Methodische und programmtechnische Weiterentwicklung des Werkzeugs MCDET zur Durchführung von integrierten deterministisch-probabilistischen Sicherheitsanalysen

GRS - 686

Gesellschaft für Anlagen-
und Reaktorsicherheit
(GRS) gGmbH

Methodische und programmtechnische Weiterentwicklung des Werkzeugs MCDET zur Durchführung von integrierten deterministisch-probabilistischen Sicherheitsanalysen

Jörg Peschke
Tanja Eraerds
Martina Kloos
Josef Scheuer
Jan Soedingrekso

November 2022

Anmerkung:

Das diesem Bericht zugrunde liegende Forschungsvorhaben wurde mit Mitteln des Bundesministeriums für für Umwelt, Naturschutz, nukleare Sicherheit und Verbraucherschutz (BMUV) unter dem Kennzeichen RS1570 durchgeführt.

Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei der GRS.

Der Bericht gibt die Auffassung und Meinung der GRS wieder und muss nicht mit der Meinung des BMUV übereinstimmen.

GRS - 686
ISBN 978-3-949088-77-3

Deskriptoren

CrewModul, Dynamische PSA, Integrierte Deterministisch-Probabilistische Sicherheitsanalyse, MCDET

Kurzfassung

In dem Vorhaben RS1570 mit wurden methodische und programmtechnische Weiterentwicklung des Werkzeugs MCDET zur Durchführung von integrierten deterministisch-probabilistischen Sicherheitsanalysen durchgeführt. Das Vorhaben bezieht sich auf das in /BMWI 21/ festgeschriebene Forschungsgebiet A zur ‚Reaktorsicherheit' und dort insbesondere auf die Zielsetzung der Forschungsförderung im FuE-Bereich A3 zur ‚Wechselwirkung Mensch-Technik und probabilistische Sicherheitsanalysen'.

Zur Fortführung der Weiterentwicklung des Analysewerkzeugs MCDET (Monte Carlo Dynamic Event Tree) und des CrewModuls zur dynamischen Modellierung und Simulation von menschlichen Handlungsabläufen unter Berücksichtigung von Unsicherheiten und Abhängigkeiten des Handlungsablaufs von Prozesszuständen wurden im Vorhaben RS1570 Arbeiten zu folgenden Themen durchgeführt:

- Erweiterung des Methodenspektrums für MCDET-Analysen. Dazu wurden zusätzliche Methoden entwickelt, mit denen zum einen Simulationsergebnisse aus der Anwendung von MCDET mit einem Rechenprogramm ausgewertet werden können und zum anderen die Einsatzmöglichkeiten von MCDET erweitert werden. So wird unter anderem eine mathematisch-statistische Methode zur Quantifizierung von Einflussgrößen auf Ergebniscluster bereitgestellt. Des Weiteren wurde eine Methode zur Ermittlung von Primimplikanten bereitgestellt, mit der im Rahmen von MCDET-Analysen zusätzlich Ereignisse für Primimplikanten berücksichtigt werden, die z. B. durch die Zeitpunkte ihres Auftretens gekennzeichnet sind.Daneben wurden auch programmtechnische Methoden entwickelt, mit denen die Reproduzierbarkeit von MCDET-Analysen gewährleistet werden können und bereits erstellte dynamische Ereignisbäume nachträglich erweitert werden können, ohne dies insgesamt neu berechnen zu müssen. Durch nachhaltige, moderne Implementierungskonzepte wurden diese Methoden so umgesetzt, dass die technischen Ressourcen (Rechenzeit, Rechenaufwand und verfügbarer Speicherplatz) effizient genutzt werden.
- Weiterentwicklung des Scheduling-Systems zum flexiblen und effizienten Einsatz von MCDET. Wesentliche Weiterentwicklungen zum Scheduling-System sind die Umstellung des MCDET-Eingabeformats auf Python-Basis und die damit verbundene Umstellung des MCDET-Kerns. Weitere wichtige Weiterentwicklungen sind die Umstellung des Crew-Moduls von MCDET auf Python-Basis und die Kopplung des Crew-Moduls mit der neuen Scheduler-Version von MCDET.

- Weitentwicklung der interaktiven Anwendung von MCDET. Wichtige Weiterentwicklungen beziehen sich auf die Erstellung des MCDET-Eingabedatensatzes, der durch das neue Python-basierte Eingabeformat den Nutzer interaktiv durch eine standardisierte Syntax und eine klare Eingabestruktur unterstützt, das Starten und Monitoren von Simulationsprozessen des Tandems MCDET/Rechenprogramm und auf die Auswertung der Simulationsergebnisse. Die Schritte des Postprocessings und die dafür geeigneten Visualisierungsmethoden wurden verfeinert und beispielhaft in Form von Jupyter Notebooks implementiert.

Abstract

In the project RS1570 the MCDET tool for integrated deterministic-probabilistic safety analyzes was further developed. The project relates to the objective on 'interaction between humans and technology and probabilistic safety analyses' defined in research area A on 'reactor safety' /BMWI 21/.

To further develop the analysis tool MCDET (Monte Carlo Dynamic Event Tree) and the CrewModule for dynamic modeling and simulation of human actions under consideration of uncertainties and dependencies following work was carried out:

- Development of additional methods for MCDET analyses. For this purpose, further methods were developed for analyzing data gathered from MCDET simulations and on the other hand to improve and extend the applicability of the tool MCDET as well as the CrewModule. A mathematical method was developed for quantifying the influence of parameters on sequence cluster which result from an MCDET-analysis. Furthermore, a method for determining prime implicants from results of a MCDET-analysis was developed. With this method events for prime implicants can also be taken into account which are characterized, for example, by the time of their occurrence. In addition, programming methods were developed which ensure the reproducibility of MCDET analyzes and which allow a subsequent expansion of dynamic event trees that have already been calculated without recalculating them in total. These methods were implemented by using modern implementation concepts to efficiently use the technical resources as computing time, computing effort and available storage space.

- Further development of the scheduling system for the flexible and efficient use of MCDET. Essential developments of the scheduling system refer to the conversion of the MCDET-input to a Python-based input format and the associated conversion of the MCDET core. Other important developments are the conversion of the CrewModule in Python code and the coupling of the CrewModule with the new scheduler version of MCDET.

