**logischer Operator‘** das Element ‘END‘ ausgewählt, um die Eingabe der Anweisung abzuschließen. Sobald das Element ‘END‘ in der Liste angeklickt wird, erscheint das in Abb. 5.25 dargestellte Dialogfenster.

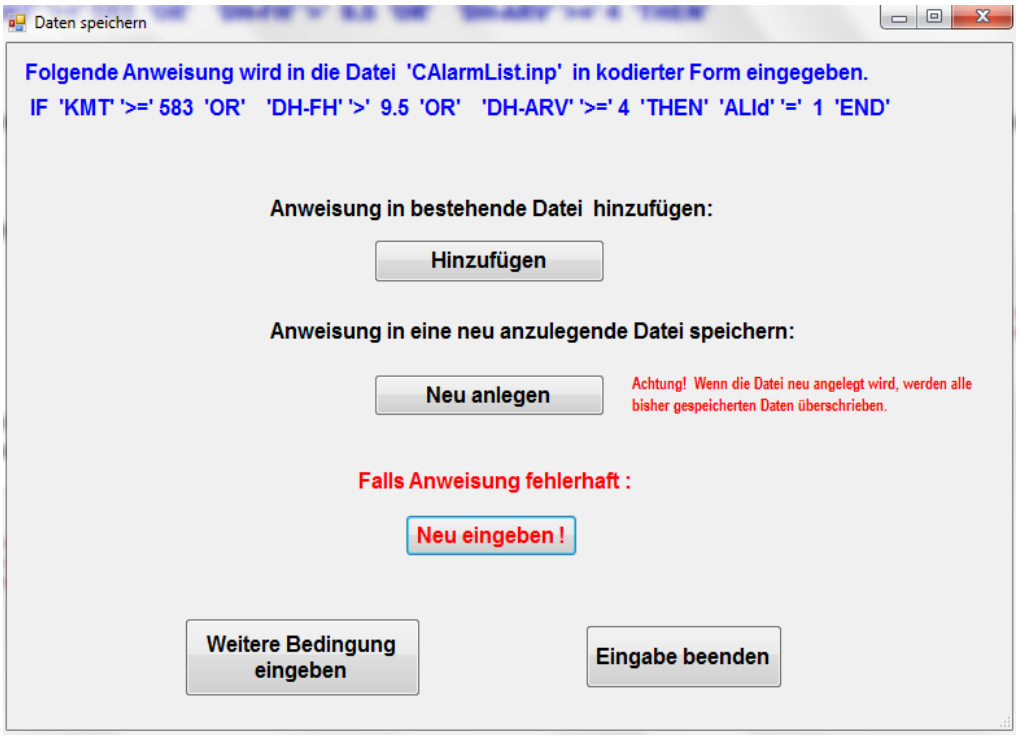


Abb. 5.25 Dialog zum Speichern der eingegebenen Anweisung

Im oberen Teil des Fensters in Abb. 5.25 wird die Information der eingegebenen Anweisung dargestellt. Hier hat der Benutzer die Gelegenheit, die eingegebene Anweisung zu überprüfen. Stellt der Benutzer fest, dass die Anweisung fehlerhaft ist, so hat er die Möglichkeit, die Anweisung neu in korrigierter Form einzugeben, indem er auf den Button **‘Neu eingeben!‘** drückt. Damit wird die bisher eingegebene Anweisung gelöscht, und es erscheint erneut das in Abb. 5.25 dargestellte Fenster. Damit kann der Benutzer seine Anweisung erneut in korrigierter Form eingeben.

Wurden Bedingungen und Anweisungen korrekt eingegeben, hat der Benutzer die Möglichkeit, seine Angaben zu bereits vorhandenen Angaben hinzuzufügen. Dies wird der Fall sein, wenn er schon im Vorfeld verschiedene andere Anweisungen spezifiziert und gespeichert hat. Durch Klick auf den Button **'Hinzufügen'** werden die im oberen Teil des Fensters der Abb. 5.25 dargestellten Angaben zu den bereits vorhandenen Angaben hinzugefügt.

Wenn bisher noch keine Angaben gemacht wurden oder die bisherigen Angaben gelöscht und durch die neuen Angaben ersetzt werden sollen, so ist der Button **'Neu anlegen'** zu drücken. In diesem Fall wird eine neue Datei angelegt und gespeichert.

Durch Anklicken des Buttons **'Weitere Bedingung eingeben'** im Fenster in Abb. 5.25 erscheint erneut das in Abb. 5.22 dargestellte Fenster zur Eingabe weiterer Spezifikationen. Durch Betätigung des Buttons **'Eingabe beenden'** kann der Benutzer die Eingabe beenden. Zusätzliche Anweisungen können jedoch zu einem beliebigen späteren Zeitpunkt durchgeführt werden.

Da die Informationen zu den spezifizierten Alarm- und Anzeigesituationen zu jedem Zeitschritt überprüft werden, kann genau festgestellt werden, wann die einzelnen Bedingungen zur Einleitung bestimmter Handlungsmaßnahmen vorliegen. Sobald in dem zugrundeliegenden Beispiel eine der drei Bedingungen erfüllt ist, wird dem Parameter ALId durch den Anweisungsteil der Wert 1 zugewiesen. Durch die Zuordnung ALId = 1 wird dem Crew-Modul mitgeteilt, dass die Kriterien zur Einleitung der Maßnahmen zur Dampferzeuger-Druckentlastung anstehen. Unmittelbar oder mit einer Verzögerungszeit wird dann die Simulation des Handlungsablaufs zur Dampferzeuger-Druckentlastung durch das Crew-Modul gestartet.

Bei komplizierter zusammengesetzten Bedingungsblöcken muss die Eingabe der Informationen in einer etwas anderen Form erfolgen. Im Folgenden wird die Eingabe verschiedener komplexer Anweisungsformen beschrieben.

Anweisungsform 1:

IF (Bed1 AND Bed2) OR (Bed1 AND Bed3) THEN Anweisung END

Da die AND- und OR-Verknüpfungen in gemischter Form vorliegen muss die Eingabe der unterschiedlichen Bedingungsblöcke getrennt über die Oberfläche erfolgen. Zuerst wird der Bedingungsblock

IF (Bed1 AND Bed2) THEN Anweisung END

in der oben beschriebenen Form eingegeben. Wenn es sich um eine Anweisung für eine Alarm- bzw. Anzeigesituation handelt, muss eine Identifikationsnummer eingegeben werden, z. B. ALId = 3. Analog wird danach der zweite Bedingungsblock

IF (Bed1 AND Bed3) THEN Anweisung END

ebenfalls in der üblichen Form eingegeben. Da dieser Bedingungsblock die gleiche Handlungsausführung zur Folge hat wie der erste Bedingungsbock, muss für beide Bedingungsblöcke die gleiche Identifikationsnummer eingegeben werden. Da jedoch gleiche Identifikationsnummer nicht vergeben werden dürfen, muss für den zweite Bedingungsblock die ALId = 3.1 eingegeben werden. Dadurch erkennt das Programm, dass dieser Bedingungsblock mit dem zuerst eingegebenen Bedingungsblock zusammenhängt. Sind weitere Bedingungsblöcke mit den obigen beiden durch eine OR-Verknüpfung verbunden, sind die jeweils zugehörigen Identifikationsnummern 3.2, 3.3 usw. für die Variable ‘ALId‘ anzugeben.

Anweisungsform 2:

Liegt die einzugebende Anweisung in der Form

IF (Bed1 OR Bed2) AND (Bed1 OR Bed3) THEN Anweisung END

vor, so muss diese Anweisung zunächst in die dazu logisch äquivalente Anweisungsform 1 transformiert werden. Die logisch äquivalente Form zu

(Bed1 OR Bed2) AND (Bed1 OR Bed3)

ist gegeben durch

(Bed1 AND Bed3) OR (Bed2 AND Bed1) OR (Bed2 AND Bed3)

Diese Anweisungsform entspricht der Anweisungsform 1 und kann in analoger Form separat über

IF (Bed1 AND Bed3) THEN Anweisung END

IF (Bed2 AND Bed1) THEN Anweisung END

IF (Bed2 AND Bed3) THEN Anweisung END

mit den Identifikationsnummern z. B. 6, 6.1 und 6.2 eingegeben werden, die dann der Variablen ‘ALId‘ in dem Anweisungsteil zugeordnet werden müssen.

5.4.3 Eingabeinformationen für Basishandlungen

Basishandlungen sind elementare Einzelhandlungen und beschreiben einfache vom Operateur durchzuführende Tätigkeiten, wie z. B. die Bedienung eines Schaltknopfes, Zurücklegen des Weges vom Maschinenhaus in die Warte oder auch einfache Handlungen im Rahmen der Kommunikation, wie z. B. Schichtleiter weist den Reaktorfahrer an, eine bestimmte Handlung durchzuführen.

Den elementaren Einzelhandlungen (Basishandlungen) sind jeweils Attribute zugeordnet, die eine nähere Beschreibung der Einzelhandlungen erlauben:

- ***IdNr*** – Identifikationsnummer der Basishandlung.
- ***Wer*** – Angabe der Person, die die Basishandlung durchführt.
- ***Beschreibung*** – Kurzbeschreibung der Basishandlung.
- ***HatEffektAuf*** – Angabe der Person oder der technischen Komponente, die durch die Basishandlung beeinflusst wird.
- ***Zeitbedarf*** – Angabe der Zeit, die die Person für die Ausführung der Basishandlung benötigt. Da die Ausführungszeit einer Basishandlung Zufallsschwankungen unterworfen ist, wird die Ausführungszeit einer Basishandlung im Allgemeinen als eine zufällige Größe (aleatorische Unsicherheit) betrachtet. Die aleatorische Unsicherheit der Ausführungszeit wird durch eine Wahrscheinlichkeitsverteilung im Probabilistik-Modul von MCDET spezifiziert. Ausführungszeiten, die nur sehr kleinen zufälligen Variationen unterworfen sind, können auch als feste Größen spezifiziert werden.
- ***Handlungsfehler:*** Dabei kann es sich um einen Auslassungsfehler der Handlung handeln (failure of omission) oder um einen Ausführungsfehler (failure of commission), bei der die Handlung in einer bestimmten Weise fehlerhaft ausgeführt wird. Die bzgl. einer fehlerhaften Handlung zugeordnete Fehlerwahrscheinlichkeit sowie

die Unsicherheit bzgl. der Fehlerwahrscheinlichkeit (epistemische Unsicherheit) ist im Probabilistik-Modul von MCDET zu spezifizieren.

Das Konzept des Crew-Moduls zur Simulation von Handlungsabläufen als dynamischer Prozess besteht im Wesentlichen darin, eine durch menschliche Handlungen durchzuführende Maßnahme in mehr oder weniger kleine Einzelhandlungen (Basishandlungen) zu zerlegen. Je feiner die Zerlegung in Einzelhandlungen erfolgt, desto detaillierter kann der Handlungsablauf der Maßnahme simuliert und bewertet werden. Der Grad, wie fein ein Handlungsablauf in Basishandlungen zerlegt wird, liegt im Ermessen des Benutzers.

Es ist zu beachten, dass sich bei einer zu groben Einteilung in Basishandlungen die Schwierigkeit ergeben kann, dass die bzgl. einer Handlung zu spezifizierenden Attribute nicht eindeutig bestimmt werden können. Wenn z. B. eine Basishandlung zu grob gewählt wird, kann es sein, dass die Handlung Einfluss auf mehrere Personen und Komponenten hat. Die Angabe des Attributs, wer die Handlung durchführt, ist somit nicht mehr eindeutig und führt beim gegenwärtigen Entwicklungsstand des Crew-Moduls zu Komplikationen. Eine sinnvolle Weiterentwicklung des Crew-Moduls bestünde darin, dass eine Handlung gleichzeitig Einfluss auf mehrere Personen oder Komponenten haben kann.

Ein weiterer Nachteil einer zu groben Zerlegung des Handlungsablaufs besteht darin, dass die Abschätzung der Ausführungszeit einer zu komplex aufgebauten Basishandlung schwieriger ist als bei einfachen Basishandlungen und dass eventuell auftretende Wechselwirkungen nicht detailliert berücksichtigt werden können.

Dialog zur Spezifikation von Basishandlungen

In diesem Dialog müssen alle Basishandlungen, die zur Durchführung einer bestimmten Maßnahme relevant sind, definiert werden. Wird im Menü ‘Crew-Modul‘ (vgl. Abb. 5.14) der Menüpunkt **‘Basishandlungen‘** angeklickt, so erscheint das in Abb. 5.26 dargestellte Fenster zur Eingabe der Basishandlungen. Über dieses Fenster können die Attribute der jeweiligen Basishandlungen spezifiziert und gespeichert werden.

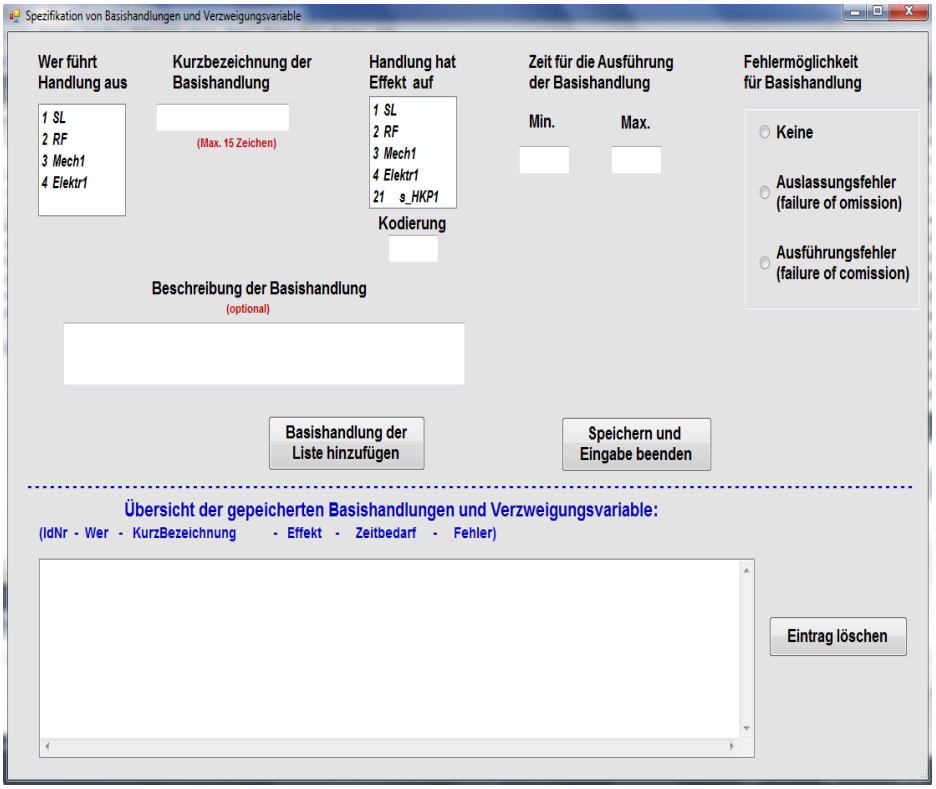


Abb. 5.26 Dialog zur Spezifikation von Basishandlungen

Wenn noch keine Basishandlung spezifiziert wurde, ist die Übersicht der bisher gespeicherten Basishandlungen und Verzweigungspunkte im unteren Teil des Fensters leer. Ansonsten werden hier zur besseren Orientierung des Benutzers alle Basishandlungen und Verzweigungsvariablen aufgeführt, die bisher für den zu analysierenden Handlungsablauf definiert wurden. Die Liste der bereits eingegebenen Basishandlungen wird nach den jeweils ausführenden Personen geordnet. D. h., alle von einer bestimmten Person durchgeführten Basishandlungen werden zusammenhängend in einer Gruppe aufgelistet. Dies dient der Übersichtlichkeit bei der Eingabe von Basishandlungen.

Bei der Spezifikation des Key-Vektors (siehe Abschnitt 5.4.1) wurden die an der Handlung beteiligten Personen bereits spezifiziert. Diese Personen werden in den beiden Listen **‘Wer führt Handlung aus‘** und **‘Handlung hat Effekt auf‘** aufgeführt. Neben den an der Handlung beteiligten Personen werden in der Liste **‘Handlung hat Effekt auf‘** auch alle Parameter aufgeführt, die den Zustand einer Komponente beschreiben und deren Zustand durch bestimmte Handlungen verändert werden kann. Diese Parameter werden mit einem voranstehenden ‘s‘ gekennzeichnet. In Abb. 5.26 ist beispielsweise der Parameter sHKP aufgeführt, der den Zustand der Hauptkühlmittelpum-

pe 1 angibt. Voraussetzung, dass der Parameter sHKP in der Liste **‘Handlung hat Effekt auf‘** aufgeführt wird ist, dass der Parameter sHKP bereits im Key-Vektor spezifiziert wurde.

Aus der Liste **‘Wer führt Handlung aus‘** ist diejenige Person auszuwählen, die die jeweilige Basishandlung durchführt. Aus der Liste **‘Handlung hat Effekt auf‘** ist diejenige Person bzw. die Komponente auszuwählen, die durch die betreffende Basishandlung beeinflusst wird. Für die **Kurzbezeichnung der Basishandlung** sollte möglichst eine strukturierte Namensgebung eingehalten werden, um die Auswahl der Basishandlungen zur Spezifikation der Handlungssequenzen (siehe Abschnitt 5.4.4) zu vereinfachen und übersichtlicher zu gestalten. Ein erster Vorschlag für eine strukturierte Namensgebung der Basishandlungen wird nachfolgend gegeben. Im Allgemeinen ist die Namensgebung dem Benutzer jedoch freigestellt.

Weiter oben wurde bereits erwähnt, dass die benötigte Zeit zur Durchführung einer Basishandlung im Allgemeinen als eine Zufallsgröße betrachtet wird, die einer Wahrscheinlichkeitsverteilung folgt. Diese Verteilungen werden im Probabilistik-Modul von MCDET berücksichtigt. Im Crew-Modul werden lediglich das Minimum und das Maximum der Ausführungszeit für die jeweilige Basishandlung angegeben. Wenn die Zeit für eine Basishandlung keine Zufallsgröße sondern ein fester Wert ist, muss für Minimum und Maximum der Ausführungszeit derselbe feste Wert eingegeben werden.

Ein weiteres, für eine Basishandlung anzugebendes Attribut betrifft die Fehlermöglichkeit für eine Basishandlung. Dabei ist zwischen den Möglichkeiten auszuwählen, dass bzgl. der betreffenden Basishandlung kein Fehler, ein Auslassungsfehler (failure of omission) oder ein Ausführungsfehler (failure of commission) auftritt. Je nachdem, welche Fehlersituation für die jeweilige Basishandlung modelliert werden soll, ist die entsprechende Option zu wählen. Sobald für die Basishandlung entweder ein Auslassungsfehler oder ein Ausführungsfehler gewählt wurde, muss für die Basishandlung eine Verzweigungsvariable im Key-Vektor hinzugefügt werden. Die Basishandlung kann durch den möglichen Fehler nicht mehr fest vorgegeben werden und hat eine oder mehrere Alternativen (Verzweigungen) mit unterschiedlichen Handlungsabläufen zur Folge. Die Kennzeichnung als Verzweigungsvariable erfolgt durch ein vorangestelltes ‘b_‘ an die Kurzbezeichnung der Basishandlung. Die Fehlerwahrscheinlichkeiten und die ggf. spezifizierten Kenntnisstandunsicherheiten der Fehlerwahrscheinlichkeit müssen im Probabilistik-Modul von MCDET spezifiziert werden.