- Further development of the interactive application of MCDET. The new Python-based input format supports the user interactively with a standardized syntax and a clear input structure. Developments have been carried out to start and monitor the simulation processes of the tandem of MCDET and the deterministic computer program and the evaluation the simulation results. The post-processing steps and the

visualization methods suitable for this task were refined and implemented as examples in the form of Jupyter notebooks.

Inhaltsverzeichnis

1 Einleitung

Das Analysewerkzeug MCDET (Monte Carlo Dynamic Event Tree) wurde mit der Zielsetzung entwickelt, Unsicherheiten und zeitabhängige Wechselwirkungen in Sicherheitsanalysen umfassend und realitätsnah berücksichtigen zu können und viele in einer klassischen probabilistischen Sicherheitsanalyse (PSA) notwendige Einschränkungen und vereinfachende Annahmen zu vermeiden.

Durch die Kopplung von MCDET mit einem deterministischen Rechenprogramm (z. B. ATHLET) wird die Durchführung einer integrierten deterministisch-probabilistischen Sicherheitsanalyse (IDPSA) von komplexen Systemen ermöglicht. Im Rahmen solcher Analysen können Abläufe, die sich durch die komplexen Wechselwirkungen zwischen physikalischem Prozess, System- und Komponentenverhalten, menschlichen Handlungen sowie zufälligen Einflüssen im zeitlichen Verlauf ergeben, simuliert und mit Eintrittswahrscheinlichkeiten bewertet werden. Damit ist MCDET neben der PSA insbesondere auch für eine deterministische Sicherheitsanalyse (DSA) geeignet, bei der im Rahmen einer erweiterten (extended) ‚Best Estimate Plus Uncertainty' (eBEPU) Analyse neben den epistemischen Unsicherheiten auch die aleatorischen Unsicherheiten bzgl. des Verhaltens von Sicherheitssystemen detaillierter berücksichtigt werden.

Obwohl bereits umfangreiche Entwicklungsarbeiten zu MCDET durchgeführt worden sind, müssen diese fortgesetzt und erweitert werden, um MCDET einerseits kontinuierlich an den Stand von Wissenschaft und Technik anzupassen und andererseits möglichst einfach, effizient und flexibel zu machen, damit es einem breiten Nutzerkreis zugänglich wird.

Folgende Weiterentwicklungen wurden durchgeführt:

i) Erweiterung des Methodenspektrums:

   Es wurden zusätzliche Methoden entwickelt, mit denen zum einen Simulationsergebnisse aus der Anwendung von MCDET mit einem Rechenprogramm ausgewertet werden können und zum anderen die Einsatzmöglichkeiten von MCDET erweitert werden.

   So wird u. a. eine mathematisch-statistische Methode bereitgestellt zur Ermittlung von Einflussgrößen auf Ergebniscluster. Mit dieser Methode kann bestimmt werden,

welche Wertekombinationen stochastischer Einflussgrößen am meisten zur Bildung bestimmter Sequenzcluster bzw. Ergebniscluster beitragen.

Des Weiteren wird eine Methode zur Ermittlung von Primimplikanten bereitgestellt. Primimplikanten in einer dynamischen PSA stellen eine Erweiterung des Minimal-Cut-Set (MCS) Konzepts aus der klassischen PSA dar. Primimplikanten sind minimale Mengen von Ereignissen, die zu unerwünschten Prozesszuständen führen, wenn sie gemeinsam auftreten. Durch die entwickelten Methoden können im Rahmen von MCDET-Analysen zusätzlich Ereignisse für Primimplikanten berücksichtigt werden, die z. B. durch die Zeitpunkte ihres Auftretens gekennzeichnet sind.

Neben den rein mathematisch-statistischen Methoden wurden auch mehr programmtechnische Methoden entwickelt, mit denen MDET-Analysen für zusätzliche Fragestellungen erweitert werden können. Dazu gehören z. B. Methoden zur Gewährleitung der Reproduzierbarkeit von MCDET-Analysen sowie zur Verwendung bereits erstellter dynamischer Ereignisbäume, um in einer ergänzenden Analyse zusätzliche Fragestellungen beantworten zu können. Durch nachhaltige, moderne Implementierungskonzepte wurden diese Methoden so umgesetzt, dass die technischen Ressourcen (Rechenzeit, Rechenaufwand und verfügbarer Speicherplatz) effizient genutzt werden.

ii) Weiterentwicklung des Scheduling-Systems zur Erweiterung der Anwendungsmöglichkeiten von MCDET:

Bei der im Vorgängervorhaben RS1529 /PES 18/ erfolgten Entwicklung eines neuen Scheduler-Systems für MCDET wurde der Schwerpunkt auf die Kopplung von MCDET mit den GRS-eigenen Programmen ATHLET bzw. ATHLET-CD gelegt. Dabei konnte in den Quellcode eingegriffen werden, wodurch die Kommunikation zwischen MCDET und ATHLET bzw. ATHLET-CD sehr komfortabel und effizient gestaltet werden konnte. Um den Anwendungsbereich von MCDET auf deterministische Rechenprogramme zu erweitern, in deren Quellcode nicht eingegriffen werden kann (closed-source Rechencodes), wurde im Vorhaben RS1570 ein Konzept für eine erforderliche Erweiterung des Scheduling-Systems entwickelt.

Wesentliche Weiterentwicklungen zum Scheduling-System sind die Umstellung des MCDET-Eingabeformats auf Python-Basis und die damit verbundene Umstellung des MCDET-Kerns. Durch das Python-basierte Eingabeformat wird dem Nutzer jetzt eine einfache und komfortable Erstellung des Eingabe-Datensatzes für eine

MCDET-Analyse ermöglicht. Außerdem erlaubt das neue Eingabeformat eine bessere Überprüfung von Eingabefehlern.

Weitere wichtige Weiterentwicklungen sind die Umstellung des Crew-Moduls von MCDET auf Python-Basis und die Kopplung des Crew-Moduls mit dem neuen Scheduler.

iii) Weitentwicklung der Benutzeroberfläche zur interaktiven Anwendung von MCDET:

Wichtige Weiterentwicklungen zur Benutzeroberfläche beziehen sich auf die Erstellung des MCDET-Eingabedatensatzes, der durch das neue Python-basierte Eingabeformat den MCDET-Nutzer interaktiv durch eine standardisierte Syntax und eine klare Eingabestruktur unterstützt, das Starten und Monitoren von Simulationsprozessen des Tandems MCDET/Rechenprogramm und auf die Auswertung der Simulationsergebnisse aus der Anwendung dieses Tandems. Die Schritte des Post-Processings und die dafür geeigneten Visualisierungsmethoden wurden verfeinert und beispielhaft in Form von Jupyter Notebooks implementiert.

Für viele Auswertungen ist es entscheidend, die Daten der Simulationspfade synchronisiert mit den Meta- und Ereignisdaten des MCDET-Kerns (Estimator) zu betrachten. Hierzu müssen die entlang der Pfade zusammengesetzten HDF5-Datentabellen in sog. Table-Joins zusammengeführt werden. Da diese Operationen bislang relativ aufwändig und fehlerträchtig durchgeführt werden mussten, wurde begonnen diese als generische Implementierung bereitzustellen, welche dann in den Visualisierungen, wie Jupyter Notebooks, eine einfache aber gleichzeitig flexible Datenabfrage erlaubt.

iv) Erstellung eines Benutzerhandbuchs für MCDET-Analysen und Analysen menschlicher Handlungsabläufe mit dem CrewModul:

Für die Benutzerführung bei MCDET-Analysen wurden gut dokumentierte Jupyter Notebooks breitgestellt, die den Nutzer mit den Schritten einer MCDET-Analyse vertraut machen. Für die Erstellung eines Benutzerhandbuchs für das CrewModul erfolgte eine detaillierte Beschreibung zur Eingabe eines Handlungsablaufs, der von stochastischen Einflüssen und Prozesszustanden abhängen kann, in die grafische Eingabeoberfläche des Crew-Moduls, für die aktuell das Mind-Mapping Tool „FreeMind“ verwendet wird.

Dieser Bericht ist wie folgt gegliedert:

- In Abschnitt 2 werden die Arbeiten zur Erweiterung des Methodenspektrums von MCDET beschrieben. Dabei werden in Abschnitt 2.1 die Konzepte vorgestellt, die eine Re-Evaluierung geänderter MCDET-Inputs auf der Basis einer bereits durchgeführten MCDET-Analyse ermöglichen. In Abschnitt 2.2 wird beschrieben, wie mit MCDET auch eine klassische Monte-Carlo-Simulation ohne Verzweigungen durchgeführt werden kann. Auf die Reproduzierbarkeit von MCDET-Analysen wird in Abschnitt 2.3 eingegangen. In Abschnitt 2.4 wird die Methode zur Ermittlung von Einflussgrößen auf Ergebniscluster vorgestellt. Die Methode zur Ermittlung von Primimplikanten wird in Abschnitt 2.5 beschrieben.

- Abschnitt 3 behandelt die Weiterentwicklungen zum Scheduling-System. Abschnitt 3.1 beschreibt die durchgeführte Umstellung des MCDET-Kerns. In Abschnitt 3.2 wird auf die Erweiterung von Zugriffsmöglichkeiten von MCDET auf ATHLET/ATHLET-CD-Größen eingegangen. Abschnitt 3.3 beschreibt die Anpassungen, die für die Nutzung eines Linux-Clusters für MCDET-Analysen notwendig sind.

- In Abschnitt 4 werden die Weiterentwicklungen zur interaktiven Anwendung von MCDET und zur Neustrukturierung des MCDET-Inputs beschrieben. Abschnitt 4.1 geht auf die Neustrukturierung des MCDET-Inputs ein. Diese hängt sehr stark mit der Umstellung des MCDET-Kerns (siehe Abschnitt 3.1) zusammen. Abschnitt 4.2 beschreibt die Entwicklungen zur einfacheren Spezifikation von Rechencode-Größen in MCDET. Auf die Entwicklung einer grafischen Benutzeroberfläche wird in Abschnitt 4.3 eingegangen

- Eine Benutzerführung für das klassische CrewModul ist in Abschnitt 5 enthalten. Mit dieser Benutzerführung wird der erste Baustein für das MCDET-Benutzerhandbuch gelegt. Um eine grundlegende Idee des CrewModuls zu erhalten, wird in Abschnitt 5.1 das Konzept und die Vorgehensweise für das CrewModul beschrieben. In Abschnitt 5.2 wird im Detail beschrieben, wie ein Handlungsablauf in der graphischen Oberfläche des CrewModuls zu spezifizieren ist.

2 Erweiterung des Methodenspektrums von MCDET

In den Abschnitten 2.1, 2.2 und 2.3 werden zunächst programmtechnische Verfahren beschrieben, mit denen MDET-Analysen für weitere Fragestellungen möglichst effizient erweitert werden können. Des Weiteren werden in den Abschnitten 2.4 und 2.5 mathematisch-statistische Methodenentwicklungen beschrieben. Diese sollen die Frage beantworten, welche Wertekombinationen stochastischer Einflussgrößen am meisten zur Bildung bestimmter Sequenzcluster bzw. Ergebniscluster beitragen. In Abschnitt 2.2 wird dazu die neu entwickelte Methode ASPIC (Quantitative Assessment of Parameter Influence on Sequence Cluster) beschrieben und an einem Anwendungsbeispiel erprobt. In Abschnitt 2.5 wird eine Methode zur Ermittlung von Primimplikanten im Rahmen einer MCDET-Analyse beschrieben. Primimplikanten stellen die Erweiterung des MCS-Konzepts aus der klassischen PSA dar und können im Rahmen einer IDPSA mit MCDET zusätzlich durch die Zeitpunkte ihres Auftretens gekennzeichnet sein. Bekannte Methoden aus der Literatur sind z. B. Karnaugh Mapping, der Quine-McCluskey Algorithmus oder evolutionäre Optimierungsmethoden (differential evolution), die iterativ die besten Kandidaten aus einem Satz von möglichen Kandidaten ermitteln. In diesem Vorhaben werden erstmals Methodenentwicklungen zur Erzeugung und Verwendung von Primimplikanten im Rahmen einer MCDET-Analyse durchgeführt.