Eine neue Basishandlung muss zunächst durch Betätigen des Buttons **‘Basishandlung der Liste hinzufügen‘** der Übersichtsliste der Basishandlungen im unteren Teil des Fensters der Abb. 5.26 hinzugefügt werden. Alle Basishandlungen, die in der Übersichtsliste aufgeführt sind, werden durch Betätigung des Buttons **‘Speichern und Eingabe beenden‘** gespeichert. Sind die Angaben zu einer Basishandlung fehlerhaft, so kann der Benutzer durch die Auswahl der fehlerhaften Basishandlung und Klick auf den Button **‘Eintrag löschen‘** (vgl. Abb. 5.26) die Basishandlung löschen und erneut in verbesserter Form eingeben.

Beispielhaft sollen im Folgenden drei verschiedene Basishandlungen eingegeben werden:

Basishandlung 1:
Der Reaktorfahrer (RF) liest die Anzeige des Druckhalter-Füllstands (DH-FH) von der entsprechenden Anzeige ab. Das Ablesen der DH-FH hat nur Einfluss auf den RF selbst, da er die entsprechende Information von der Anzeige abliest und in Abhängigkeit dieser Information seine weitere Handlung bestimmt wird. Für das Ablesen der Anzeige benötigt er minimal 4 s und maximal 10 s. Für das Ablesen des DH-FHs von der entsprechenden Anzeige soll kein Fehler unterstellt werden.

Basishandlung 2:
*Der Reaktorfahrer (*RF) informiert den Schichtleiter (SL) über den Füllhöhestand des Druckhalters. Die Information hat Einfluss auf den SL, da er die entsprechende Information vom RF erhält. Für diese Kommunikation benötigt der RF zwischen 5 s und 15 s bei einem Modalwert von 8 s. Mit einer Wahrscheinlichkeit von 0,05 vergisst der RF den SL zu informieren.

***Basishandlung 3*:**
Der Reaktorfahrer drückt den Knopf zum Abschalten der Hauptkühlmittelpumpe 1 (HKP1). Diese Aktion hat Einfluss auf die HKP1, die durch die Handlung abgeschaltet wird. Für diese Aktion soll eine Ausführungszeit zwischen 8 s und 12 s angenommen werden.

An dieser Stelle soll nochmals ausdrücklich betont werden, dass die Wahrscheinlichkeitsverteilungen der Ausführungszeiten und die Fehlerwahrscheinlichkeiten von Handlungen im Probabilistik-Modul von MCDET zu spezifizieren sind. Die Angaben der Ausführungszeiten im Zusammenhang mit der Spezifikation von Basishandlungen dienen

dazu, dass die über das Probabilistik-Modul gelieferten Werte für die jeweiligen Ausführungszeiten den entsprechenden Basishandlungen eindeutig zugeordnet werden können.

Zur Spezifikation der Basishandlung 1 werden die in Abb. 5.27 dargestellten Eingaben vorgenommen. Durch Klick auf den Button **'Basishandlung der Liste hinzufügen'** wird die Basishandlung der Übersichtsliste der Basishandlungen mit einer automatisch zugeordneten Identifikationsnummer (hier: 2001) hinzugefügt und der obere Teil des Fensters wird auf die Default-Stellung zurückgesetzt, so dass die Eingaben für die nächste zu spezifizierende Basishandlung 2 durchgeführt werden können.

Die automatische Zuordnung der Identifikationsnummern folgt einem bestimmen Muster. Dabei werden die Identifikationsnummern der Basishandlungen in Abhängigkeit derjenigen Person hochgezählt, die die jeweilige Basishandlung ausführt. Die erste eingegebene Basishandlung wird vom RF durchgeführt, der im Key-Vektor als zweite Person nach dem SL eingegeben wurde. Die Reihenfolge der im Key-Vektor eingegebenen Personen wird auch in der Liste **'Handlung hat Effekt auf'** angezeigt.

Jede Basishandlung, die der RF durchführt, erhält eine Identifikationsnummer, die mit der Ziffer 2 beginnt. Die Identifikationsnummern der Basishandlungen, die der SL durchführt, beginnen mit der Ziffer 1. Die Identifikationsnummern einer Person werden für jede neu eingegebene Basishandlung, die die Person ausführt, hochgezählt. Da für den RF bisher noch keine Basishandlung eingegeben wurde, wird für die erste eingegebene Basishandlung des RF die Identifikationsnummer 2001 vergeben. Dies wird in Abb. 5.28 deutlich, nach dem die erste Basishandlung der Übersichtsliste durch Betätigung des Buttons **'Basishandlung der Liste hinzufügen'** hinzugefügt wurde.

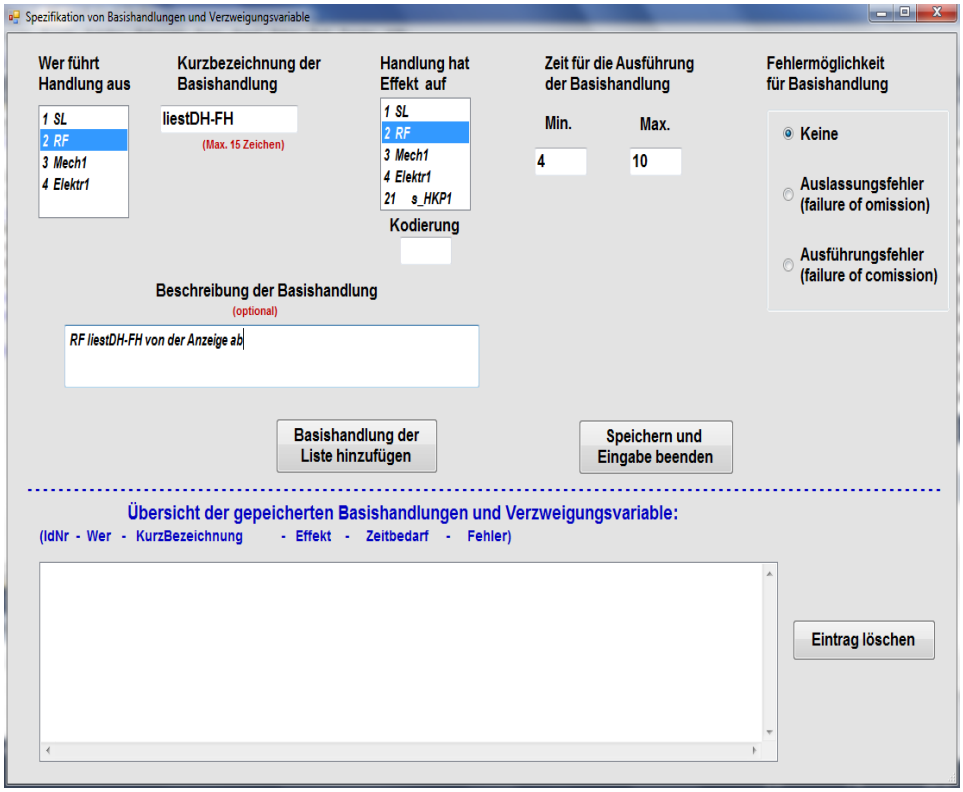


Abb. 5.27 Dialog zur Spezifikation der Basishandlung 1

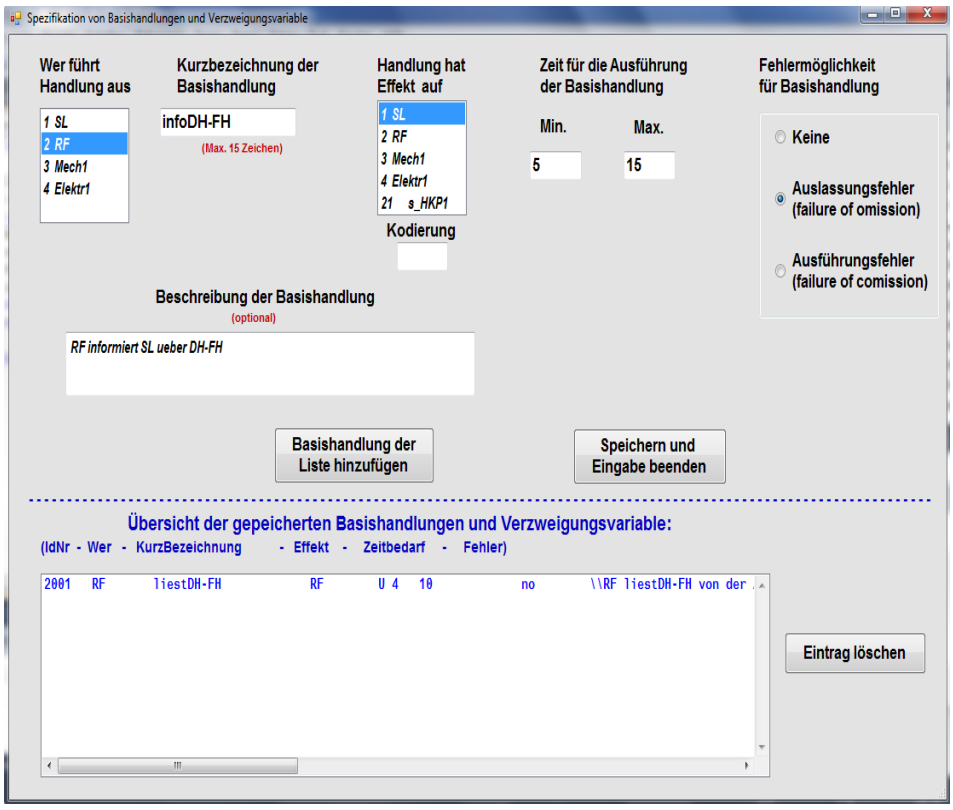


Abb. 5.28 Dialog zur Spezifikation der Basishandlung 2

Die Angaben für die zweite Basishandlung sind im Eingabefenster der Abb. 5.28 dargestellt. Für Basishandlung 2 ist ein Auslassungsfehler möglich. Deshalb muss für die Basishandlung eine Verzweigungsvariable im Key-Vektor definiert werden. Darauf wird der Benutzer explizit hingewiesen. Wenn er den Button **'Basishandlung der Liste hinzufügen'** anklickt, erscheint die Information

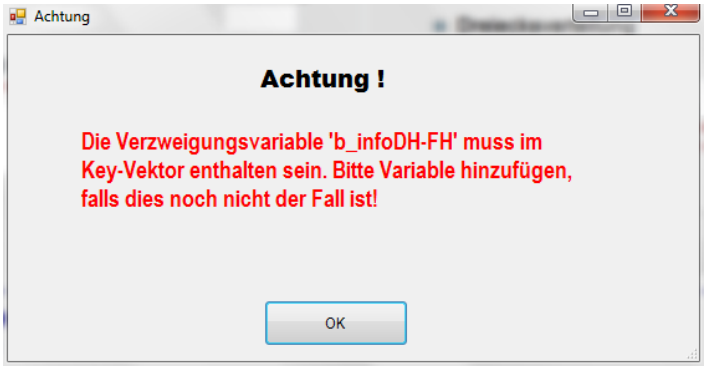


Abb. 5.29 Meldung zur Eingabe der Variable 'b_infoDH-FH'

Wenn die Variable 'b_infoDH-FH' noch nicht im Key-Vektor enthalten ist, so muss dies nachträglich erfolgen, damit die unterschiedlichen Handlungssequenzen in Abhängigkeit davon berücksichtigt werden können, ob der RF dem SL die Information über den DH-FH gegeben hat oder dies vergessen hat. Die nachträgliche Spezifikation der Verzweigungsvariablen 'b_infoDH-FH' im Key-Vektor erfolgt über die in Abschnitt 5.4.1 beschriebene Vorgehensweise.

Bezeichnung für Basishandlung 3 ist 'actHKP1'. Diese Basishandlung hat Einfluss auf eine technische Komponente (HKP1 wird abgestellt) statt auf eine der beteiligten Personen. Im Key-Vektor wurde bereits der Parameter 'sHKP1' definiert, der den Betriebszustand der HKP1 angibt, z. B. 0 – HKP1 ist abgestellt und 1 – HKP1 läuft. In der Liste **'Handlung hat Effekt auf'** werden neben den Personen auch alle Parameter aufgeführt, die den Zustand einer Komponente beschreiben und im Key-Vektor spezifiziert sind. Diese Parameter sind durch ein dem Parameternamen vorangestelltes 's' gekennzeichnet. In der Liste **Handlung hat Effekt auf** ist dem Parameter 'sHKP1' automatisch die Zahl 21 zugeordnet. Diese Zahl gibt an, an welcher Stelle des Key-Vektors der Parameter steht.

Wenn also die Basishandlung 3 eingegeben wird, muss in der Liste **'Handlung hat Effekt auf'** der Parameter 'sHKP1' angewählt werden. Da durch die Basishandlung der

Zustand der Komponente verändert wird, muss dies durch eine bestimmte Codierung gekennzeichnet sein. Die Codierung von Zustandsänderungen von Komponenten, die im Key-Vektor definiert wurden, erfolgt automatisch, indem der Wert im Key-Vektor ein negatives Vorzeichen erhält. Die Codierung der durch die Basishandlung erfolgten Zustandsänderung der HKP1 ist demnach -21. Die Funktionalität der Codierung wird über den in Abschnitt 5.4.5 beschriebenen Dialog spezifiziert. Nach der Eingabe der Basishandlung 3 sieht man in Abb. 5.30 die automatisch zugeordnete Codierung in der Übersichtsliste der Basishandlungen.

Durch Betätigen des Buttons **‘Speichern und Eingabe beenden‘** werden die Informationen der bisher eingegebenen Basishandlungen, die in der Übersichtsliste des Fensters aufgeführt sind, gespeichert. Achtung ! Verlässt der Benutzer den Dialog ohne den Button **‘Speichern und Eingabe beenden‘** anzuklicken, werden die neuen Basishandlungen nicht gespeichert und gehen verloren.

Die Daten der Basishandlungen werden strukturiert gespeichert und angeordnet, so dass man eine schnelle und geordnete Übersicht der Basishandlungen erhält, die für die jeweiligen Operateure bereits spezifiziert worden sind. Diese geordnete Darstellung ist auch für die Spezifikation der Handlungssequenzen hilfreich, die in Abschnitt 5.4.4 beschrieben wird.

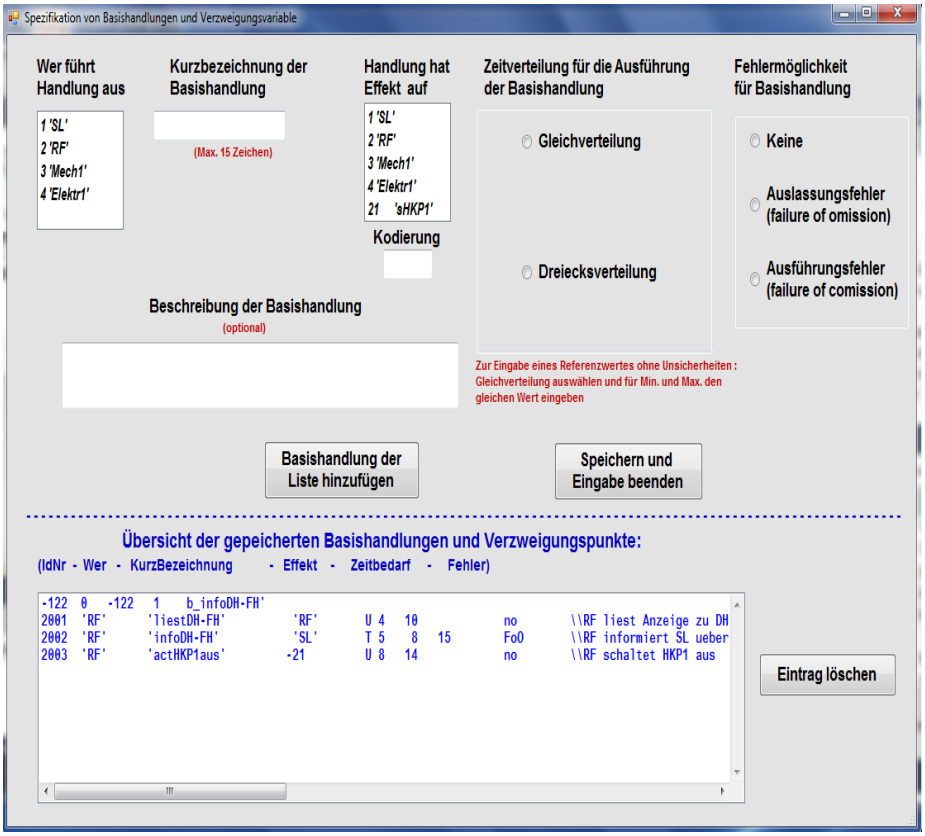


Abb. 5.30 Übersichtsliste nach Spezifikation der Basishandlungen 1 – 3

Wenn der Dialog zur Eingabe der Basishandlungen erneut aufgerufen wird, werden automatisch alle Basishandlungen, die bisher gespeichert wurden, in die Übersichtsliste des Fensters geschrieben (vgl. Abb. 5.30).

Wie im obigen Beispiel beschrieben wurde, müssen für einige Basishandlungen bestimmte Codierungen bzgl. des Attributs **'Handlung hat Effekt auf'** angegeben werden. Solche Codierungen werden für Basishandlungen vergeben,

- die eine Änderung eines Komponentenzustandes zur Folge haben,
- bei denen die ausführende Person Informationen von bestimmten Prozessgrößen benötigt und in Abhängigkeit von der Prozessgröße der weitere Handlungsablauf bestimmt wird (z. B. Operateur kontrolliert den Füllhöhestand des Druckhalters (DH-FH). Wenn DH-FH < 9 m, führt er seine Routinearbeiten fort. Wenn DH-FH > 9.5 m, besteht eine Alarmierungssituation und er muss den SL informieren) oder
- bei Verzweigungsvariablen.

Die Codierung wird benötigt, um dem Programm mitzuteilen, was die jeweilige Basishandlung bewirken soll (vgl. Abschnitt 5.4.5).

Im nachfolgenden Abschnitt soll etwas ausführlicher auf die Notwendigkeit zur Definition von Verzweigungsvariablen bei Handlungsfehlern eingegangen werden.

5.4.3.1 Spezifikation von Verzweigungsvariablen

Wie in der Übersichtsliste in Abb. 5.30 zu erkennen ist, werden neben den Basishandlungen alle Verzweigungsvariable automatisch in der Übersichtsliste aufgeführt. Dies geschieht aus dem Grund, dass Verzweigungsvariable neben den Basishandlungen wesentliche Bestandteile einer Handlungssequenz darstellen.

Verzweigungen ergeben sich im Handlungsablauf unter anderem dadurch, dass durch menschliche Fehler Handlungen unterlassen oder falsch ausgeführt werden können. Eine Handlung wird z. B. mit einer bestimmten Wahrscheinlichkeit p durchgeführt und mit der Wahrscheinlichkeit (1 - p) nicht durchgeführt. Wird beispielsweise die Durchführung einer Handlung unterlassen, so kann die Unterlassung unentdeckt bleiben oder sie wird nach einer bestimmten Zeit bemerkt, so dass die Handlung nach einer gewissen Verzögerungszeit nachgeholt werden kann. Die unentdeckte Unterlassung und die Entdeckung der Unterlassung, die zur nachträglichen Durchführung der Handlung führt, initiieren unterschiedliche Handlungsabläufe mit unterschiedlichen Wahrscheinlichkeiten.