2.1 Re-Evaluierung geänderter MCDET-Inputs auf Basis einer bereits durchgeführten MCDET-Analyse

Anwendungen haben gezeigt, dass sich im Rahmen der Auswertung von Ergebnissen aus einer durchgeführten MCDET-Analyse im Nachhinein zusätzliche Fragestellungen ergeben können, deren Untersuchung einen wichtigen Beitrag zur Erweiterung des Wissensstandes der Analyse liefern können. Dabei ist oftmals die Situation gegeben, dass z. B. zusätzliche stochastische Einflussgrößen oder eine Verlängerung der Rechnungen der Unfallsequenzen im Rahmen der MCDET-Analyse interessant gewesen wären. Um diese Aspekte zu berücksichtigen, musste bisher die gesamte rechenzeitintensive MCDET-Analyse nochmals durchgeführt werden. Eine erhebliche Ersparnis des Rechenaufwandes kann aber dadurch erreicht werden, wenn die bisher gerechneten Sequenzen beibehalten und nur die neuen bzw. veränderten Teilsequenzen gerechnet und dem dynamischen Ereignisbaum hinzugefügt werden.

Es gibt zwei Möglichkeiten, wie eine nachträgliche Erweiterung einer bereits durchgeführten MCDET-Analyse durchgeführt werden kann. Abschnitt 2.1.1 beschreibt die

Thematik und ein Konzept zur Durchführung einer nachträglichen Verlängerung der Rechenzeit von Sequenzen (siehe Abb. 2.1 (a)). Die nachträgliche Erweiterung von bereits verfügbaren DETs um zusätzliche Sub-DETs (siehe Abb. 2.1 (b)) wird in Abschnitt 2.1.2 beschrieben. In Abschnitt 2.1.3 wird beschrieben, wie eine Re-Evaluierung eines bestehenden DETs (siehe Abb. 2.1 (c)) durch alternative Pfadwahrscheinlichkeiten aufgrund epistemischer Unsicherheiten durchgeführt werden kann.

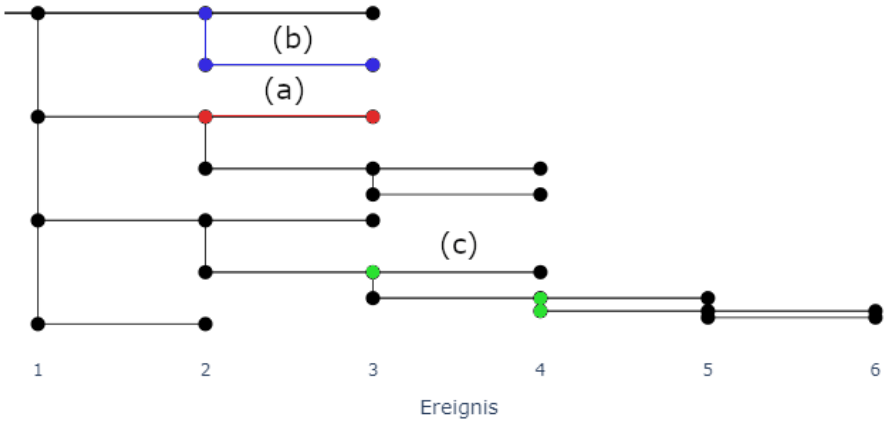


Abb. 2.1 Re-Evaluierung eines bestehenden DETs durch nachträgliche Verlängerung von Sequenzen (a), Hinzufügen von Sub-DETs durch die Abzweigung zusätzlicher Pfade (b) und die Berücksichtigung alternativer Pfadwahrscheinlichkeiten an bestimmten (grün markierten) Verzweigungspunkten (c)

2.1.1 Nachträgliche Verlängerung der Rechenzeit

Eine nachträgliche Verlängerung der Rechenzeit bestehender Sequenzen kann in folgenden Situationen erwünscht sein:

i) Für die ursprünglich durchgeführte Analyse wurde ein Ende der Rechenzeit von T_{end} gewählt. D. h., alle Sequenzen, die nicht vorher durch ein anderes Abbruchkriterium beendet wurden, wurden maximal bis zum Zeitpunkt T_{end} gerechnet. Wenn sich im Rahmen des Post-Processings herausstellt, dass sich neben den Sequenzen, bei denen man sich relativ sicher sein kann, dass sie im sicheren oder unerwünschten Zustand verbleiben, auch Sequenzen identifizieren, bei denen der weitere Verlauf unsicher ist, kann für diese Sequenzen eine nachträgliche Verlängerung der Rechenzeit wünschenswert sein. Damit könnte man erkennen, ob und wann diese zunächst unsicheren Sequenzen im weiteren Verlauf in einen sicheren bzw. unerwünschten Zustand laufen.

ii) In einer MCDET-Analyse kann eine sog. Cut-off-Wahrscheinlichkeit $p_{cut\text{-}off}$ vom Benutzer spezifiziert werden. Wenn bei einer Verzweigung für eine alternative Sequenz eine Wahrscheinlichkeit p berechnet wird, die kleiner als $p_{cut\text{-}off}$ ist, so wird die Sequenz mit $p < p_{cut\text{-}off}$ nicht mehr gerechnet. Wenn in einer Analyse mehrere Ereignisalternativen mit kleinen Wahrscheinlichkeiten vorkommen, so kann der Wert von $p_{cut\text{-}off}$ relativ schnell unterschritten werden. Wenn im Nachhinein erkannt wird, dass zu viele interessante Sequenzen aufgrund der Unterschreitung von $p_{cut\text{-}off}$ nicht gerechnet wurden, sollte die Möglichkeit bestehen, auf den abgebrochenen Sequenzen aufzusetzen und diese der bestehenden Analyse nachträglich hinzuzufügen. Damit können auch die Auswirkungen von interessanten Ereigniskombinationen, die zwar mit sehr kleiner Wahrscheinlichkeit auftreten, aber zu großen Schädigungen führen können, umfassender berücksichtigt werden.

Im Rahmen des Projektes wurde untersucht, nach welchem Konzept eine nachträgliche Verlängerung der Rechenzeit von Sequenzen für eine MCDET-Anwendung umgesetzt werden kann. Bei einer MCDET-Anwendung, die mit komplexen, langlaufenden Rechnungen eines deterministischen Rechencodes verbunden ist, spielt die Abwägung der benötigten Rechenzeit eine wesentliche Rolle. Denn auch bei den heute verfügbaren Rechenleistungen kann die Anzahl der zu rechnenden Sequenzen so groß werden, dass eine Anwendung in vollem Umfang nicht praktikabel ist. Die Überlegungen, die zur Konzeptentwicklung geführt haben, wurden deshalb alle vor dem Hintergrund durchgeführt, den benötigten Rechenaufwand möglichst gering zu halten.