Um alternative Handlungsabläufe modellieren zu können, muss im Key-Vektor eine Verzweigungsvariable spezifiziert werden. Aus programmtechnischen Gründen ist es wichtig, bei der Namensgebung der Verzweigungsvariablen die Konvention einzuhalten, dass dem Namen der Verzweigungsvariablen ein ‘b_‘ (‘b_‘ für branching point) vorangestellt wird. Um Konflikte mit anderen Variablen auszuschließen, ist darauf zu achten, dass die voranstehende Kennzeichnung ‘b_‘ ausschließlich für Verzweigungsvariable in der Namensgebung verwendet wird.

Wenn beispielsweise durch menschliche Fehler alternative Handlungsabläufe bzgl. des Abschaltens der Hauptkühlmittelpumpen möglich sind, so muss im Key-Vektor eine diesbezügliche Verzweigungsvariable angelegt werden. Diese kann z. B. mit dem Namen ‘b_HKPaus‘ bezeichnet werden. Dabei ist die voranstehende Kennzeichnung ‘b_‘ erforderlich. Dadurch erkennt das Programm, dass es sich hier um eine Verzweigung

handelt, bei der unterschiedliche Handlungsabläufe durchzuführen sind. Wurde die Indikatorvariable ‘b_HKPaus‘ im Key-Vektor spezifiziert, so wird als Verzweigungsvariable erkannt und mit einer automatisch zugeordneten Identifikationsnummer gespeichert, wenn der Button **‘Speichern und Eingabe beenden‘** gedrückt wird.

Die Identifikationsnummern, die den Basishandlungen und Verzweigungsvariablen automatisch zugeordnet werden, werden bei der Erstellung der Handlungssequenzen benötigt, wie in Abschnitt 5.4.4 ausführlich beschrieben wird.

5.4.3.2 Strukturierung und Konventionen für die Kurzbezeichnung von Basishandlungen

Um bei der Erstellung von Handlungssequenzen (siehe Abschnitt 5.4.4) die einzubindenden Basishandlungen möglichst einfach und ohne Verwechslungen auswählen zu können, ist es sinnvoll, eine bestimmte Struktur bei den Kurzbezeichnungen der Basishandlungen einzuhalten.

Im Folgenden werden die zur Verfügung gestellten Kategorien von Handlungen beschrieben, die einen Großteil der zu spezifizierenden Basishandlungen abdecken. Zu jeder Handlung, die einer solchen Kategorie zugeordnet werden kann, wird eine Struktur zur Kurzbezeichnung der Basishandlung vorgeschlagen. Die Einhaltung dieser Struktur erleichtert die Einordnung und Übersicht der zu spezifizierenden Basishandlungen, ist jedoch nicht zwingend erforderlich. Der Benutzer hat prinzipiell die freie Wahl, wie er seine spezifizierten Basishandlungen benennen möchte. Die vorgeschlagenen Kategorien von Basishandlungen sind:

- **Ablesen/Lesen:**
  In diese Kategorie fallen Basishandlungen wie z. B. das Ablesen von Anzeigen (z. B. im Rahmen von Kontrollen), Lesen von Textabschnitten im BHB oder NHB, Ablesen von Indikatoren oder Signalen, Lesen von Rückmeldungsformularen etc. Als allgemeine Bezeichnungsstruktur solcher Basishandlungen wird ‘liestxxx‘ vorgeschlagen, wobei für xxx in Kurzform angegeben werden kann, was gelesen oder abgelesen wird. Wenn z. B. die Basishandlung darin besteht, dass die Füllstandsanzeige des Druckhalters abgelesen wird, so kann für diese konkrete Basishandlung beispielsweise die Kurzbezeichnung ‚liestDH-FH‘ verwendet werden. Wer die Anzeigen abliest, ist durch die Person festgelegt, die die Handlung ausführt.

- **Kommunikation:**

  Die Basishandlungen dieser Kategorie bestehen aus Handlungen, bei denen eine Kommunikation zwischen Personen stattfindet. Die Kommunikation kann z. B. darin bestehen, dass eine Person einer anderen Person eine Anweisung gibt, über bestimmte Sachverhalte informiert oder eine Person über einen Sachverhalt befragt, worauf die befragte Person entsprechend antworten muss. Die Einbeziehung der Kommunikation als eine Basishandlung ist deshalb wichtig, da durch die Informationen der Kommunikation bestimmte Handlungen initiiert werden können. Mit einer gewissen Wahrscheinlichkeit können Informationen aber auch falsch interpretiert oder Anweisungen nicht befolgt werden, was ggf. zu unerwünschten Handlungen führen kann.

  Eine Anweisung kann mit der Verpflichtung verbunden sein, eine Rückmeldung zu geben. In dem Wissen, die Anweisung gegeben zu haben, kann z. B. der Schichtleiter beim angewiesenen Operateur nach einer gewissen Zeit nachfragen, um sich zu vergewissern, dass der Operateur die Anweisung ausgeführt hat. Somit kann mit einer Anweisung auch eine Recovery-Handlung verbunden sein, bei der eine nicht befolgte Anweisung erneut gegeben wird oder ein unerwünschter Handlungsablauf durch Fehlinterpretation korrigiert wird.

  Bei einem Informationsaustausch zwischen Personen können die Handlungen von Operateuren beeinflusst werden. So kann es durchaus einen Unterschied im Handlungsablauf in Abhängigkeit davon geben, ob ein Operateur eine bestimmte Information bekommt oder nicht. Wenn die Durchführung einer Handlung beispielsweise erst dann begonnen werden kann, wenn eine andere Handlung abgeschlossen ist so muss der betroffene Operateur darüber informiert werden. Erhält er die Information nicht, so muss er entsprechend nachfragen und kann seine Handlung erst mit einer entsprechenden Verzögerungszeit beginnen.

  Als allgemeine Bezeichnungsstruktur von Kommunikationshandlungen, bei der es sich nur um die Übermittlung von Informationen handelt, wird ‘infoxxx‘ vorgeschlagen, wobei für xxx in Kurzform angegeben werden kann, um welche Information es sich handelt.

  Als allgemeine Bezeichnungsstruktur von konkreten Anweisung wird ‘Anwxxx‘ vorgeschlagen, wobei für xxx in Kurzform angegeben werden kann, um welche Anweisung es sich handelt.

- **Aktion:**

  Die Basishandlungen der Kategorie ‘Aktion‘ bezieht sich auf allgemeine, einfache Handlungsausführungen an Komponenten, wie z. B. Schalter x betätigen, um Ventile zu schließen, oder Knöpfe y_1-y_4 betätigen, um die vier Hauptkühlmittelpumpen abzuschalten. In den meisten Fällen werden die Standardaktionen über Betätigung der entsprechenden Knöpfe und Schalter in der Warte vorgenommen. Mit ‚Aktion‘ können aber nicht nur einfache Handlungsausführungen definiert werden, sondern auch komplexere, aus mehreren Einzelhandlungen zusammengesetzte Handlungen. Bei komplex zusammengesetzten Aktionen ist darauf zu achten, dass hier keine Wechselwirkungen vorkommen. Falls dies nicht vorausgesetzt werden kann, sollte die komplexe Aktion in mehrere Einzelhandlungen zerlegt.

  Als allgemeine Bezeichnungsstruktur solcher Aktionshandlungen wird ‘actxxx‘ vorgeschlagen, wobei für xxx in Kurzform angegeben werden kann, um welche Aktion es sich handelt.

- **Weg zurücklegen:**

  In vielen Fällen bestehen die Handlungen darin, dass Personen Wege von einem zum anderen Ort zurückgelegen müssen. Diese Wege nehmen in der Regel eine gewisse Zeit in Anspruch. Dadurch können die Wegezeiten einen erheblichen Einfluss auf die Dynamik der Handlungsabläufe haben.

  Als allgemeine Bezeichnungsstruktur solcher Handlungen, bei denen von Personen Wege zurückgelegt werden müssen, wird ‘wegxxx‘ vorgeschlagen, wobei für xxx in Kurzform angegeben werden kann, wohin die jeweilige Person geht oder um welchen Weg es sich handelt.

- **Verzögerung:**

  Die Durchführung einer bestimmten Handlung kann es aus verschiedenen Gründen mit einer Verzögerung beginnen. Diese Verzögerung wird als eigenständige Basishandlung betrachtet, da durch die Verzögerungszeit die Dynamik des Handlungsablaufs beeinflusst werden kann.

  Als allgemeine Bezeichnungsstruktur solcher Verzögerungszeiten (delay), die einer Handlungsausführung vorausgehen, wird ‘dlyxxx‘ vorgeschlagen, wobei für xxx in Kurzform angegeben werden kann, welche Handlung von der Verzögerung betroffen ist.

5.4.4 Dialog zur Erstellung von Handlungssequenzen

Basishandlungen werden benötigt, um bestimmte Handlungsabläufe zu beschreiben. Die Handlungsabläufe werden durch Teilabläufe, den so genannten Handlungssequenzen, beschrieben. Die Handlungssequenzen werden ausgeführt, wenn bestimmte ihnen zugeordnete Bedingungen erfüllt sind. Die Spezifikation der Handlungssequenzen erfolgt im Wesentlichen durch eine sequentielle Abfolge von elementaren Einzelhandlungen (Basishandlungen).

Jede einzelne Handlungssequenz, die bei Vorliegen einer bestimmten Bedingung ausgeführt wird, beschreibt einen Teil des Handlungsablaufs. Die jeweilige Bedingung wird durch einen bestimmten Zustand beschrieben (z. B. Füllhöhestand des Druckhalters > 9.5 m, wodurch eine bestimmte Alarm- bzw. Anzeigesituation vorliegt). Durch die Zuordnung einer speziellen Bedingung zu jeder Handlungssequenz ist es möglich, Handlungsabläufe z. B. in Abhängigkeit von

- Systemzustand,
- Prozesszustand,
- Zustand des bisher erfolgten Handlungsablaufs,
- kognitiven und ergonomischen Faktoren und
- stochastischen Einflussfaktoren

zu berücksichtigen. Die Flexibilität des Konzepts erlaubt es dem Benutzer, die Handlungsabläufe in Wechselwirkung mit Prozesszuständen und stochastischen Einflussgrößen in einem beliebigen Detaillierungsgrad darzustellen. Die den Handlungssequenzen zugeordneten Bedingungen dienen als Auslöser der jeweiligen Handlungssequenzen.

Im Folgenden wird der Dialog zur Eingabe von Handlungssequenzen und der ihnen zugeordneten Bedingungen beschrieben. Die Beschreibung wird an einem Beispiel demonstriert. Die betrachteten Handlungssequenzen sind alternative Abläufe nach der Basishandlung, bei der der Reaktorfahrer (RF) einen Druckhalter-Füllstand (DH-FH) > 9,5 m von der Anzeige abliest. Im Folgenden soll er den Schichtleiter (SL) informieren den Schichtleiter. Hierbei wird ein möglicher menschlicher Fehler angenommen, der darin besteht, dass der RF vergisst, den SL über den DH-FH > 9,5 m zu informieren. Folgende alternative Abläufe sind möglich:

- Alternative 1:
  Der RF begeht keinen Fehler und informiert den SL unmittelbar, nachdem er von der Anzeige den DH-FH > 9,5 m abgelesen hat. Es wird geschätzt, dass diese Situation mit einer Wahrscheinlichkeit von 0,90 auftritt.

- Alternative 2:
  Der RF vergisst zunächst, den SL zu informieren. Er bemerkt dies aber innerhalb einer Zeitspanne zwischen 20 s und 60 s und informiert den SL mit einer Verzögerungszeit. Für diese Alternative wird eine Wahrscheinlichkeit von 0,095 angenommen.

- Alternative 3:
  Der RF vergisst vollständig, den SL zu informieren. Die Tatsache, dass DH-FH > 9,5 m erreicht hat, wird durch den SL selbst oder durch andere Operateure in der Warte nach einer gewissen längeren Zeit, die zwischen 120 s und 360 s liegt, erkannt. Es wird angenommen, dass diese Situation mit einer Wahrscheinlichkeit von 0,005 eintritt.

Die einzelnen Verzweigungen, die sich durch den menschlichen Fehler in der Basishandlung ‚infoDH-FH' ergeben, werden durch die Verzweigungsvariable 'b_infoDH-FH' charakterisiert. D. h., wenn

- b_infoDH-FH = 1, dann liegt die Situation der Alternative 1 vor,
- b_infoDH-FH = 2, die Situation der Alternative 2 und
- b_infoDH-FH = 3, dann liegt die Situation der Alternative 3 vor.

Die Zuordnung der alternativen Werte für die Verzweigungsvariable 'b_infoDH-FH' und der Wahrscheinlichkeiten, mit denen die Alternativen eintreten, erfolgt im Probabilistik-Modul von MCDET gemäß den in Abschnitt 3.2 beschriebenen Eingaben.

Im Folgenden werden die Handlungssequenzen beschrieben, die bzgl. des Beispiels über die Benutzeroberfläche zu spezifizieren sind.

- **Handlungssequenz1:** Diese Handlungssequenz wird aktiviert, wenn der Variablen 'ALId' des Key-Vektors über die Eingabeinformationen der relevanten Alarme und Anzeigen (siehe Abschnitt 5.4.2) die Identifikationsnummer der entsprechenden Anzeige (DH-FH > 9,5 m) zugewiesen wird. Der RF, der die entsprechende Anzeige kontrolliert, liest einen DH-FH > 9,5 m ab (RF führt Basishandlung liestDH-FH

aus). Sobald die Anzeige einen DH-FH > 9.5 m anzeigt, soll der RF den SL über den DH-FH informieren. Da bzgl. dieser Basishandlung ein menschlicher Fehler auftreten kann (d. h. RF vergisst, den SL über den DH-FH zu informieren), sind an dieser Stelle bzgl. des weiteren Handlungsablaufs die oben genannten Alternativen zu berücksichtigen. Dies wird in Handlungssequenz 1 durch die Verzweigungsvariable ‘b_infoDH-FH‘ gekennzeichnet. Da der weitere Ablauf vom Wert der Variablen 'b_infoDH-FH‘ abhängt, muss die Handlungssequenz 1 an dieser Stelle unterbrochen werden. Die erste Handlungssequenz besteht also aus der Basishandlung, dass der RF einen DH-FH > 9.5 m von der Anzeige abliest und der Verzweigungsvariablen ‘b_infoDH-FH‘.

- **Handlungssequenz 2:** Diese Handlungssequenz wird unter folgender Bedingung aktiviert: Handlungssequenz 1 muss durchgeführt sein und der RF begeht keinen Fehler (siehe Alternative 1), d. h. die Verzweigungsvariable ‘b_infoDH-FH‘ hat den Wert 1 (b_infoDH-FH = 1). In diesem Fall informiert er sofort den SL über den Füllhöhestand des Druckhalters (RF infoDH-FH). Nachdem der RF den SL informiert hat, überzeugt sich der SL von der Richtigkeit, indem er den DH-FH selbst von der Anzeige abliest (SL liestDH-FH). Um sich zur Anzeige zu begeben und den DH-FH abzulesen, benötigt er nach Abschätzung zwischen 7 s und 15 s. Daraufhin nimmt der SL das Notfallhandbuch (NHB) und liest die entsprechende Stelle über die durchzuführenden Maßnahmen (SL liestNHB). Es wird geschätzt, dass er dafür zwischen 25 s und 50 s benötigt. Als nächstes gibt der SL dem RF die Anweisung, die Hauptkühlmittelpumpe 1 (HKP1) abzustellen (SL AnwHKP1aus). Für die Anweisung benötigt der SL zwischen 5 s und 9 s. Der RF begibt sich zum entsprechenden Schaltpult und drückt den Knopf zum Abstellen der HKP1 (RF actHKP1aus), wofür er zwischen 8 s und 16 s benötigt. Bzgl. dieser Handlungen werden keine weiteren menschlichen Fehler angenommen.

- **Handlungssequenz 3:** Diese Handlungssequenz wird unter der Bedingung aktiviert, dass Handlungssequenz 1 durchgeführt wurde und der RF einen Fehler begeht, der durch die Alternative 2 beschrieben ist. In diesem Fall hat die Verzweigungsvariable ‘b_infoDH-FH‘ den Wert 2 (b_infoDH-FH = 2). Wenn diese Bedingung vorliegt, vergisst der RF zunächst den SL über den DH-FH zu informieren, bemerkt seine Unterlassung jedoch nach einer Verzögerungszeit. Die Folge davon ist, dass der RF den SL erst nach einer Verzögerungszeit über den DH-FH > 9,5 m informieren kann (‘RF dlyInfoDH-FH‘). Es wird angenommen, dass die zufällige Verzögerungszeit zwischen 20 s und 60 s liegt. Wenn der SL schließlich informiert wird, werden die gleichen Handlungen durchgeführt, die für Handlungssequenz 2

beschrieben wurden. D. h., SL liestDH-FH, SL liestNHB, SL AnwHKP1aus und RF actHKP1aus.

- **Handlungssequenz 4:** Diese Handlungssequenz wird unter der Bedingung aktiviert, dass Handlungssequenz 1 durchgeführt wurde und der RF einen Fehler begeht, der durch die Alternative 3 beschrieben ist. In diesem Fall erhält die Verzweigungsvariable ‘b_infoDH-FH‘ den Wert 3 (b_infoDh-FH = 3). Unter dieser Bedingung vergisst der RF vollständig, den SL über den DH-FH zu informieren. Dazu wird angenommen, dass es nach einer Zufallszeit zwischen 120 s und 360 s entweder dem SL selbst oder einer anderen Person in der Warte auffällt, dass der DH-FH > 9,5 m beträgt (SL dlyDH-FH). Daraufhin nimmt der SL das NHB und liest die entsprechende Stelle über die durchzuführenden Maßnahmen, wofür er zwischen 25 s und 50 s benötigt (SL liestNHB). Als nächstes gibt der SL dem RF die Anweisung, die Hauptkühlmittelpumpe 1 (HKP1) abzustellen (SL AnwHKP1aus). Für die Anweisung benötigt der SL zwischen 5 s und 9 s. Der RF begibt sich zum entsprechenden Schaltpult und drückt den Knopf zum Abstellen der HKP1, wofür er zwischen 8 s und 16 s benötigt (RF actHKP1aus). Bzgl. dieser Handlungen werden keine weiteren menschlichen Fehler angenommen werden.

Um diese vier beschriebenen Handlungssequenzen einzugeben, muss im Menü ‘Crew-Modul‘ (vgl. Abb. 5.14) der Menüpunkt **‘Handlungssequenzen‘** angeklickt werden. Danach erscheint das in Abb. 5.28 dargestellte Dialogfenster zur Eingabe der Handlungssequenzen. Im Folgenden wird davon ausgegangen, dass alle zur Spezifikation der Handlungssequenzen benötigten Basishandlungen bereits eingegeben und gespeichert wurden.

Beim Aufruf des Dialogs werden automatisch alle gespeicherten Handlungen in die Liste der Basishandlungen zusammen mit den ihnen automatisch zugeordneten Identifikationsnummern eingelesen (siehe Abb. 5.31). Über die jeweiligen Identifikationsnummern der Basishandlungen werden die einzelnen Handlungssequenzen zusammengestellt. Zusätzlich zu den Basishandlungen ist die Verzweigungsvariable ‘b_infoDH-FH‘‘ für die alternativen Abläufe aufgeführt.

Für jede Handlungssequenz, die spezifiziert wird, muss der Benutzer eine Identifikationsnummer (IdNr) festlegen, die der Handlungssequenz eindeutig zugeordnet ist. Die Wahl der Identifikationsnummer ist zwar beliebig, sollte aber aus Gründen der

Übersichtlichkeit einer bestimmten Struktur folgen, die der Nutzer allerdings selbst bestimmen kann. Die Handlungssequenz 1 erhält hier die IdNr 1.

Zur Eingabe von Handlungssequenz 1 sind aus der Liste der Basishandlungen (vgl. Abb. 5.31) alle Handlungen der Reihe nach anzuklicken, wie sie in der Beschreibung (Modellierung) der Handlungssequenz 1 vorkommen. D. h., zur Eingabe der Handlungssequenz 1 wird zuerst die Basishandlung mit der Nummer 2001 angeklickt, die dann automatisch in der Liste der Handlungssequenz erscheint. Danach wird die Verzweigungsvariable ‘b_infoDH-FH‘ ausgewählt. Die Liste der Handlungssequenzen hat dann die in Abb. 5.31 dargestellten Einträge.

Die Handlungssequenz muss an dieser Stelle unterbrochen werden, da im Folgenden alternative Handlungssequenzen in Abhängigkeit vom Wert der Verzweigungsvariablen ‘b_infoDH-FH‘ möglich sind. Dieser Wert wird der Verzweigungsvariablen im Ablauf der Simulation gemäß den Informationen im Probabilistik-Modul zugeordnet.

Wenn die Basishandlungen der einzugebenden Handlungssequenz zusammengestellt worden sind, muss der Button **‘Bedingung zur Aktivierung der Handlungsliste‘** betätigt werden, um die Bedingung einzugeben, unter der die Handlungsliste aktiviert und berechnet wird. Nach Betätigung des Buttons erscheint der Dialog zur Eingabe der Bedingung. Dieses Dialogfenster entspricht dem in Abb. 5.22. Die Funktion der Oberfläche ist bereits in Abschnitt 5.4.2 ausführlich beschrieben worden, in der die Bedingungen für Alarm- und Anzeigesituationen eingegeben werden konnten.

Eine obligatorische Variable, die bei jeder Spezifikation einer Bedingung zur eindeutigen Zuordnung einer Handlungssequenz angegeben werden muss, ist die Variable ‘HLcont‘, die im Key-Vektor automatisch angelegt wird (vgl. Abschnitt 5.4.1). Die Variable ‘HLcont‘ gibt an, an welche der zuvor berechneten Handlungssequenzen (gekennzeichnet durch die IdNr) die aktuell eingegebene Sequenz angehängt werden soll. Da die hier einzugebende Handlungssequenz mit der IdNr 1 den Beginn des Handlungsablaufs beschreibt, ist für die Variable HLcont = 0 einzusetzen, da die Sequenz auf keiner vorangegangenen Handlungssequenz aufsetzt.

Eine weitere Bedingung zur Aktivierung der Handlungssequenz mit der IdNr 1 ist, dass die Alarm- bzw. Anzeigesituation 1 vorliegt, die durch die Variable ALId = 1 beschrieben wird. Der Parameter ‘ALId‘ wird ebenfalls als Voreinstellung automatisch im Key-Vektor angelegt (vgl. Abschnitt 5.4.1) und erhält seine Werte durch die Spezifikationen

der Alarm- und Anzeigesituationen, die unter dem entsprechenden Menüpunkt definiert werden können. Wenn z. B. der DH-FH > 9,5 m ist, wird dem Parameter ‘ALId‘ der Wert 1 zugewiesen. Die Bedingung, die über die Oberfläche zur Aktivierung der Handlungssequenz mit der IdNr 1 einzugeben ist, lautet somit: ALId = 1 AND HLcont = 0

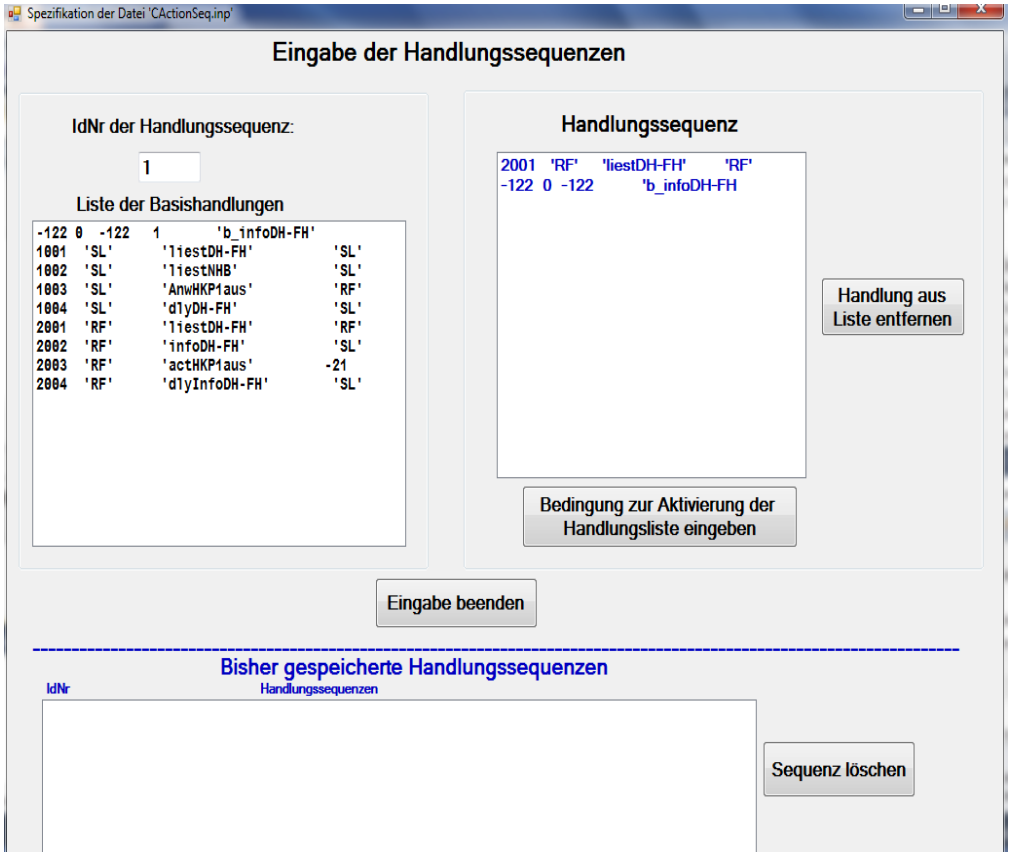


Abb. 5.31 Dialog zur Spezifikation von Handlungssequenzen

Durch Anklicken des Buttons **‘Bedingung zur Aktivierung der Handlungsliste‘** im Dialog in Abb. 5.31 erscheint das Dialogfenster in Abb. 5.22, in der die Bedingung eingegeben werden kann. Wird die Eingabe mit einem END abgeschlossen, so erscheint automatisch das in Abb. 5.32 dargestellte Fenster, mit dem die Angaben zu einer Handlungssequenz entweder einer bestehenden Datei hinzugefügt oder in einer neu anzulegenden Datei gespeichert werden können.

Die Bedingung wird im oberen Teil des Fensters angegeben und kann vor der Speicherung nochmals überprüft werden. Falls die Spezifikation der Bedingung fehlerhaft ist, kann sie durch die Betätigung des Buttons ‘**Neu eingeben !‘** erneut spezifiziert werden.

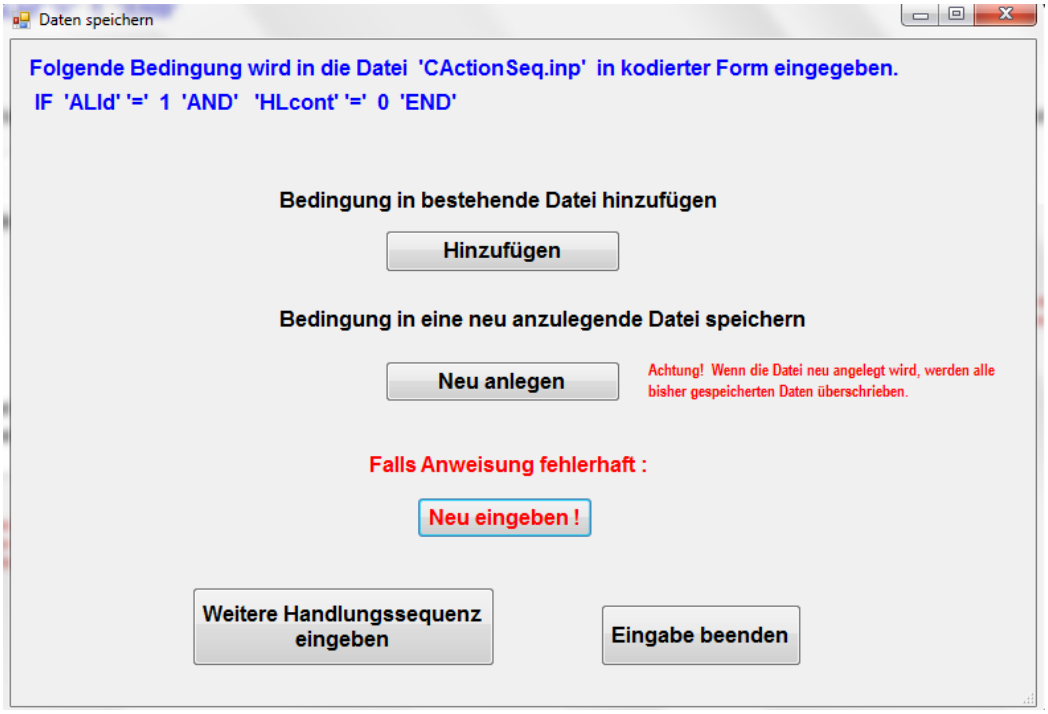


Abb. 5.32 Dialog zum Speichern der Handlungssequenz mit zugehöriger Bedingung

Wird nach Abspeicherung der Bedingung durch **‘Hinzufügen‘** oder **‘Neu anlegen‘** im Dialog in Abb. 5.32 der Button **‘Weitere Handlungssequenz eingeben‘** angeklickt wird, erscheint erneut das in Abb. 5.31 dargestellte Dialogfenster, wobei die zuvor eingegebene Information der Handlungssequenz mit der IdNr 1 in der Liste der bisher gespeicherten Handlungssequenzen angezeigt wird. Die Liste der Handlungssequenz ist leer und wartet auf die Auswahl der Basishandlungen zur Eingabe einer neuen Handlungssequenz.

Handlungssequenz 2 wird die IdNr 2 zugeordnet. Für diese Handlungssequenz werden gemäß obiger Beschreibung die Basishandlungen mit folgenden Nummern aus der Liste der Basishandlungen (vgl. Abb. 5.31) der Reihe nach ausgewählt: 2002, 1001, 1002, 1003 und 2003. Die ausgewählten Basishandlungen beschreiben dann die auszuführenden Handlungen der Handlungsliste 2. Um die Handlungsliste 2 in der Simulation durch das Crew-Modul zu aktivieren, muss eine eindeutige Bedingung zur Aktivierung spezifiziert werden. Zur Aktivierung der Handlungssequenz mit der IdNr 2 muss vorher die Handlungssequenz mit der IdNr 1 durch das Crew-Modul berechnet worden sein. Dies wird durch HLcont = 1 gekennzeichnet. Des Weiteren muss nach der obigen Beschreibung der Handlungssequenz 2 die Verzweigungsvariable ‘b_infoDh-FH‘ den Wert 1 haben. Nach Betätigung des Buttons **‘Bedingung zur Aktivierung der Handlungsliste‘** erscheint die entsprechende Oberfläche, über die die Bedingung HLcont

= 1 AND b_infoDH-FH = 1 einzugeben ist. Durch die Sequenz der Basishandlungen und die zugehörige Bedingung ist die zweite einzugebende Handlungssequenz spezifiziert. Wenn die Spezifikation der Bedingung mit einem END abgeschlossen wird, erscheint der Dialog in Abb. 5.32, in der durch Betätigung des Buttons **‘Hinzufügen‘** die Informationen der Handlungssequenz IdNr 2 gespeichert werden.

Für Handlungssequenz 3 wird die IdNr 3 vergeben. Aus der Liste der Basishandlungen werden gemäß der obigen Beschreibung für die Handlungssequenz der Reihe nach die Handlungen mit den Nummern 2004, 2002, 1001, 1002, 1003 und 2003 ausgewählt. Die für die Handlungssquenz 3 einzugebende Bedingung lautet: HLcont = 1 AND b_infoDH-FH = 2. Wenn die Spezifikation der Bedingung mit einem END abgeschlossen wird, erscheint wieder das Dialogfenster der Abb. 5.30, über das die eingegebenen Informationen gespeichert werden können.

Handlungssequenz 4 wird die IdNr 4 zugeordnet. Nach obiger Beschreibung werden für diese Handlungssequenz die Basishandlungen 1004, 1002, 1003 und 2003 der Reihe nach aus der Liste der Basishandlungen ausgewählt. Da die Handlungssequenz auf der Sequenz mit der IdNr 1 aufsetzt und die Verzweigungsvariable ‘b_infoDH-FH‘ den Wert 3 haben muss, lautet die einzugebende Bedingung: HLcont = 1 AND b_infoDH-FH = 3. Wenn die Spezifikation der Bedingung mit einem END abgeschlossen wird, erscheint wieder das Dialogfenster der Abb. 5.30, über das die eingegebenen Informationen gespeichert werden können.

Eine Handlungssequenz ist grundsätzlich bei einer Verzweigungsvariablen zu unterbrechen. Es können aber auch Basishandlungen auftreten, bei denen Informationen von Prozessgrößen abgefragt werden und es in Abhängigkeit der Prozessgrößen zu unterschiedlichen Handlungsabläufen kommen kann. Wenn solch eine Basishandlung in einer Handlungssequenz auftritt, muss die Basishandlung eine bestimmte Codierung < 0 aufweisen und die Handlungssequenz muss bei dieser Basishandlung ebenfalls unterbrochen werden.

Wie die jeweiligen Handlungssequenzen aufgebaut werden müssen, liegt vor allem am Kontext der Basishandlungen, die den Handlungsablauf beschreiben, und an den stochastischen Einflussgrößen sowie den Wechselwirkungen zwischen Handlungsablauf und der Prozessdynamik.

5.4.5 Dialog zur Spezifikation von Zustandsänderungen durch Basishandlungen

Handlungen, die eine Änderung eines Komponentenzustandes zur Folge haben (z. B. RF stellt HKP1 aus) sowie Handlungen, bei denen Informationen bestimmter Prozessgrößen benötigt werden, von denen der weitere Handlungsablauf abhängt, müssen durch eine bestimmte Codierung gekennzeichnet werden, die aus einer negativen Zahl besteht. Ebenso werden Verzweigungsvariablen mit einer negativen Zahl codiert. Die Codierung von Verzweigungsvariablen erfolgt automatisch und erhält Werte < -100. Der Verzweigungsvariablen ‘b_infoDH-FH‘ wurde z. B. automatisch die Codierung -122 zugeordnet.

Wenn Handlungen eine Zustandsänderung einer Komponente zur Folge haben und der Zustand der Komponente durch einen Parameter im Key-Vektor beschrieben wird, wobei der Parameter mit einem voranstehenden 's‘ gekennzeichnet sein muss (z. B. ‘sHKP1‘), wird die Codierung automatisch über den Dialog zur Eingabe von Basishandlungen vorgenommen. Dies wurde bereits in Abschnitt 5.4.3 beschrieben. Codierungsnummern von Handlungen, bei denen Informationen von Prozessgrößen benötigt werden, sind frei wählbar. Bei der Vergabe von Codierungen ist vor allem darauf zu achten, dass die Codierungen einer Handlung eindeutig vergeben werden.

Durch die Codierungen von Basishandlungen wird angegeben, welche Basishandlung eine Änderung des Komponentenzustands oder ein sonstiges relevantes Ereignis zur Folge hat. Unter dem Menüpunkt **‘Zustandsänderungen durch Handlungen‘** des Menüs ‘Crew-Modul‘ kann definiert werden, welche Zustandsänderungen bei Parametern des Key-Vektors durch die verschiedenen Codierungen durchgeführt werden.

Dies soll an dem in Abschnitt 5.4.4 verwendeten Beispiel demonstriert werden. Die Basishandlungen in dem Beispiel sind in Abb. 5.31 in der entsprechenden Liste aufgeführt. Aus dieser Liste ist erkenntlich, dass die Verzweigungsvariable ‘b_infoDh-FH‘ die Codierung -122 und die Basishandlung mit der Identifikationsnummer 2003 die Codierung - 21 hat. Wenn in einer Handlungsliste die Verzweigungsvariable ‘b_infoDH-FH‘ auftritt, muss der Zustand der Variablen von 0 auf 1 gesetzt werden (als Ausgangszustand hat eine Verzweigungsvariable den Wert 0). Dazu wird im Menü der Abb. 5.14 der Menüpunkt **‘Zustandsänderung durch Handlungen‘** ausgewählt, und es erscheint das in Abb. 5.33 dargestellt Fenster.

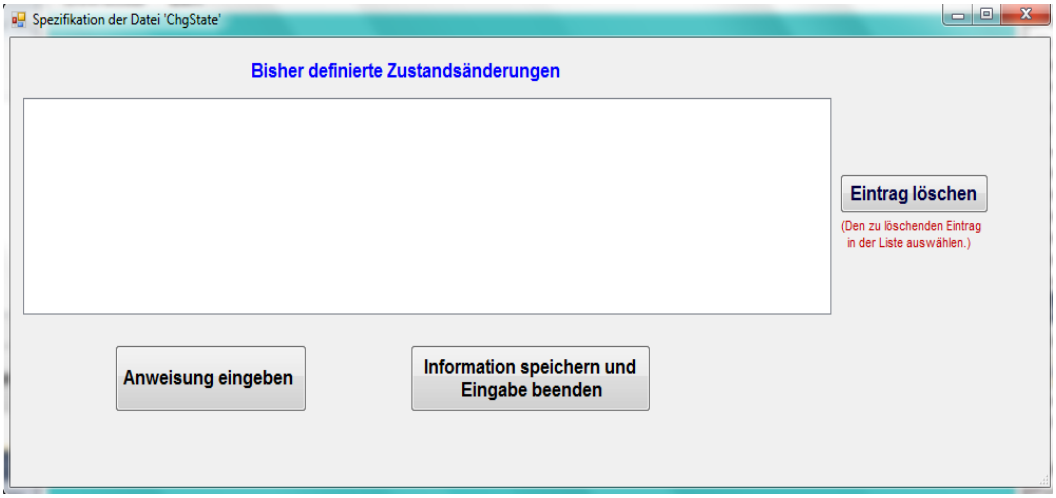


Abb. 5.33 Dialog zur Eingabe von Zustandsänderungen durch Basishandlungen

Da bisher noch keine Anweisungen eingegeben wurden, ist die Liste der bisher definierten Zustandsänderungen leer. Um das Programm anzuweisen, das die Verzweigungsvariable ‘b_infoDH-FH‘ vom Zustand 0 auf den Zustand 1 gesetzt werden soll, wenn die Verzweigungsvariable in einer Handlungsliste auftritt, so ist der Button **‘Anweisung eingeben‘** zu drücken. Danach erscheint das in Abb. 5.34 dargestellte Fenster.