Erster Schritt hin zu einer Sequenzverlängerung ist das Post-Processing. Im Post-Processing müssen die oben beschriebenen Sequenzen erkannt und selektiert werden. Eine Herausforderung in der Bestimmung der Sequenzen, die oben unter i) benannt wurden, ist es, ein Kriterium für die Einordnung der Sequenzen zu finden. Sequenzen, bei denen sich das System bereits über einen längeren Zeitraum im gleichen Zustand befindet und keine zufälligen Veränderungen des Systemzustands mehr stattfinden, können z. B. eindeutig eingeordnet werden. Sequenzen, bei denen die weitere Entwicklung des Systemzustands nicht eindeutig vorhergesagt werden kann, können durch eine nachträgliche Verlängerung der Rechenzeit länger gerechnet werden und damit Aufschluss auf die weitere Entwicklung geben.

Das Auswählen von Sequenzen für die Re-Evaluierung erfolgt im Post-Processing, und zwar auf Basis der MCDET-Ausgabedatei (HDF5). Die hier nutzbaren Methoden erlauben es, die Auswahlkriterien für die fortzusetzenden Sequenzen sehr flexibel zu

formulieren und bestimmen die entsprechenden HDF5-Ergebnisknoten (Tabellen/Gruppen). Anhand dieser Ergebnisknoten können alle Informationen ermittelt werden, welche für das Aufsetzen neuer Sequenzsimulationen notwendig sind. Zu den wichtigsten zählen hierbei

- der Simulationszeitpunkt an dem eine Sequenzsimulation aufsetzen soll,
- der am oder möglichst kurz vor dem Simulationszeitpunkt liegende Simulationszustand (Restart-File) und
- das Arbeitsverzeichnis der Eltern-Simulation bzw. der Simulation, die fortgesetzt werden soll.

Wie in der Auflistung oben ersichtlich, ist ein sog. Restart-File eine notwendige Voraussetzung für die nachträglich durchzuführenden Berechnungen. Optimalerweise wird zum Rechenzeitende einer jeden Sequenz ein Restart-File erzeugt, das den aktuellen Prozesszustand zu diesem Zeitpunkt speichert. Um die durch $p < p_{cut\text{-}off}$ abgebrochenen Sequenzen nachträglich berechnen zu können, sollte zusätzlich gewährlistet sein, dass zu jedem Verzweigungspunkt, in dem alternative Sequenzen mit bestimmten Wahrscheinlichkeiten abzweigen, ebenfalls ein Restart-File erzeugt wird. Dies ist z. B. bei einer Kopplung mit dem ATHLET-Rechencode gegeben. Dies muss aber nicht bei jedem Rechencode der Fall sein. Dieses Restart-File wird in eine neu gestartete Simulation eingeladen und ggf. wird das Zeitende der Simulation angepasst. Zusammen mit den angepassten probabilistischen Daten, wie die hier angenommene Cut-off-Wahrscheinlichkeit, kann daraus im Scheduler eine Task erzeugt werden, welche der Taskliste hinzugefügt und daraufhin weiter abgearbeitet wird. Dies ist bereits jetzt für MCDET möglich, erfordert aber, dass dies für jede ausgewählte Sequenz einzeln durchgeführt wird.

Durch zukünftige Entwicklungsarbeiten könnten die identifizierten Sequenzen dem Scheduler beim Start als Argumente übergeben werden. So würde die automatisierte nachträgliche Verlängerung der Rechenzeit als alternative Betriebsart des Schedulers implementiert werden.

2.1.2 Erweiterung eines Dynamischen Ereignisbaumes um Sub-DETs

Eine nachträgliche Erweiterung eines bestehenden DETs durch Sub-DETs ist sinnvoll, wenn für eine bereits durchgeführte Analyse der Einfluss zusätzlicher stochastischer Einflussgrößen (z. B. der Ausfall weiterer Komponenten bei Anforderung oder während der Laufzeit) untersucht werden soll und eine neue MCDET-Analyse mit dem erweiterten

vollständigen Satz von stochastischen Größen zu zeitaufwändig ist, weil das mit MCDET gekoppelte Rechenprogramm sehr rechenzeitintensiv ist.

Für eine MCDET-Analyse werden normalerweise zunächst die Eingabedaten in der zugehörigen Eingabedatei spezifiziert und dann die Simulationsläufe basierend auf dieser Eingabedatei (und eventuell einer weiteren Eingabedatei für den Rechencode) gestartet. Zu den wesentlichen Eingabedaten von MCDET gehören die Trigger (Systembedingungen) und die von ihnen auszulösenden Aktionen während eines MCDET/Rechencode-Laufs. Die wichtigsten Aktionen sind das zufällige Ausspielen von Werten für stetige stochastische Einflussgrößen (z. B. Ausfallzeitpunkte von Komponenten) und die durchzuführenden Zustandsänderungen von diskreten stochastischen Einflussgrößen (z. B. Ausfall von Systemen und Komponenten).

Jede der zufällig ausgespielten Werte für die stetigen stochastischen Einflussgrößen wird für die Konstruktion eines DETs herangezogen. D. h. es wird eine Stichprobe von DETs generiert, wobei jeder DET durch eine Wertekombination für die stetigen stochastischen Einflussgrößen gekennzeichnet ist. Die Zustandsänderungen von diskreten stochastischen Einflussgrößen werden in Form von Verzweigungen innerhalb eines jeden DETs realisiert. D. h. die Zustandsänderungen selbst werden in neuen Simulationspfaden (Kind-Pfade) realisiert, die vom aktuellen Simulationspfad (Eltern-Pfad) abzweigen, während der Zustand im aktuellen Pfad unverändert bleibt. Gleichzeitig werden alle Pfade (Elternpfad und Kind-Pfade) am Verzweigungspunkt mit ihren jeweiligen Eintrittswahrscheinlichkeiten bewertet. Diese Eintrittswahrscheinlichkeiten müssen zusammen mit den Zustandsänderungen in der MCDET-Eingabedatei spezifiziert werden.

Die Zeitpunkte der Verzweigungen ergeben sich durch die zugehörigen Trigger. Sie sind entweder deterministisch oder zufällig. Ein deterministischer Verzweigungszeitpunkt liegt vor, wenn sich der zugehörige Trigger z. B. explizit auf eine bestimmte Systemzeit (während der Simulation berechnete Zeit, z. B. `time` = 1000 s) oder auf eine bestimmte initiierte Änderung des Systemzustands bezieht (z. B. eine bestimmte Komponente wird von 0 auf 1 gesetzt). Ein zufälliger Verzweigungszeitpunkt ergibt sich, wenn der Trigger z. B. durch das Übereinstimmen der Systemzeit mit einem zufällig ausgespielten Wert für den Ausfallzeitpunkt einer Komponente gekennzeichnet ist (z. B. `time` = `tfail`, wobei `tfail` die zufällig ausgespielten Ausfallzeitpunkte speichert).