Die Codierungswerte von Basishandlungen werden der Variablen ‘chgCode‘ zugewiesen, die automatisch im Key-Vektor angelegt wird (vgl. Abb. 5.17). Deshalb ist die Variable ‘chgCode‘ in der Parameterliste schon als Voreinstellung ausgewählt, um festzulegen, welche Codierung spezifiziert werden soll. Alle Basishandlungen, die eine Codierung < 0 aufweisen, werden in der Liste **‘Codierung der Basishandlung‘** aufgeführt, um die Spezifikation einfacher und übersichtlicher zu gestalten.

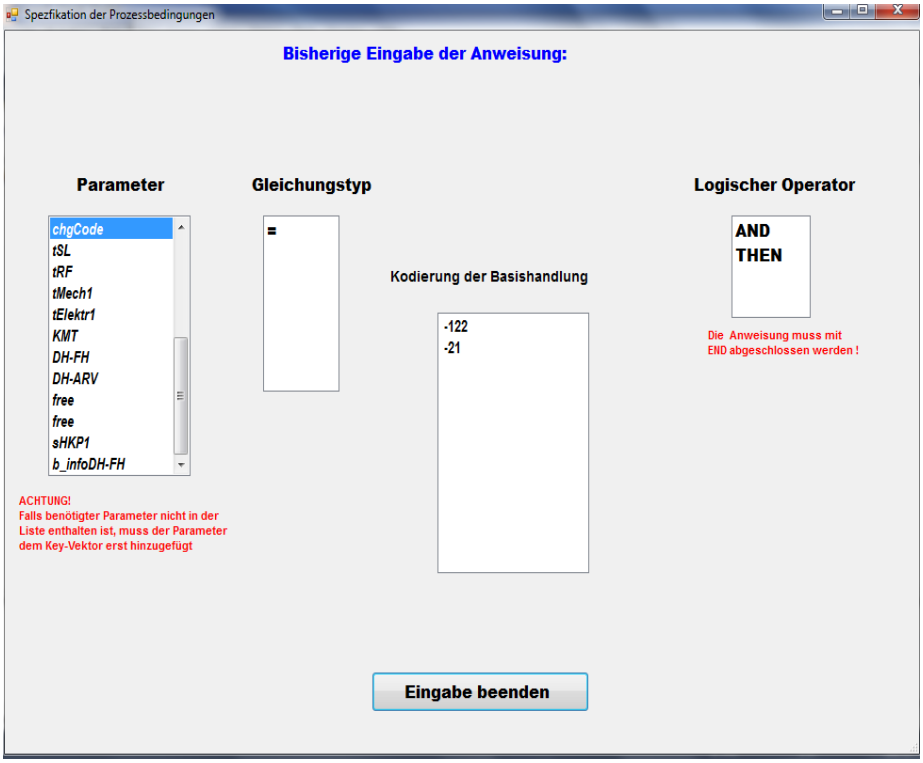


Abb. 5.34 Dialog zur Spezifikation von Zustandsänderungen durch Basishandlungen

Da die Verzweigungsvariable ‘b_infoDH-FH‘ die Codierung -122 hat und vom Wert 0 auf den Wert 1 gesetzt werden soll, muss über die Oberfläche folgende Anweisung eingegeben werden:

chgCode = -122 AND b_infoDH-FH = 0 THEN b_infoDH-FH = 1

Dies erfolgt durch einfaches Anklicken der jeweiligen Parameter, Operatoren und Werte in dem in Abb. 5.34 dargestellten Fenster. Wenn der Benutzer seine Anweisung eingegeben hat und mit END abschließt, so erscheint das in Abb. 5.35 dargestellte Fenster, mit dem die eingegebene Anweisung, die im oberen Teil des Fensters dargestellt ist, überprüft werden kann. Ist die Anweisung fehlerhaft kann sie durch Betätigung des Buttons **Neu eingeben** erneut spezifiziert werden.

Ist die Anweisung korrekt, muss der Button **‘Übernehmen‘** gedrückt werden, wodurch das Dialogfenster in Abb. 5.33 erneut erscheint und die gerade eingegebene Anweisung in der entsprechenden Liste der bisher definierten Zustandsänderungen angezeigt wird.

Abb. 5.35 Dialog zum Speichern der Anweisung

Die Basishandlung ‘RF actHKP1aus‘, in der der RF die HKP1 abstellt, hat die Codierung -21. Durch die Codierung -21 wird dem Programm mitgeteilt, dass nach Durchführung dieser Basishandlung der Zustand der HKP1 geändert werden muss. Der Zustand der HKP1 wird im Key-Vektor durch die Variable ‘sHKP1‘ beschrieben. Die Zustände sind beschrieben durch

- sHKP1 = 1 - Pumpe läuft.
- sHKP1 = 0 - Pumpe ist abgestellt.

Wenn in der Simulation des Handlungsablaufs eine Handlungssequenz berechnet wird, in der die Basishandlung ‘RF actHKP1aus‘ mit der Codierung -21 auftritt, wird zu dem betreffenden Zeitpunkt, zu dem die Handlung durchgeführt wird, die Codierung -21 automatisch der Variablen ‘chgCode‘ des Key-Vektors übertragen. Der Codierung -21 muss nun die Funktion zugeordnet werden, dass der Zustand der Hauptkühlmittelpumpe 1 vom Zustand 1 (d. h. Pumpe läuft) in den Zustand 0 (d. h. Pumpe aus) geändert werden muss. D. h., durch die Codierung -21 wird das Programm veranlasst, den Zustand der HKP1 von 1 auf 0 zu setzen. Die Funktion der Codierung muss über eine entsprechende Anweisung definiert werden.

Wird in dem Fenster der Abb. 5.33 der Button **‘Anweisung eingeben‘** gedrückt, so erscheint erneut das Dialogfenster in Abb. 5.34, in der die Anweisung zum Abstellen

der HKP1 durch Anklicken der entsprechenden Parameter und Operatoren eingegeben werden kann. Die einzugebende Anweisung lautet in diesem Fall:

chgCode = - 21 AND sHKP1 = 1 THEN sHKP1 = 0

Wenn die Anweisung mit einem END angeschlossen wird, so erscheint das in Abb. 5.34 dargestellte Fenster, wobei die neu eingegebene Anweisung im oberen Teil des Fensters aufgeführt ist und auf Korrektheit überprüft werden kann. Ist die Anweisung richtig, so erscheint durch Anklicken des Buttons '**Übernehmen'** erneut das Fenster in Abb. 5.33, wobei die bisher durchgeführten Anweisungen in dem entsprechenden Feld der bisher definierten Zustandsänderungen aufgeführt sind. Mit dem Button **'Information speichern und Eingabe beenden'** werden alle spezifizierten Anweisungen gespeichert.

5.5 Menü 'Simulator Runs'

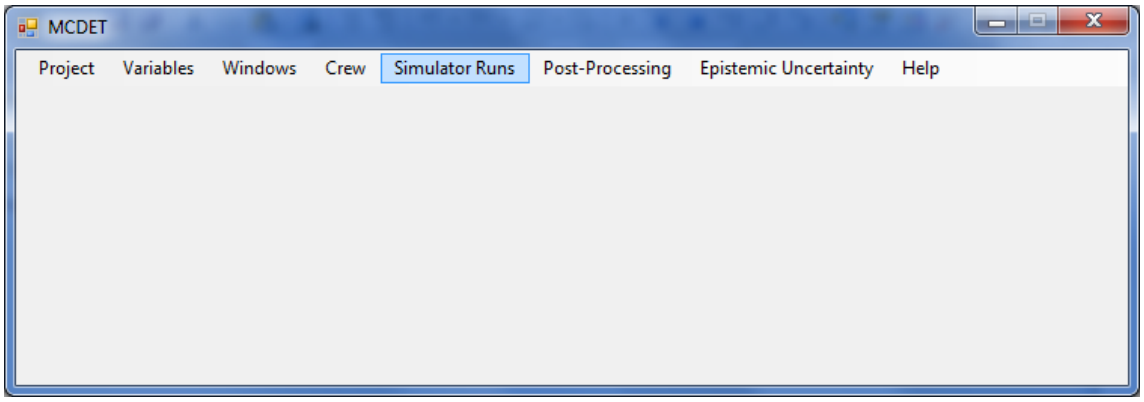


Abb. 5.36 Menü 'Simulator Runs' der MCDET-Menüleiste

Zum Menü 'Simulator Runs'(siehe Abb. 5.36) sind bisher noch keine Dialoge entworfen worden. Zu den Informationen, die hier abgefragt werden müssen, gehören

- die Anzahl der Monte-Carlo-Simulationen, die zur Berücksichtigung der epistemischen Unsicherheiten durchgeführt werden sollen (z. B. 60),
- die Anzahl der Monte-Carlo-Simulationen bzw. die Anzahl der dynamischen Ereignisbäume, die zur Berücksichtigung der stetigen aleatorischen Unsicherheiten durchgeführt bzw. berechnet werden sollen (z. B. 50),

- die Anzahl der dynamischen Ereignisbäume, die beispielhaft ausgedruckt werden sollen (z. B. 1),
- der Anfangswert des Zufallszahlengenerators (SEED) zum Ausspielen von Werten für die Monte-Carlo-Simulation (z. B. 123457),
- der Wert für die Wahrscheinlichkeit einer Simulationssequenz, der zum Abbruch führt (z. B. 1.0 E-04),
- das anzuwendende Rechenprogramm (z. B. ATHLET),
- die Anzahl der zu verwendenden Rechenknoten in einem Rechencluster (z. B. 40).

5.6 Menü ‘Post-Processing’

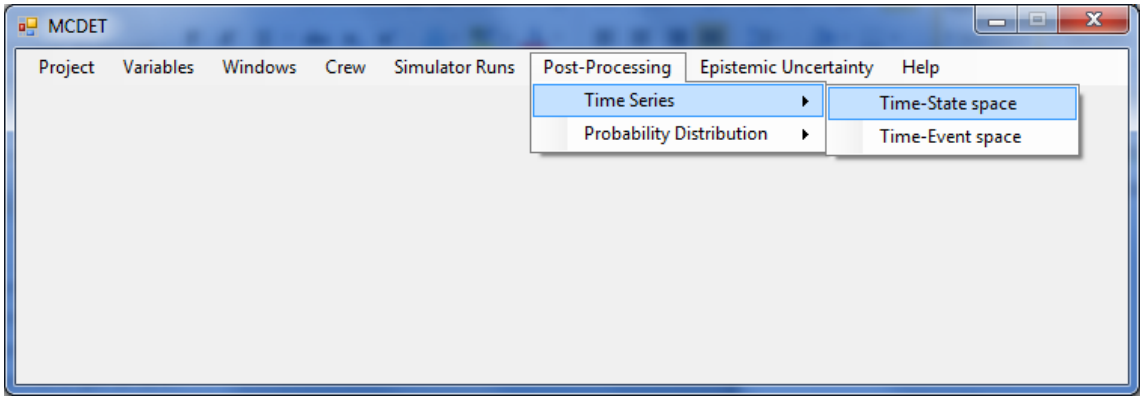


Abb. 5.37 Menü ‘Post-Processing‘ der MCDET-Menüleiste mit Fokus auf das Untermenü ‘Time Series‘

Das Menü ‘Post-Processing‘ der MCDET-Menüleiste (vgl. Abb. 5.37) beinhaltet die beiden Untermenüs ‘Time Series‘ und ‘Probability Distribution‘. Im Untermenü ‘Time Series‘ erfolgt die Auswertung der zeitlichen Abfolge sowohl von System- und Prozesszuständen als auch der zufälligen Ereignisse, die in einer Simulationslauf aufgetreten sind.

Die Auswahl des Unterpunkts ‘Time-State space (Zeit-Zustands-Ebene)‘ im Untermenü ‘Time Series‘ liefert Ereignisbaum- bzw. Szenario-spezifische Ergebnisse zur zeitlichen Entwicklung von System- und Prozessgrößen, wie z. B. die Kernaustrittstemperatur oder die UO_2-Schmelzemasse bei einem Unfallszenario (vgl. Abb. 5.38), den Füllstand

in einem Tank-System (vgl. Abb. 5.39) oder die Temperatur innerhalb eines Kabelmantels bei einem Brandszenario (vgl. Abb. 5.40). Der Dialog zum Abfragen der jeweils erforderlichen Informationen ist bisher noch nicht entworfen worden.

Die Auswahl des Unterpunkts ‘Time-Event space‘ im Untermenü ‘Time Series‘ liefert Ereignisbaum-spezifische Ergebnisse zur zeitlichen Abfolge von Ereignissen. Der Dialog zum Abfragen der erforderlichen Informationen ist bisher noch nicht entworfen worden. Eine mögliche grafische Darstellung, die aus dem Dialog resultiert, ist in Abb. 5.41 zu finden. Allerdings bedarf diese Darstellung einer Überarbeitung, um die Ereignisse in einer Baumstruktur entlang der Zeitachse besser zu verdeutlichen. Ein neuer Ansatz könnte mit Hilfe der Analysewerkzeuge Freemind, Graphviz oder PlantUML verwirklicht werden. Diese Analysewerkzeuge können aus dem Internet kostenlos heruntergeladen werden. Die Erprobung der Möglichkeiten dieser Tools und die Entwicklung einer neuen Darstellung eines dynamischen Ereignisbaums sollen in einem entsprechenden Nachfolgevorhaben erfolgen.

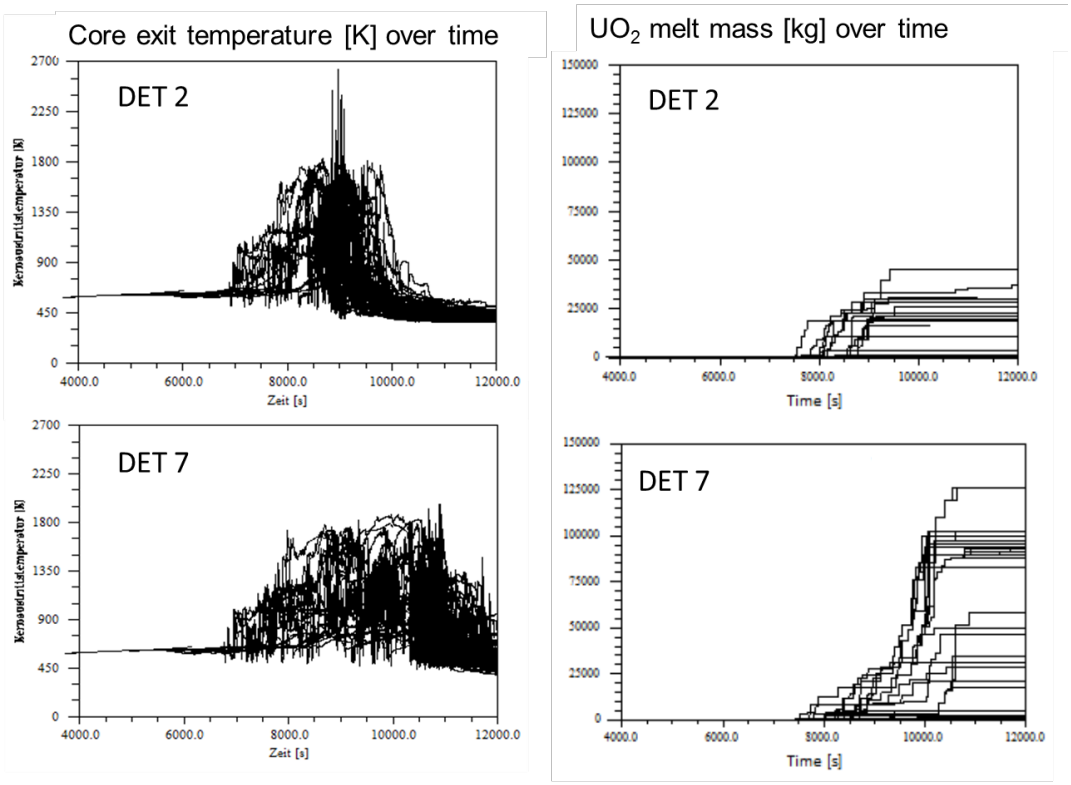


Abb. 5.38 Zeitliche Entwicklung der Kernaustrittstemperatur und der UO_2-Schmelzemasse für zwei ausgewählte dynamische Ereignisbäume aus der MCDET-Analyse eines Station-Blackout-Szenarios

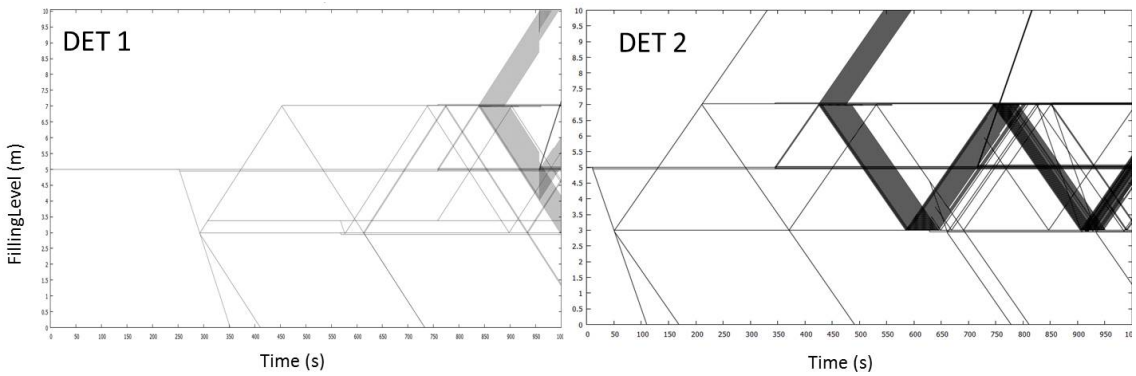


Abb. 5.39 Zeitliche Entwicklung des Füllstands in einem Tank für zwei ausgewählte dynamische Ereignisbäume aus einer MCDET-Analyse des beispielhaften Tank-Systems (vgl. Anhang)

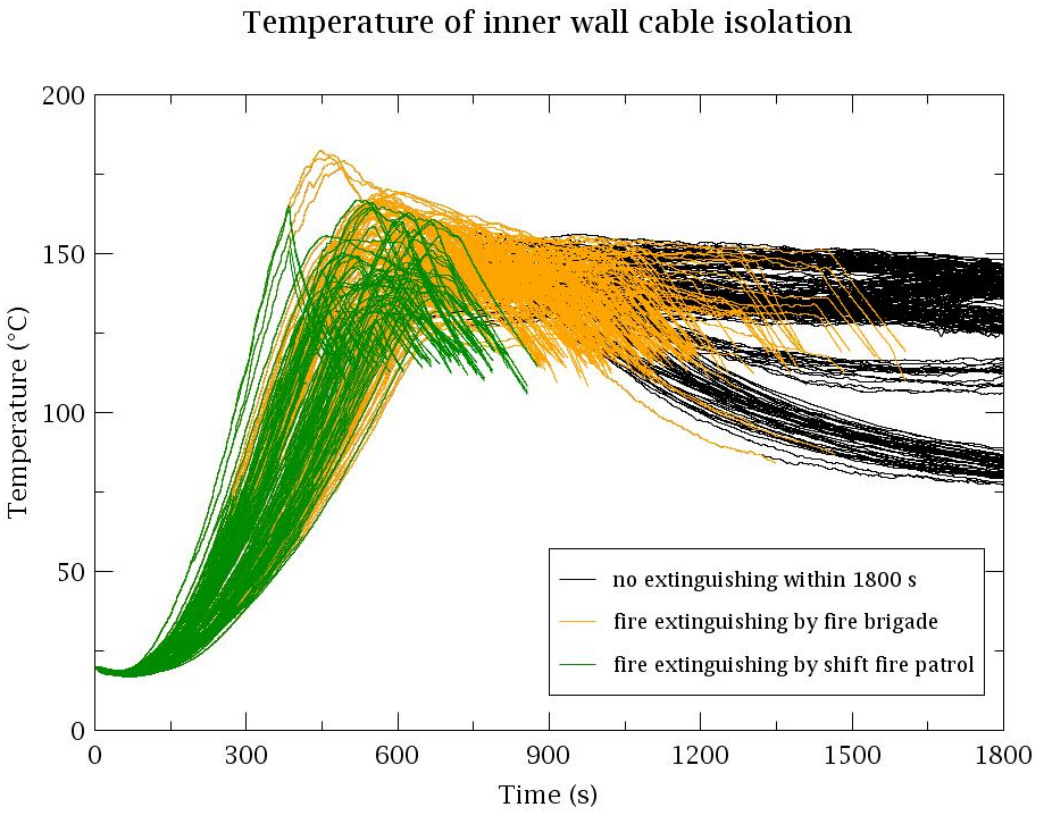


Abb. 5.40 Zeitliche Entwicklung der Temperatur innerhalb eines Kabelmantels für alle 60 dynamischen Ereignisbäume aus einer MCDET-Analyse eines Brandszenarios

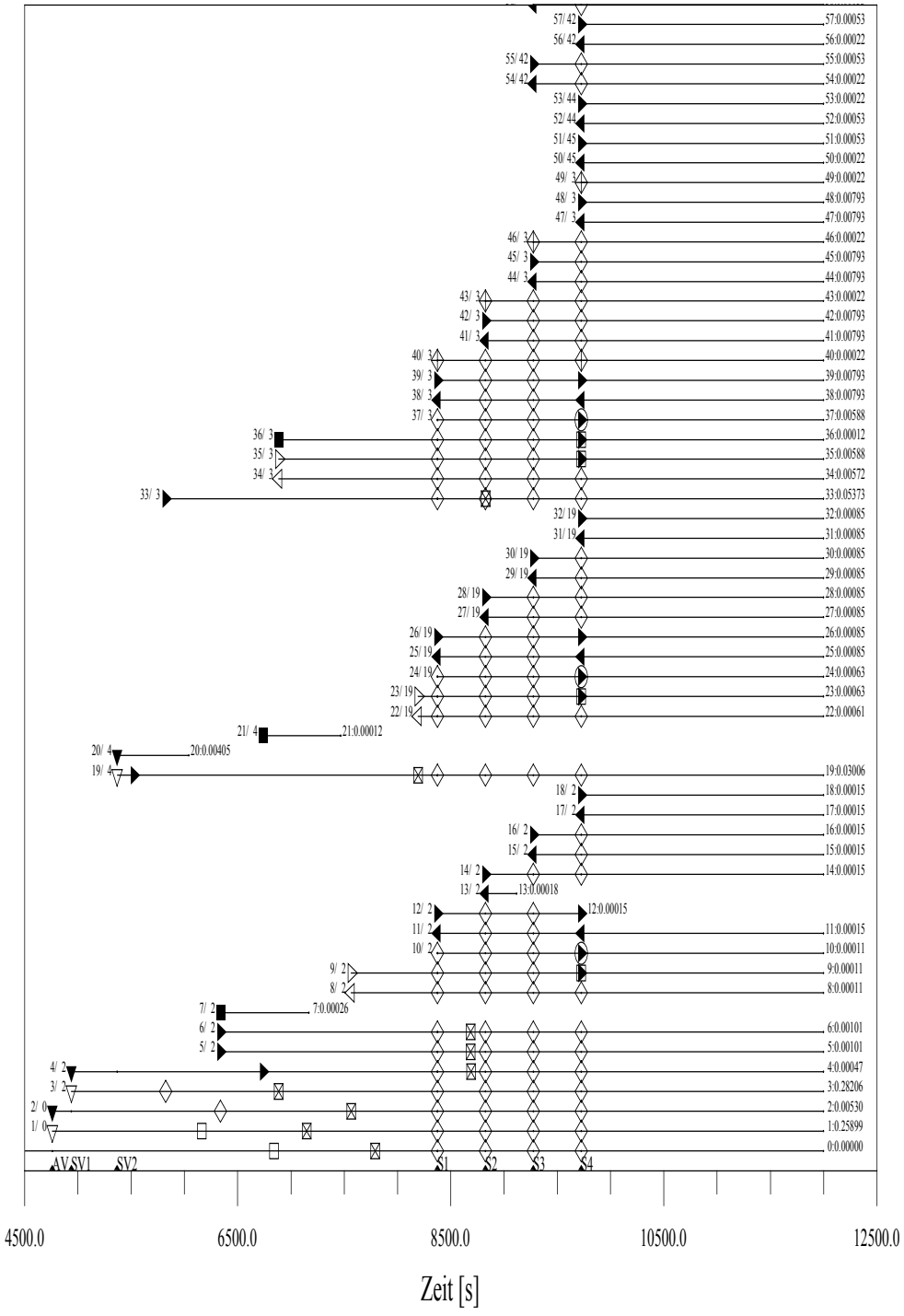


Abb. 5.41 Zeitliche Abfolge von Ereignissen für eine Auswahl von Sequenzen aus einem dynamischen Ereignisbaum (Symbole kennzeichnen berücksichtigte Ereignisse in einem Unfallszenario)

Im Untermenü ‘Probability Distribution‘ des Menüs ‘Post-Processing‘ erfolgt die Auswertung zur Bewertung der Konsequenzen eines Störfall- bzw. Unfallablaufs. Die Auswertung kann Ereignisbaum-spezifisch (DET specific) oder Anwendungs- bzw. Szena-

rio-spezifisch (scenario specific) erfolgen. Beispiele grafischer Darstellungen für Ereignisbaum-spezifische Verteilungen sind in Abb. 5.43 bis einschließlich Abb. 5.45 zu finden. Abb. 5.44 enthält im Vergleich zu den anderen Abbildungen zusätzlich Informationen zum berechneten Wert einer Prozessgröße (z. B. insgesamt erzeugte H_2-Masse) in jeder Sequenz des betrachteten Ereignisbaums. Szenario-spezifische Verteilungen sind in Abb. 5.46 und Abb. 5.47 zu finden. Die Dialoge hierzu sind bisher noch nicht entworfen worden.

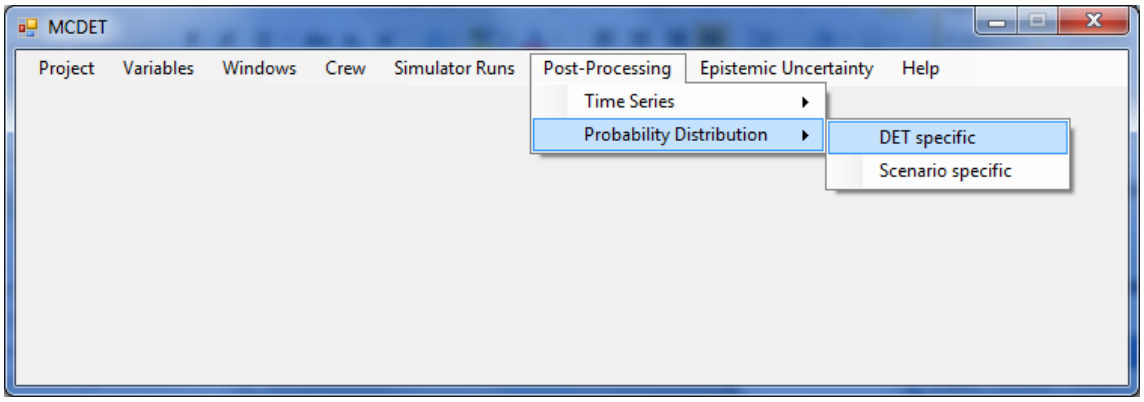


Abb. 5.42 Menü ‘Post-Processing‘ der MCDET-Menüleiste mit Fokus auf das Untermenü ‘Probability Distribution‘

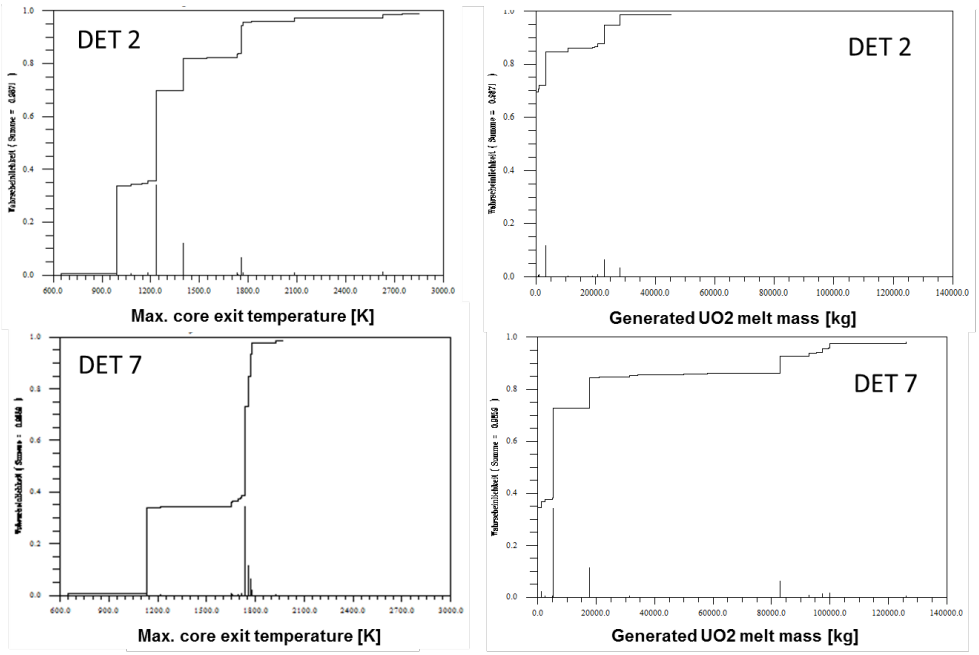


Abb. 5.43 Verteilungen der maximalen Kernaustrittstemperatur und der insgesamt erzeugten UO_2-Schmelzemasse für zwei ausgewählte dynamische Ereignisbäume aus der MCDET-Analyse eines Station-Blackout-Szenarios

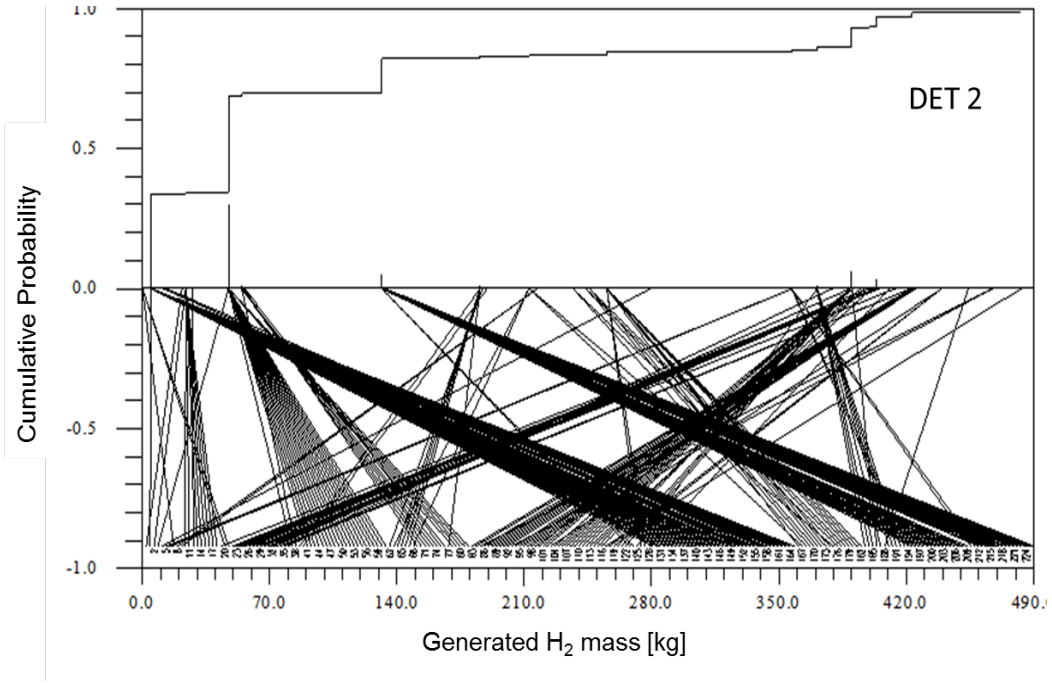


Abb. 5.44 Verteilung der insgesamt erzeugten H_2-Masse inklusive des Beitrags der einzelnen Sequenzen für einen ausgewählten dynamischen Ereignisbaum aus der MCDET-Analyse eines Station-Blackout-Szenarios

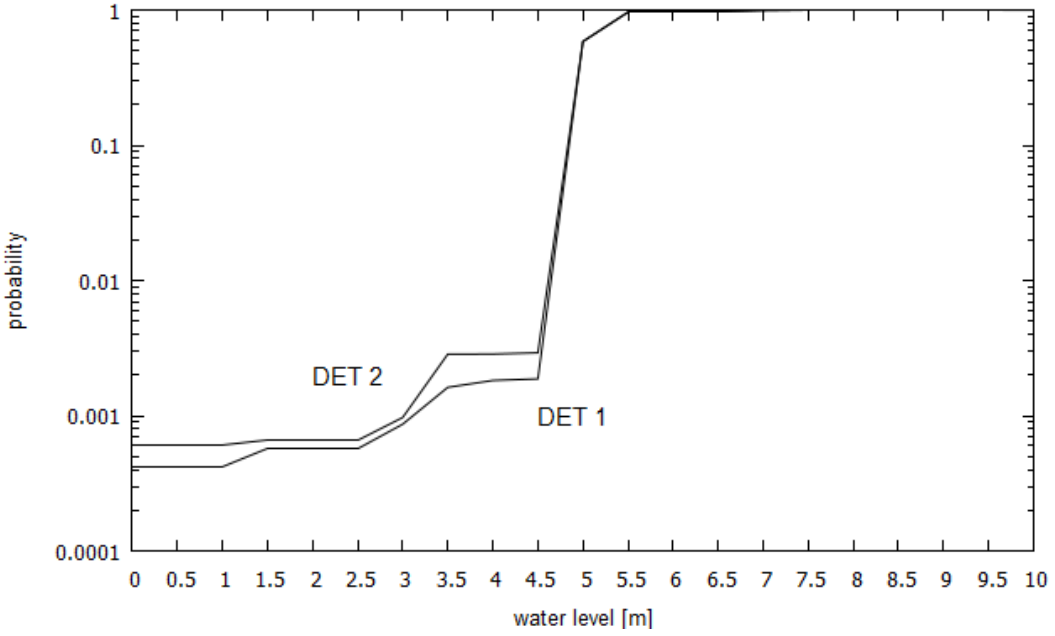


Abb. 5.45 Verteilungen des Füllstands in einem Tank für zwei ausgewählte dynamische Ereignisbäume aus einer MCDET-Analyse des beispielhaften Tank-Systems (vgl. Anhang)

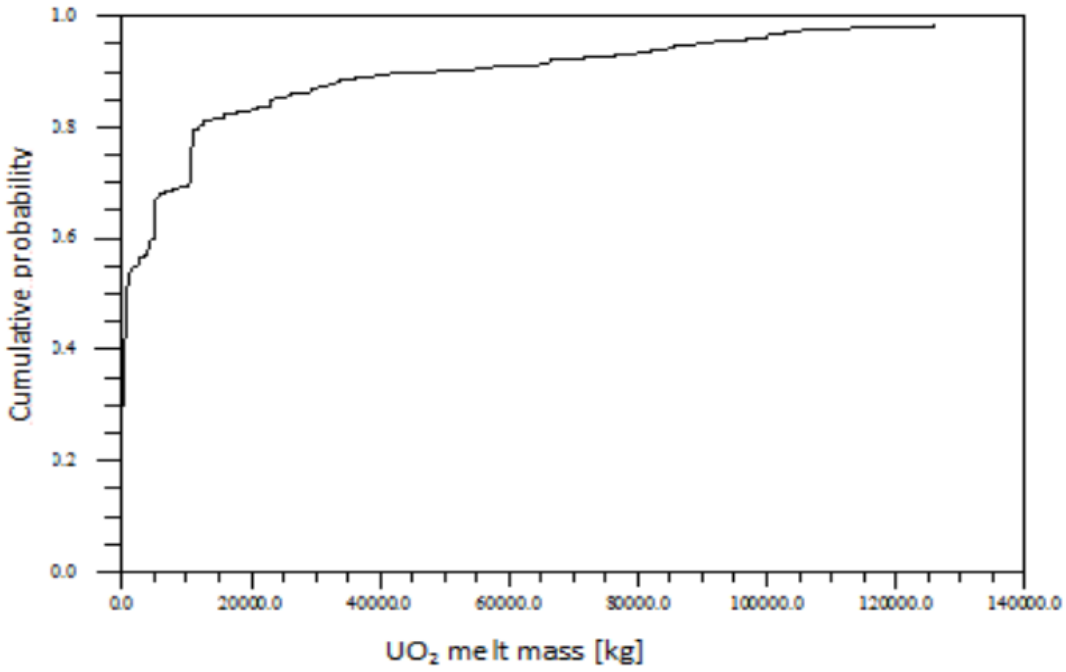


Abb. 5.46 Szenario-spezifische Verteilung der insgesamt erzeugten H_2-Masse über alle 50 dynamischen Ereignisbäume aus der MCDET-Analyse eines Station-Blackout-Szenarios

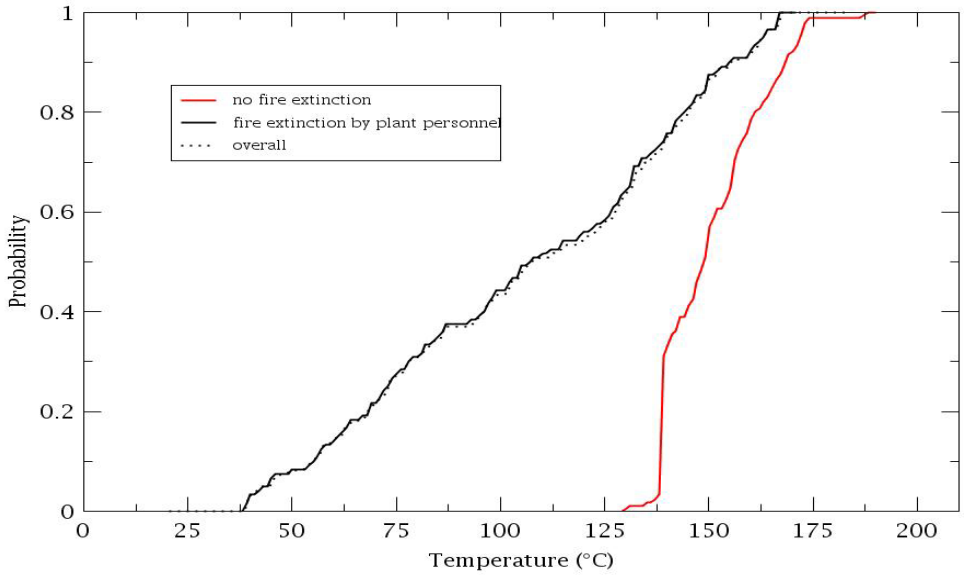


Abb. 5.47 Szenario-spezifische Verteilungen der Temperatur innerhalb eines Kabelmantels über alle 60 dynamischen Ereignisbäume aus der MCDET-Analyse eines Brandszenarios

5.7 Menü ‘Epistemic Uncertainty‘

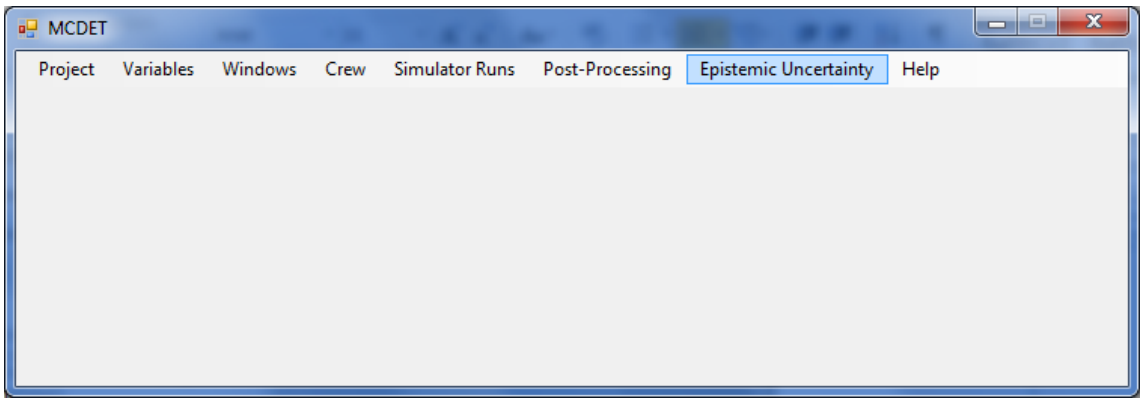


Abb. 5.48 Menü ‘Epistemic Uncertainty‘ der MCDET-Menüleiste

Unter dem Menü ‘Epistemic Uncertainty‘ sind Angaben darüber zu machen, für welche Ergebnisgrößen die epistemische Unsicherheit quantifiziert werden sollen und welcher Quantifizierungsansatz dabei angewendet werden soll. In Frage kommende Ergebnisgrößen sind die Wahrscheinlichkeiten von Prozess- und Systemzuständen, insbesondere von Schadenszuständen.