Ein Sub-DET wird durch eine Verzweigung zu einem deterministischen oder zufälligen Zeitpunkt initiiert. Der kleinste Sub-DET ist ein einziger Simulationspfad, der von einem

anderen Simulationspfad abzweigt, aber selbst keine weiteren Abzweigungen bis zu seinem Ende mehr hat. Die Komplexität eines Sub-DETs hängt von den mit Verzweigungen verbundenen Triggern ab, die in einem abgezweigten Simulationspfad noch auftreten können (siehe Abb. 2.2).

Haben sich bei der Auswertung einer bereits durchgeführten MCDET-Analyse zusätzliche Fragestellungen bzgl. des Einflusses weiterer stochastischer Einflussgrößen ergeben (z. B. Ausfall einer zusätzlichen Komponente, die in der bereits durchgeführten Analyse wie vorgesehen funktionierte), kann eine ergänzende Analyse auf der Basis der Ergebnisse in der Ausgabedatei (HDF5) der bereits durchgeführten MCDET-Analyse stattfinden. Dafür müssen zunächst die Eingabedaten für die Erstellung des Sub-DETs in einer neuen Eingabedatei spezifiziert werden. Dazu zählen die Trigger und die dazugehörigen Aktionen, die für die Erstellung des Sub-DETs erforderlich sind. Als zusätzliche Eingabe muss die HDF5-Ausgabedatei angegeben werden und der Trigger, der die Erstellung des Sub-DETs auslöst. Wenn die Eingabedaten vollständig sind, kann MCDET gestartet werden.

In einem Vorverarbeitungsschritt zum eigentlichen Rechenlauf bestimmt MCDET anhand der Zustandstabellen (state tables) der HDF5-Ausgabedatei, in welchen bereits gerechneten Simulationspfaden der Trigger für den Sub-DET eintritt. Die Zustandstabellen sind für jeden gerechneten Simulationspfad vorhanden und dokumentieren die Entwicklung seines Prozess- und Systemzustands. Ist der Zeitpunkt des Triggers nicht explizit vorgegeben (z. B. Trigger = Komponente wird von 0 auf 1 gesetzt), wird dieser anhand der Zustandstabellen bestimmt. Wenn der Zeitpunkt des Triggers zufällig ist und der Trigger durch einen bestimmten Systemzustand gekennzeichnet ist (z. B. `time = tfail`, `K = 1`, wobei `tfail` die zufällig ausgespielten Ausfallzeitpunkte der erfolgreich gestarteten Komponente K speichert mit `K = 1`), dann wird zunächst pro DET der alten MCDET-Analyse ein Wert für den Zeitpunkt zufällig ausgespielt. Anschließend wird anhand der Zustandstabellen bestimmt, in welchen gerechneten Simulationspfaden der durch den Trigger beschriebene Systemzustand (`K = 1`) zum zufällig ausgespielten Trigger-Zeitpunkt realisiert ist.

Wenn die Simulationspfade und die zugehörigen Eintrittszeitpunkte des Sub-DET-Triggers vorliegen, werden im nächsten Schritt die letzten Verzweigungen vor Eintritt des Sub-DET-Triggers bestimmt. Dies erfolgt anhand der zu den jeweiligen Simulationspfaden gehörenden Ereignistabellen (event tables) der HDF5-Ausgabedatei. Da während eines MCDET/ATHLET-Simulationslaufs bei jeder Verzweigung eine Restart-Datei

angelegt wird, erhält man mit den letzten Verzweigungen vor Eintritt des Triggers die spätesten Zeitpunkte, an denen die neue MCDET-Analyse auf den Ergebnissen der alten Analyse aufsetzen muss, damit die Erstellung des Sub-DETs in den betroffenen Simulationspfaden ausgelöst wird. Zusätzlich werden aus den Ereignistabellen die Wahrscheinlichkeiten der Simulationspfade bis einschließlich des letzten Verzweigungspunkts ermittelt.

Die Informationen über die Simulationspfade, für die der Sub-DET nachträglich erstellt werden soll, und ihre letzten Verzweigungen vor Eintritt des Sub-DET-Triggers werden im MCDET-Scheduler verarbeitet. Dieser organisiert den Ablauf der neuen MCDET/Rechencode-Läufe zur Generierung des Sub-DETs für jeden im Preprocessingschritt bestimmten Simulationspfad der alten MCDET-Analyse. Der Scheduler geht hierbei nach dem im Abschnitt 2.1.1 beschriebenen Verfahren vor und nutzt für die neu zu erzeugenden Simulationspfade die nächstliegenden Restart-Files, um die Simulationsläufe zu starten. In den jeweils dafür erzeugten Tasks werden auch hier sowohl die Simulationszustände als auch die probabilistischen Daten nach den Vorgaben des Estimators modifiziert, bevor die Tasks in der Task-Liste des Schedulers abgearbeitet werden können.

Die für jeden Simulationspfad bestimmte Wahrscheinlichkeit bis einschließlich des letzten Verzweigungszeitpunkts wird in der zugehörigen Datamap für die Erstellung des Sub-DETs abgelegt. Basierend auf dieser Pfadwahrscheinlichkeit wird in Abhängigkeit von den weiteren Verzweigungen die Wahrscheinlichkeiten der Simulationspfade des Sub-DETs ermittelt.

Die Pfad-Wahrscheinlichkeiten der bereits gerechneten Simulationspfade werden in einem Post-Processing-Schritt korrigiert. Dabei werden die Wahrscheinlichkeiten aller Simulationspfade, von denen der Sub-DET abzweigt (Elternpfade des Sub-DETS), nachträglich mit der Komplementärwahrscheinlichkeit (1 - p) des Sub-DETs multipliziert. Dieselbe Korrektur erfolgt auch für die Wahrscheinlichkeiten sämtlicher Simulationspfade, die nach der Abzweigung des Sub-DETs von den Elternpfaden des Sub-DETs abzweigen (siehe Abb. 2.2).

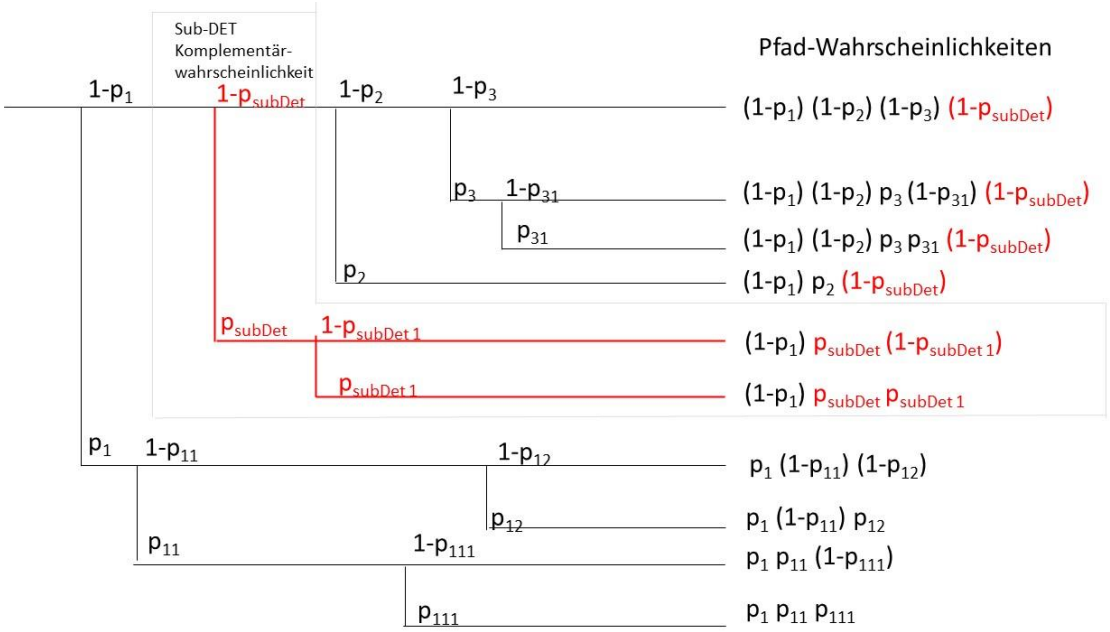


Abb. 2.2 Erweiterung eines bestehenden DETs (schwarz) durch einen Sub-DET (rot) einschließlich der Pfadwahrscheinlichkeiten des Sub-DETs und der korrigierten Pfad-Wahrscheinlichkeiten des bestehenden DETs

2.1.3 Re-Evaluierung eines bestehenden DETs durch Berücksichtigung alternativer Pfad-Wahrscheinlichkeiten aufgrund epistemischer Unsicherheiten

In einer MCDET-Analyse kann der Einfluss sowohl epistemischer als auch aleatorischer Unsicherheiten auf ein Simulationsergebnis untersucht werden. Die Werte der epistemischen Variablen werden genau wie die der stetigen aleatorischen Variablen aus den zugrundeliegenden Verteilungen ausgespielt und in den Simulationspfaden eines DETs berücksichtigt. Die verschiedenen Simulationspfade eines DETs ergeben sich aus der Berücksichtigung aller alternativen Werte der diskreten aleatorischen Variablen.

Die Werte der epistemischen und stetigen aleatorischen Variablen werden in zwei verschachtelten Schleifen berücksichtigt. Die äußere Schleife ist für die epistemischen und die innere Schleife für die stetigen aleatorischen Variablen zuständig.

Der Ablauf kann wie folgt skizziert werden:

```
Schleife bzgl. epistemischer Variablen, i = 1, …, n_epruns
  Ausspielen der Wertekombination ep(i, :) = (ep(i, 1), …, ep(i, n_ep))
    Schleife bzgl. stetiger aleatorischer Variablen, j = 1, …, n_alruns
     Ausspielen der Wertekombination al(j, :) = (al(j, 1), …, al(j, nal))
        Generieren und probabilistische Bewertung der Simulationspfade eines
        DETs unter Berücksichtigung von ep(i, :) und al(j, :)
    Ende Schleife für stetige aleatorische Variablen
Ende Schleife für epistemische Variablen
```

Die verwendeten Abkürzungen haben folgende Bedeutungen:

n_{ep}: Anzahl der epistemischen Variablen

n_{epruns}: Anzahl der Wertekombinationen für die epistemischen Variablen

$ep(i, :) = (ep(i, 1), \ldots, ep(i, n_{ep}))$:
Wertekombination Nr. i für die epistemischen Variablen $1, \ldots, n_{ep}$

n_{al}: Anzahl der stetigen aleatorischen Variablen

n_{alruns}: Anzahl der Wertekombinationen für die stetigen aleatorischen Variablen

$al(j, :) = (al(j, 1), \ldots, al(j, n_{al}))$:
Wertekombination Nr. j für die stetigen aleatorischen Variablen $1, \ldots, n_{al}$

Sollten sich die epistemischen Unsicherheiten ausschließlich auf Wahrscheinlichkeiten von Komponenten- und Systemzuständen zu deterministischen Zeitpunkten beziehen (z. B. Ausfallwahrscheinlichkeiten von Komponenten bei Anforderung oder zu vorgegebenen festen Zeitpunkten, Wahrscheinlichkeiten für Schädigungsgrade von Komponenten), wirken sich diese ausschließlich auf die probabilistische Bewertung der Simulationspfade eines DETs aus. An den Simulationspfaden selbst ändert sich nichts. Unterschiedliche Simulationspfade erhält man erst dann, wenn sich die epistemischen Unsicherheiten z. B. auch auf Modellparameter oder Prozessgrößen beziehen.