Mögliche Quantifizierungsansätze sind:

1. ein Unsicherheitsbereich für die Ergebnisgröße in Form eines ein- oder zweiseitigen Toleranzintervalls (z. B. das zweiseitige (95 %, 95 %) Toleranzintervall, das mindestens 95 % der möglichen Werte einer Ergebnisgröße enthält bei einer statistischen Sicherheit von mindestens 95 %, /WIL 41/, /WIL 42/);

2. ein Unsicherheitsbereich für die Ergebnisgröße in Form eines ein- bzw. zweiseitigen Intervalls entsprechend den Ungleichungen nach Chebychev bzw. Cantelli /KOT 82/ (z. B. ein Unsicherheitsbereich der mindestens 95 % der möglichen Werte enthält):

   Chebychev:

$$P(\left|P_{Ep} - E(P_{Ep})\right| \geq t) \leq \frac{Var(P_{Ep})}{t^2}, t \geq 0$$

   Cantelli:

$$P(P_{Ep} - E(P_{Ep}) \geq t) \leq \frac{Var(P_{Ep})}{Var(P_{Ep}) + t^2}, t \geq 0$$

$$P(P_{Ep} - E(P_{Ep}) \leq -t) \leq \frac{Var(P_{Ep})}{Var(P_{Ep}) + t^2}, t \geq 0$$

wobei $E(P_{Ep})$ der Erwartungswert und $Var(P_{Ep})$ die Varianz der Ergebnisgröße P_{Ep} sind;

3. eine passende Verteilung für die Ergebnisgröße und die Angabe z. B. der 5 %-, 50 %- und 95 %-Quantile.

Alle drei Quantifizierungsansätze erfordern die Durchführung von Monte-Carlo-Simulationen auf Basis einer Stichprobe von Wertekombinationen für die epistemischen Variablen. Quantifizierungsansatz 1 erfordert eine Mindestanzahl von Rechenläufen, aus denen die Toleranzgrenzen bestimmt werden. So sind beispielsweise für die Ermittlung der einseitigen oberen (95 %, 95 %) Toleranzgrenze mindestens 59 Wertekombinationen für die epistemischen Variablen zu betrachten, d. h. es müssen mindestens 59 Ergebnisgrößen vorliegen, die ausschließlich aufgrund des gemeinsamen Einflusses der epistemischen Variablen variieren.

Beim Quantifizierungsansatz 2 werden aus einer Stichprobe von Ergebnisgrößen, die ausschließlich aufgrund des gemeinsamen Einflusses der epistemischen Variablen variieren, zunächst der Erwartungswert $E(P_{Ep})$ und die Varianz $Var(P_{Ep})$ der Ergebnisgröße P_{Ep} geschätzt. Für die Stichprobe ist dabei kein Mindestumfang wie beim Ansatz 1 erforderlich. Die Schätzer für Erwartungswert und Varianz werden in den Ungleichungen von Chebychev bzw. Cantelli /KOT 82/ verwendet, um den Unsicherheitsbereich abzuschätzen. Diese Abschätzung ist sehr konservativ.

Beim Quantifizierungsansatz 3 werden genau wie beim Quantifizierungsansatz 2 zunächst der Erwartungswert $E(P_{Ep})$ und die Varianz $Var(P_{Ep})$ der Ergebnisgröße P_{Ep} geschätzt. Dann erfolgt die Annahme eines bestimmten Verteilungstyps für P_{Ep} (z. B. Beta- oder Lognormal-Verteilung). Aus den Schätzern für $E(P_{Ep})$ und $Var(P_{Ep})$ werden schließlich die Verteilungsparameter geschätzt. Damit liegt dann die Verteilung vor, aus der sich schnell die gewünschten Quantile bestimmen lassen.

6 Zusammenfassung

Die im Rahmen des vom Bundesministerium für Wirtschaft und Energie (BMWi) geförderten Forschungs- und Entwicklungsvorhaben RS1198 durchgeführten Weiterentwicklungen zur MCDET-Methode und zum zugehörigen Rechenwerkzeug für probabilistische Dynamikanalysen tragen wesentlich zur Verbesserung der praktischen Durchführbarkeit und Effizienz sowie der Qualitätssicherung von MCDET-Analysen bei.

Der neue Ansatz zur Steuerung von MCDET-Simulationen ermöglicht eine effiziente Kopplung von MCDET mit deterministischen Rechenprogrammen (wie z. B. ATHLET oder MELCOR) sowie einen effizienten Ablauf bei der Berechnung der Sequenzen einer Vielzahl von dynamischen Ereignisbäumen. Mehrere Sequenzen können gleichzeitig von parallel durchgeführten Simulationsprozessen berechnet werden. Diese Prozesse können auf verfügbare Recheneinheiten verteilt werden, um die Rechenkapazität von Mehrkern-Systemen zu nutzen und so die Gesamtrechenzeit drastisch zu reduzieren.

Um MCDET an verschiedene Rechenprogramme anbinden zu können, wurde an der bisherigen modularen Aufteilung der Programmteile von MCDET mit einigen Änderungen festgehalten. Das Scheduler-Modul übernimmt die Steuerung der Simulationsprozesse des Rechenprogramms sowie deren Verwaltung und die Organisation des Datenaustauschs. Die zentrale Steuerung der Simulationsprozesse wird zukünftig nicht mehr über einen dateibasierten Datenaustausch sondern über synchronisierbare Netzwerkverbindungen erfolgen. Dadurch wird sowohl die I/O-Last (Eingabe/Ausgabe-Last) durch dafür notwendige Festplattenzugriffe reduziert als auch die Verwaltung entsprechender Steuerdateien vermieden. Eine weitere Reduzierung der I/O-Last wird dadurch erreicht, dass die bisherige Stapel-basierte Speicherung der Restartpunkte eines Simulationsprozesses nicht mehr erforderlich ist.

Die Eingabedateien des verwendeten Rechenprogramms müssen nicht speziell für die Kopplung mit MCDET angepasst werden. Dadurch erleichtert sich die Pflege der Eingabedateien erheblich, und Unstimmigkeiten zwischen verschiedenen Versionen werden vermieden. Für die Behandlung von Verzweigungen bei der Simulation eines dynamischen Ereignisbaums ist es wichtig, dass das verwendete Rechenprogramm eine Möglichkeit bietet, eine laufende Simulation zu duplizieren.

Für eine effiziente Kommunikation des Probabilistik-Moduls mit dem Scheduler-Modul wurde eine Kommunikationsbibliothek entwickelt. Mittels der hierbei eingesetzten Datenverwaltung (Hashtabellen) können die prozessspezifischen Werte sämtlicher Variablen, die die Grundlage für die Berechnungen von MCDET bilden, bereitgehalten und bei Bedarf verändert werden.

Um den Rechenaufwand von MCDET-Analysen weiter zu reduzieren, wurde ein neuer Algorithmus zur Simulation einer Stichprobe von mehreren dynamischen Ereignisbäumen entwickelt und implementiert. Statt jeden Baum separat zu konstruieren, wird jetzt ein einziger dynamischer Mega-Ereignisbaum erstellt, der sämtliche Ereignisbäume einer Stichprobe umfasst. Dadurch wird ein weiterer Beitrag zu einem effizienten Ablauf der Simulationsläufe geleistet und erhebliche Rechenzeit eingespart, da identische Sequenzen in den einzelnen Ereignisbäumen nur einmal gerechnet werden.

Durch die Berücksichtigung sogenannter absorbierender Zustände, die in der MCDET-Eingabedatei spezifiziert werden können, wird weitere Rechenzeit eingespart, da ein Simulationslauf vorzeitig abgebrochen wird, sobald der berechnete Systemzustand mit einem angegebenen absorbierenden Zustand übereinstimmt. Absorbierende Zustände sind sichere, Gefährdungs- oder Schadenszustände, in denen ein System verbleiben wird, wenn es diese erreicht hat, weil es keine Einflussfaktoren mehr gibt, die eine relevante Zustandsänderung bewirken können.

Viel Rechenzeit kann auch durch eine Online-Klassifizierung von Simulationsläufen eingespart werden. Dabei wird jeder Simulationslauf von Beginn an überwacht und während der Laufzeit daraufhin klassifiziert, ob er weitergerechnet werden sollte oder nicht. Nach Auswertung der Fachliteratur kommen als geeignete Methoden dafür insbesondere ein auf Hidden-Markov-Modellen basierendes Verfahren sowie ein probabilistisches Clusteranalyse-Verfahren in Frage. Beide Verfahren wurden bereits erfolgreich im Zusammenhang mit der Berechnung eines dynamischen Ereignisbaums bzw. mit der Durchführung von Monte-Carlo-Simulationen angewendet.

Aufgrund der Auswertung von Anwendererfahrungen mussten die erforderlichen Eingaben für eine MCDET-Anwendung teilweise überarbeitet werden. Insbesondere wurde die Bedingung für die Aktivierung eines Zeit-Zustandsfensters um die zusätzliche Angabe eines Stimulus erweitert. Dadurch wird sichergestellt, dass das Zeit-Zustandsfester erst dann aktiviert wird, wenn der zugehörige Stimulus eingetreten ist,

d. h. wenn sich eine Variable gegenüber dem letzten berechneten Zeitpunkt eines Simulationslaufs aktuell in den als Stimulus charakterisierten Zustand geändert hat.

Um weitere Analysemöglichkeiten mit MCDET zu schaffen, wurden zusätzliche Funktionalitäten implementiert. Sie ermöglichen die Ermittlung von Zeit-Zustandsvariablen als Funktion von Prozessgrößen, die ausschließliche Durchführung von Monte-Carlo-Simulation ohne die Simulation eines dynamischen Ereignisbaums sowie die Berücksichtigung einer Verzweigung (d. h. alternativer Systemzustände einschließlich probabilistischer Bewertung) bereits zu Beginn einer MCDET-Simulation.

Zur Verbesserung der praktischen Anwendbarkeit des MCDET-Analysewerkzeugs (inklusive Probabilistik-Modul, Crew-Modul und Steuerungs-Modul) wurde damit begonnen, eine grafische Benutzeroberfläche zu konzipieren. Das Konzept sieht eine Menü- und Dialog-gesteuerte Kommunikation vor (ähnlich wie beim Analysewerkzeug SUSA für Unsicherheits- und Sensitivitätsanalysen). Die Programmierung der Oberfläche wird mittels der sogenannten Qt-Bibliothek erfolgen. Bei Qt handelt es sich um eine C^{++}-Klassenbibliothek für die plattformübergreifende Programmierung grafischer Benutzeroberflächen.

Im Zusammenhang mit der Entwicklung der Benutzeroberfläche wurde zudem ein neues Konzept für eine einfachere Anwendung des Crew-Moduls entwickelt. In der ursprünglichen Version des Crew-Moduls mussten Restart-Informationen definiert werden, was sich als sehr umständlich, aufwändig und fehleranfällig erwiesen hat. Durch die Einführung eines sogenannten Key-Vektors ist dies zukünftig nicht mehr erforderlich.

Im Rahmen dieses Vorhabens wurde die bisherige MCDET-Version erfolgreich angewendet, um ein komplexes Brand-Szenario zu analysieren /PES 14/. Gegenstand der Analyse waren die Brandbekämpfungsmaßnahmen in einem Kernkraftwerk, wenn die automatische Sprinkleranlage im Brandraum nicht funktioniert. Die Analyse machte intensiven Gebrauch vom MCDET Crew-Modul, um die Personalhandlungen zur Brandbekämpfung zu modellieren. Zur Simulation der Brandentwicklung wurde das Rechenprogramm FDS (*F*ire *D*ynamics *S*imulator) angewendet.

Die komplexe Anwendung im Rahmen des Forschungs-und Entwicklungsvorhabens RS1198 bestätigt ergänzend zu den umfangreichen Analysebeispiele in der Vergangenheit (siehe /HOF 01/ und /PES 06/) die praktische Anwendbarkeit von MCDET mit

einem mittlerweile vertretbaren Rechenaufwand. Zudem verdeutlichen sie den Informationsgewinn, den Analysen mittels MCDET für die Bewertung der Reaktorsicherheit bieten können. Für unterschiedliche zeitliche Entwicklungen von Ereignisabläufen erhält man Informationen sowohl zu entsprechenden Prozessabläufen als auch zu den Wahrscheinlichkeiten für diese Prozessabläufe. Für sicherheitsbezogene Bewertungen lassen sich daraus dann Wahrscheinlichkeiten für kritische Prozesszustände ermitteln.

Die erheblichen Fortschritte bei der Weiterentwicklung der MCDET-Methodik und des zugehörigen Analysewerkzeugs wurden auf diversen internationalen Fachkonferenzen und Workshops vorgestellt und mit einem breiten Fachpublikum diskutiert /KLO 11/, /KLO 11a/, /KLO 12/, /KLO 12a/, /KLO 12b/, /KLO 13/, /KLO 13a/, /KLO 13b/, /KLO 13c/, /PES 12/, /PES 12a/.

Mit den im Vorhaben RS1198 realisierten Weiterentwicklungen wurde die Zielsetzung dieses Arbeitspunktes erreicht. Eine Erprobung der neuen MCDET-Version an einem komplexen Anwendungsbeispiel in Verbindung mit dem Rechenprogramm ATHLET-CD ist im Rahmen eines Nachfolgevorhabens geplant. Als Anwendungsfall soll dabei ein nach den Erkenntnissen der Reaktorunfälle von Fukushima Dai-ichi als relevantes Unfallszenario erkanntes Szenario mit Ausfall der externen Stromversorgung (‘Station Black-Out‘) mit thermisch induziertem Dampferzeugerheizrohr-Versagen analysiert und bewertet werden.

Literaturverzeichnis

/ADO 11/ Adolfsson, Y., et al. (eds.): In Proceedings of the Deterministic/Probabilistic Safety Analysis Workshop. VTT-R-07266-11, VTT, Espoo, Finnland, Oktober 2011.

/ADO 12/ Adolfsson, Y., et al. (eds.): In IDPSA-Integrated Deterministic-Probabilistic Safety Assessment, 2nd Workshop, KTH, Stockholm, November 19-21, 2012.

/ALD 13/ Aldemir, T.: A survey of dynamic methodologies for probabilistic safety assessment of nuclear power plants, Annals of Nuclear Energy, 52, 2013, S. 113-124.

/BRU 13/ Brunett, A., et al.: Dynamic assessment of low probability containment failure modes, ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis, Columbia, SC, USA, September 22 - 26, 2013.

/CAT 10/ Catalyurek, U., et al.: Development of a code-agnostic computational infrastructure for the dynamic generation of accident progression event trees, Reliability Engineering and System Safety, 95, 2010, S. 278-294.

/DEN 13/ Denman, M. R., et al.: Discrete dynamic event tree analysis of small modular reactor severe accident management, ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis, Columbia, SC, USA, September 22 - 26, 2013.

/DEN 13a/ Denman, M. R., et al.: Safety relief valve cyclic failure analysis for use in discrete dynamic event trees, ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis, Columbia, SC, USA, September 22 - 26, 2013.

/HOF 01/ Hofer, E., et al.: Methodenentwicklung zur simulativen Behandlung der Stochastik in probabilistischen Sicherheitsanalysen der Stufe 2, GRS-A-2997, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, Garching, Dezember 2001.

/IZQ 94/ Izquierdo, J. M.: Automatic generation of dynamic event trees: a tool for integrated safety assessment, in: Aldemir, T. (Ed.): Reliability and safety assessment of dynamic process systems, Springer-Verlag, Heidelberg, 1994, S. 135-50.

/KLO 06/ Kloos, M., J. Peschke: MCDET - A Probabilistic Dynamics Method Combining Monte Carlo Simulation with the Discrete Dynamic Event Tree Approach, Nuclear Science and Engineering, 153, 2006, S. 137-156.

/KLO 11/ Kloos, M.: Research Activities of Germany´s GRS in the Field of Dynamic PSA, in: Proceedings of ANS PSA 2011 International Topical Meeting on Probabilistic Safety Assessment and Analysis, Wilmington, NC, March 13-17, 2011, on CD-ROM, American Nuclear Society, LaGrange Park, IL, USA, 2011.

/KLO 11a/ Kloos, M.: Capability of the MCDET Method in the Field of Dynamic PSA, in: Proceedings of Annual European Safety and Reliability Conference 2011 (ESREL 2011)1, Troyes, Frankreich, 17. - 23. September 2011.

/KLO 12/ Kloos, M., J. Peschke, W. Pointner: Advanced Method for Considering Epistemic and Aleatory Uncertainties in Integrated Deterministic Probabilistic Safety Analyses, Technical Meeting on Combining Insights from Deterministic and Probabilistic Safety Analyses, OECD Workshop, Pisa, 11–15 Juni 2012.

/KLO 12a/ Kloos, M., J. Peschke: MCDET Approach for Considering Epistemic and Aleatory Uncertainties in Accident Simulations, NUTHOS-9 Conference, Kaohsiung Taiwan, September 2012.

/KLO 12b/ Kloos, M., J. Peschke: Overview on the MCDET method and further developments, 2nd IDPSA-Workshop, KTH, Stockholm, 19.-21. November 2012.

/KLO 13/ Kloos, M.: Scope of results from probabilistic dynamics analyses with the MCDET tool, NURETH-15 Workshop W-05 on ‘Combining Deterministic and Probabilistic Methods for Comprehensive Safety Margin Assessment’, Pisa, Italien, 12. Mai, 2013.

/KLO 13a/ Kloos, M.: MCDET Tool, Workshop on Dynamic PSA, ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis, Columbia, SC, September 22, 2013.

/KLO 13b/ Kloos, M.: MCDET Demonstration, Workshop on Dynamic PSA, ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis, Columbia, SC, September 22, 2013.

/KLO 13c/ Kloos, M., J. Hartung, J. Peschke, J. Scheuer, : Updates and Current Application of the MCDET Simulation Tool for Dynamic PSA, in: Proceedings of ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis, Columbia, SC, September 22-26, 2013, on CD-ROM, American Nuclear Society, LaGrange Park, IL, USA, 2013.

/KLO 15/ Kloos, M.: Weiterentwicklung des Analysewerkzeugs SUSA für Unsicherheits- und Sensitivitätsanalysen im Rahmen einer fortschrittlichen PSA, GRS-, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gmbH, Köln, Garching, Mai 2015.

/KOT 82/ Kotz, S., N. L. Johnson: Encyclopedia of statistical sciences, Volume 1, John Wiley & Sons, 1982.

/MUN 99/ Munoz, R., et al.: A Second Generation Scheduler for Dynamic Event Trees, in: J. M. Aragones, C. Ahnert, and O. Cabellos. (Eds.), Senda Editorial, S. A., Madrid, Spain,1999.

/OTT 02/ Ottmann, T., P. Widmayer: Algorithmen und Datenstrukturen, 4. Auflage, Spektrum Akademischer Verlag; Heidelberg, Berlin, 2002.

/PES 06/ Peschke, J., M. Kloos, W. Faßmann, M. Sonnenkalb: Methodenentwicklung für die Berücksichtigung menschlicher Eingriffe im Rahmen einer dynamischen PSA der Stufen 1 und 2, GRS-A-3340, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, Garching, 2006.

/PES 12/ Peschke, J., M. Kloos: Options to Consider Reliability Information in a Dynamic PSA with the MCDET Method, in: 11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference 2012 (PSAM11 ESREL 2012), ISBN: 978-1-62276-436-5**,** Curran Associates, Inc., Red Hook, NY, 2012.

/PES 12a/ Peschke, J., M. Kloos: Analysis of a Station Black-Out Scenario, 2nd IDPSA-Workshop, KTH, Stockholm, 19.-21. November 2012.

/PES 14/ Peschke, J., et al.: Probabilistische Dynamikanalyse eines Brandszenarios unter Berücksichtigung von Brandbekämpfungsmaßnahmen durch das Anlagenpersonal, Technischer Fachbericht, GRS-331, ISBN 978-3-944161-11-2, -, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, Köln, Garching, Juni 2014.

/RAB 89/ Rabiner, L. R., B. H. Juang: An introduction to hidden Markov models, IEEE ASSP Magazine, Vol.3, No 1, S. 4-16, 1986.

/SIL 03/ Silberschatz A., P. B. Galvin, G. Gagne: Operating System Concepts, Sixth Edition, John Wiley & Sons, 2003.

/VOG 06/ Voggenberger, T., F. Cester, H. Deitenbeck: ATLAS XP - Erweiterung des Einsatzspektrums für komplexe Analyseanforderungen, GRS-A-3353, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, Garching, 2006.

/WIL 41/ Wilks, S.: Determination of sample sizes for setting tolerance limits, Annals of Mathematical Statistics 1 (1), S. 91-96, 1941.

/WIL 42/ Wilks, S.: Statistical prediction with special reference to the problem of tolerance limits, Annals of Mathematical Statistics 13 (4), S. 400-409, 1942.

/ZAM 13/ Zamalieva, D., A. Yilmaz, T. Aldemir: A probabilistic model for online scenario labeling in dynamic event tree generation, Reliability Engineering and System Safety, 120, S: 18-26, 2013.

/ZHO 14/ Zhong, S., H. Langseth, T. Dyre Nielsen: A classification-based approach to monitoring the safety of dynamic systems, Reliability Engineering and System Safety, 121, S. 61-71, 2014.

/ZIO 12/ Zio, E., F. Di Maio: Fault Diagnosis and Failure Mode Estimation by a Data-Driven Fuzzy Similarity Approach, International Journal of Performability Engineering, Vol. 8, No.1, S. 49-66, Januar 2012.

Abbildungsverzeichnis

Abb. 2.1 MCDET-Module für die Kopplung mit einem deterministischen Rechenprogramm und deren Zusammenspiel 9

Abb. 2.2 Ablauf der Berechnung eines dynamischen Ereignisbaumes 12

Abb. 2.3 Prozesssequenz bei der Berechnung eines dynamischen Ereignisbaums 17

Abb. 3.1 MCDET-Eingabedatei: Eingabe von Simulator-Variablen 20

Abb. 3.2 MCDET-Eingabedatei: Eingabe von Zeit-Zustandsvariablen 22

Abb. 3.3 MCDET-Eingabedatei: Eingabe von epistemischen Variablen 22

Abb. 3.4 MCDET-Eingabedatei: Eingabe von aleatorischen Variablen für die Berücksichtigung mittels Monte-Carlo-Simulation (Beispiel 1) 24

Abb. 3.5 MCDET-Eingabedatei: Spezifikation eines ‘Auswahlfensters‘ (Sampling Window), Beispiel 1 (vgl. Abb. 5.6) 27

Abb. 3.6 MCDET-Eingabedatei: Spezifikation eines ‘Verzweigungsfensters‘ (Transition Window) 28

Abb. 3.7 MCDET-Eingabedatei: Angaben zur Berechnung einer Verzweigungswahrscheinlichkeit 29

Abb. 3.8 MCDET-Eingabedatei: Spezifikation eines Funktionsfensters (Function Window) 29

Abb. 3.9 MCDET-Eingabedatei: Eingabe von aleatorischen Variablen für die Berücksichtigung mittels Monte-Carlo-Simulation, Beispiel 2 32

Abb. 3.10 MCDET-Eingabedatei: Initialisierung einer Simulator-Variablen mit einer aleatorischen Variablen 32

Abb. 3.11 MCDET-Eingabedatei: Spezifikation eines ‘Auswahlfensters‘ (Sampling Window), Beispiel 2 33

Abb. 3.12 MCDET-Eingabedatei: Spezifikation eines Verzweigungsfensters für die Zuordnung einer aleatorischen Variablen zu einer Simulator-Variablen 33

Abb. 3.13 MCDET-Eingabedatei: Spezifikation eines Verzweigungsfensters für eine Verzweigung am Anfang eines dynamischen Ereignisbaums 34

Abb. 4.1 Stichprobe dynamischer Ereignisbäume 38

Abb. 4.2 Stichprobe dynamischer Ereignisbäume mit baumspezifischen Verteilungen für eine Prozessgröße 39

Abb. 4.3 MCDET-Eingabedatei: Spezifikation von absorbierenden Fenstern (Absorbing Windows) 41

Abb. 5.1 Menü ‘Project‘ der MCDET-Menüleiste 45

Abb. 5.2 Menü ‘Variables‘ der MCDET-Menüleiste 46

Abb. 5.3 Dialog zur Eingabe von Simulator-Variablen (vgl. Abb. 3.1) 47

Abb. 5.4 Dialog zur Eingabe von epistemischen Variablen (vgl. Abb. 3.3) 47

Abb. 5.5 Menü ‘Windows‘ der MCDET-Menüleiste 48

Abb. 5.6 Auswahlfenster mit der Bedingung ‘Init‘ und der Berücksichtigung von Verteilungsparametern als feste Werte (vgl. Abb. 3.5) 49

Abb. 5.7 Auswahlfenster mit der Bedingung ‘Init‘ und der Berücksichtigung von Verteilungsparametern als epistemische Variable 49

Abb. 5.8 Auswahlfenster für den zufälligen Ausfallzeitpunkt einer Komponente (Beispiel 1) 50

Abb. 5.9 Auswahlfenster für den zufälligen Ausfallzeitpunkt einer Komponente (Beispiel 2) 50

Abb. 5.10 Verzweigungsfenster für Ausfall einer Komponente bei Anforderung 51

Abb. 5.11 Verzweigungsfenster für Ausfall einer Komponente zu einem zufälligen Zeitpunkt 51

Abb. 5.12 Funktionsfenster für Anzahl der Anforderungszyklen (vgl. Abschnitt 3.2.2) 51

Abb. 5.13 Absorbierende Fenster mit Bedingungen für die Beendigung eines Simulationslaufs 52

Abb. 5.14 Hauptdialogfenster des Crew-Moduls 53

Abb. 5.15 Dialog zur Spezifikation der an einer Handlung beteiligten Personen 58

Abb. 5.16 Mitteilung zur Anzahl der beteiligten Personen 58

Abb. 5.17 Dialog zur Spezifikation der Parameter im Key-Vektor 59

Abb. 5.18 Dialog zur Änderung eines Parameters im Key-Vektor 62

Abb. 5.19 Dialog zum Hinzufügen eines Parameters in den Key-Vektor 63

Abb. 5.20 Dialog zur Spezifikation der Alarm- bzw. Anzeigenbedingungen 65

Abb. 5.21 Meldung zur Eingabe der Identifikationsnummer 66

Abb. 5.22 Dialog zur Spezifikation von Alarmen, Anzeigen und Zustandsänderungen 67

Abb. 5.23 Meldung zur Eingabe von Zahlenwert oder Parameter 69

Abb. 5.24 Dialog nach Eingabe des ersten Bedingungsblocks 70

Abb. 5.25 Dialog zum Speichern der eingegebenen Anweisung 71

Abb. 5.26 Dialog zur Spezifikation von Basishandlungen 76

Abb. 5.27 Dialog zur Spezifikation der Basishandlung 1 80

Abb. 5.28 Dialog zur Spezifikation der Basishandlung 2 80

Abb. 5.29 Meldung zur Eingabe der Variable ‘b_infoDH-FH‘ 81

Abb. 5.30 Übersichtsliste nach Spezifikation der Basishandlungen 1 – 3 83

Abb. 5.31 Dialog zur Spezifikation von Handlungssequenzen 93

Abb. 5.32 Dialog zum Speichern der Handlungssequenz mit zugehöriger Bedingung 94

Abb. 5.33 Dialog zur Eingabe von Zustandsänderungen durch Basishandlungen.... 97

Abb. 5.34 Dialog zur Spezifikation von Zustandsänderungen durch Basishandlungen 98

Abb. 5.35 Dialog zum Speichern der Anweisung 99

Abb. 5.36 Menü ‘Simulator Runs‘ der MCDET-Menüleiste 100

Abb. 5.37 Menü ‘Post-Processing‘ der MCDET-Menüleiste mit Fokus auf das Untermenü ‘Time Series‘ 101

Abb. 5.38 Zeitliche Entwicklung der Kernaustrittstemperatur und der UO_2-Schmelzemasse für zwei ausgewählte dynamische Ereignisbäume aus der MCDET-Analyse eines Station-Blackout-Szenarios 102

Abb. 5.39 Zeitliche Entwicklung des Füllstands in einem Tank für zwei ausgewählte dynamische Ereignisbäume aus einer MCDET-Analyse des beispielhaften Tank-Systems (vgl. Anhang) 103

Abb. 5.40 Zeitliche Entwicklung der Temperatur innerhalb eines Kabelmantels für alle 60 dynamischen Ereignisbäume aus einer MCDET-Analyse eines Brandszenarios 103

Abb. 5.41 Zeitliche Abfolge von Ereignissen für eine Auswahl von Sequenzen aus einem dynamischen Ereignisbaum (Symbole kennzeichnen berücksichtigte Ereignisse in einem Unfallszenario) 104

Abb. 5.42 Menü ‘Post-Processing‘ der MCDET-Menüleiste mit Fokus auf das Untermenü ‘Probability Distribution‘ 105

Abb. 5.43 Verteilungen der maximalen Kernaustrittstemperatur und der insgesamt erzeugten UO_2-Schmelzemasse für zwei ausgewählte dynamische Ereignisbäume aus der MCDET-Analyse eines Station-Blackout-Szenarios 105

Abb. 5.44 Verteilung der insgesamt erzeugten H_2-Masse inklusive des Beitrags der einzelnen Sequenzen für einen ausgewählten dynamischen Ereignisbaum aus der MCDET-Analyse eines Station-Blackout-Szenarios 106

Abb. 5.45 Verteilungen des Füllstands in einem Tank für zwei ausgewählte dynamische Ereignisbäume aus einer MCDET-Analyse des beispielhaften Tank-Systems (vgl. Anhang) 106

Abb. 5.46 Szenario-spezifische Verteilung der insgesamt erzeugten H_2-Masse über alle 50 dynamischen Ereignisbäume aus der MCDET-Analyse eines Station-Blackout-Szenarios 107

Abb. 5.47 Szenario-spezifische Verteilungen der Temperatur innerhalb eines Kabelmantels über alle 60 dynamischen Ereignisbäume aus der MCDET-Analyse eines Brandszenarios 107

Abb. 5.48 Menü ‘Epistemic Uncertainty‘ der MCDET-Menüleiste 108

Glossar

BHB	Betriebshandbuch
BMWi	Bundesministerium für Wirtschaft und Energie
DH-FH	Druckhalter-Füllstandshöhe
DSA	Deterministische Sicherheitsanalyse
HKP	Hauptkühlmittelpumpe
IDPSA	Integrated Deterministic Probabilistic Safety Analysis
I/O-Last	Eingabe/Ausgabe-Last
MCDET	Monte Carlo Dynamic Event Tree
NHB	Notfallhandbuch
PSA	Probabilistische Sicherheitsanalyse
RF	Reaktorfahrer
SL	Schichtleiter

A Anhang: Tank-System

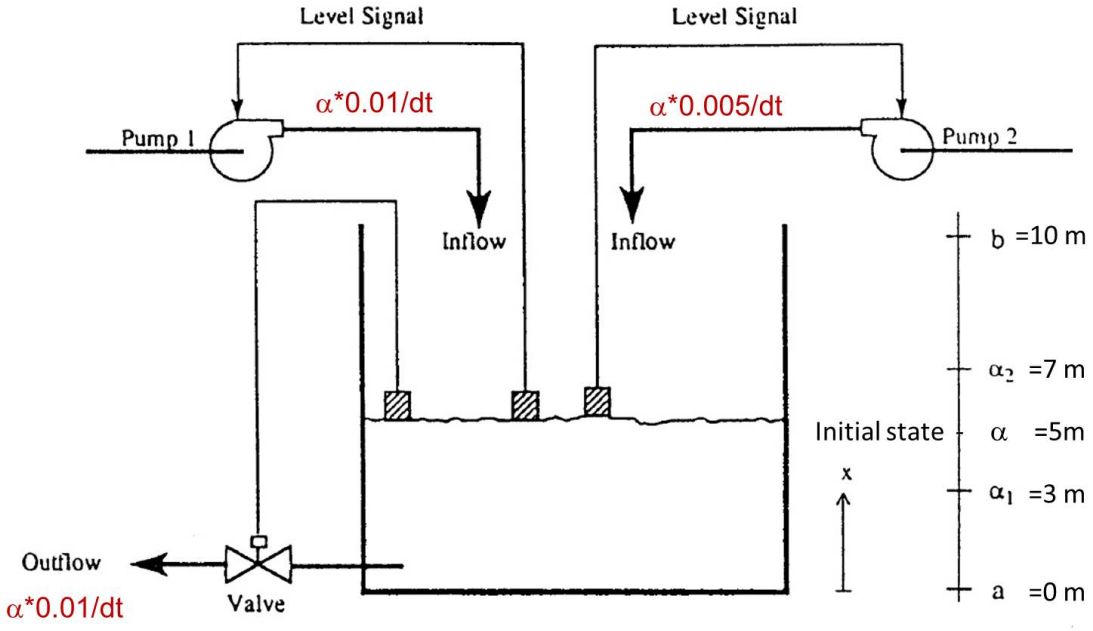


Abb. A.1 Tank-System inklusive wichtiger Kenngrößen

Tank-System inklusive wichtiger Kenngrößen

Das Tank-System besteht aus einem mit Flüssigkeit gefüllten Speicher-Behälter. Die Füllhöhe der Flüssigkeit in diesem Behälter wird durch drei aktive Komponenten - zwei Pumpen und ein Ventil - unter Verwendung eines Regelsystems gesteuert. Die Pumpen haben die Aufgabe, Flüssigkeit in den Behälter nachzufüllen, wenn die Flüssigkeit auf einen bestimmten Füllstand $\alpha 1$ abgesunken ist. Überschreitet der Füllstand einen bestimmten Wert $\alpha 2$, sorgt das Ventil dafür, Flüssigkeit aus dem Behälter bis unter diesen Wert $\alpha 2$ abzulassen. Die Komponenten erfüllen also die Aufgabe, den Flüssigkeitsstand im Behälter nicht zu tief absinken bzw. zu hoch ansteigen zu lassen.

Unerwünschte Zustände des Tank-Systems treten ein,

- wenn der Füllstand den Wert a = 0 m erreicht
- wenn die Flüssigkeit über den oberen Rand des Behälters läuft, d. h. wenn der Füllhöhestand den Wert b = 10 m überschreitet.

Die drei Komponenten des Tank-Systems arbeiten unabhängig voneinander und können die Zustände ‘an’, ‘aus’ und ‘ausgefallen’ bei den Pumpen und ‘offen’, ‘geschlossen’ und ‘ausgefallen’ beim Ventil annehmen. Die Signalgebung zum aktuellen Füllstand ist zuverlässig. Allerdings gibt es Fehlsignale, die zum unbeabsichtigten Starten und Stoppen der Pumpen bzw. Öffnen und Schließen des Ventils führen.

Der Zustand ‘ausgefallen’ wird unterschieden in ‘offen ausgefallen’ und ‘geschlossen ausgefallen’. Für das Ventil heißt ‘offen ausgefallen’, dass es sich nicht mehr schließen lässt, und für die Pumpen, dass sie nicht mehr ausgeschaltet werden können. ‘Geschlossen ausgefallen’ heißt für das Ventil, dass es sich nicht mehr öffnen lässt, und für die Pumpen, dass sie nicht mehr gestartet werden können.

Eine detaillierte Beschreibung des Tank-Systems ist in /HOF 01/ zu finden.

Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH

Schwertnergasse 1
50667 Köln
Telefon +49 221 2068-0
Telefax +49 221 2068-888

Forschungszentrum
85748 Garching b. München
Telefon +49 89 32004-0
Telefax +49 89 32004-300

Kurfürstendamm 200
10719 Berlin
Telefon +49 30 88589-0
Telefax +49 30 88589-111

Theodor-Heuss-Straße 4
38122 Braunschweig
Telefon +49 531 8012-0
Telefax +49 531 8012-200

www.grs.de

ISBN 978-3-944161-10-5