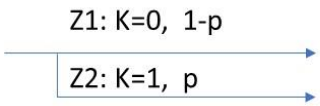
Für den Fall, dass sich die epistemischen Unsicherheiten ausschließlich auf Zustandswahrscheinlichkeiten zu deterministischen Zeitpunkten beziehen, kann die äußere Schleife für die epistemischen Variablen auf nur einen Durchgang begrenzt werden. D. h. es wird nur eine ausgespielte Wertekombination (z. B. die erste) für die epistemischen Variablen betrachtet und in der inneren Schleife zusammen mit jeder ausge-

spielten Wertekombination für die stetigen aleatorischen Variablen bei der Generierung und probabilistischen Bewertung der Simulationspfade eines DETs berücksichtigt. Im Post-Processing werden dann die probabilistischen Bewertungen der Simulationspfade der generierten DETs in Abhängigkeit von den anderen ausgespielten Wertekombinationen für die epistemischen Variablen angepasst. Damit dieser Schritt durchgeführt werden kann, müssen für die Verzweigungswahrscheinlichkeiten eines DETs die Formeln für ihre Berechnung bekannt sein. Diese Formeln werden deshalb während der Durchführung von MCDET-Simulationen in einer HDF5-Datei abgespeichert. Im Post-Processing werden die Formeln dann verwendet, um die Verzweigungswahrscheinlichkeiten in Abhängigkeit von den ausgespielten Wertekombinationen für die epistemischen Variablen neu zu berechnen. Für die Wahrscheinlichkeiten der vollständigen Simulationspfade eines DETs werden schließlich die Produkte aus den Wahrscheinlichkeiten beteiligter Simulationszweige ermittelt.

Im Folgenden werden beispielhafte Formeln zur Berechnung von Verzweigungswahrscheinlichkeiten mit epistemischen Unsicherheiten angegeben. Dabei wird angenommen, dass bei den MCDET-Simulationen die erste ausgespielte Wertekombination `(ep(1, 1), …, ep(1, n_ep))` für die epistemischen Größen berücksichtigt wurde. In den Formeln bezeichnet `ep(i, j)` den `i`-ten ausgespielten Wert der epistemischen Variablen `j`, und `Rep(i, j)` steht für den Quotienten `ep(i, j)/ep(1, j)`.

Beispiel 1:

Bei der Verzweigung in Beispiel 1 wird der Zustand von Komponente K geändert. `K = 0` mit Wahrscheinlichkeit `(1-p)` und `K = 1` mit Wahrscheinlichkeit `p`.

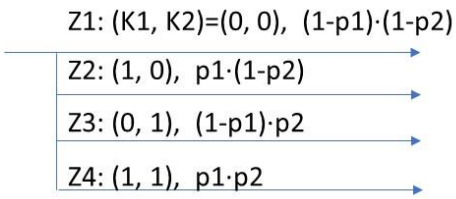


Wenn `p = ep(1, 1) = 0.0097`, dann lauten die Formeln für die Zweige `Z1` und `Z2`:

```
Z1: 0.9903 * (1. – ep(i, 1)) / (1. – ep(1, 1))
Z2: 0.0097 * Rep(i, 1)
```

Beispiel 2:

Bei der Verzweigung in Beispiel 2 werden gleichzeitig die Komponenten `K1` und `K2` geändert. `K1 = 0` mit Wahrscheinlichkeit `(1-p1)`, `K1 = 1` mit Wahrscheinlichkeit `p1`. `K2 = 0` mit Wahrscheinlichkeit `(1-p2)`, `K2 = 1` mit Wahrscheinlichkeit `p2`.

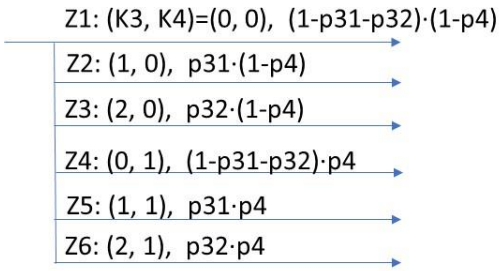


Wenn `p1 = 0.012` (fester Wert) und `p2 = ep(1, 1) = 0.0097`, dann lauten die Formeln für die vier Zweige `Z1, …, Z4`:

```
Z1: 0.9784164 * (1. – ep(i, 1)) / (1. – ep(1, 1))
Z2: 0.0118836 * (1. – ep(i, 1)) / (1. – ep(1, 1))
Z3: 0.0095836 * (1. – ep(i, 1)) / (1. – ep(1, 1))
Z4: 0.0001164 * Rep(i, 1)
```

Beispiel 3:

Bei der Verzweigung in Beispiel 3 werden gleichzeitig die Zustände der Komponenten `K3` und `K4` geändert. Für Komponente `K3` gibt es drei und für Komponente `K4` zwei mögliche Zustände. `K3 = 0` mit Wahrscheinlichkeit `(1-p31-p32)`, `K3 = 1` mit Wahrscheinlichkeit `p31` und `K3 = 2` mit Wahrscheinlichkeit `p32`. `K4 = 0` mit Wahrscheinlichkeit `(1-p4)`, `K4 = 1` mit Wahrscheinlichkeit `p4`.



Wenn `p31 = ep(1, 2) = 0.00261`, `p32 = ep(1, 3) = 0.00766` und `p4 = ep(1, 4) = 0.0012`, dann lauten die Formeln für die sechs Zweige `Z1, …, Z6`:

```
Z1: 0.98973 * (1. – ep(i, 2) – ep(i, 3)) / (1. – ep(1, 2) – ep(1, 3))
   * 0.99880 * (1. – ep(i, 4) / (1. – ep(1, 4)
Z2: 0.00261 * Rep(i, 2) * 0.99880 * (1. – ep(i, 4) / (1. – ep(1, 4)
Z3: 0.00766 * Rep(i, 3) * 0.99880 * (1. – ep(i, 4) / (1. – ep(1, 4)
Z4: 0.98973 * (1. – ep(i, 2) – ep(i, 3)) / (1. – ep(1, 2) – ep(1, 3))
   * 0.00120 * Rep(i, 4)
Z5: 0.00261 * Rep(i, 2) * 0.00120 * Rep(i, 4)
Z6: 0.00766 * Rep(i, 2) * 0.00120 * Rep(i, 4)
```

Wenn nur eine Wertekombination für die epistemischen Größen betrachtet wird, stellt sich folgendes Problem: Bei der Generierung eines DETs werden diejenigen Simulationspfade vorzeitig beendet, deren Wahrscheinlichkeit zu klein ist und unter eine vorgegebene sogenannte Cut-off-Wahrscheinlichkeit fällt. Wenn nun die ausgewählte Wertekombination für die epistemischen Größen viele vergleichsweise geringe Werte für die Zuverlässigkeitskenngrößen enthält und dadurch die Pfadwahrscheinlichkeiten geringer ausfallen, werden hier aufgrund der Cut-off-Wahrscheinlichkeit eventuell Simulationspfade nicht gerechnet, die mit anderen Wertekombinationen für die epistemischen Größen sehr wohl noch gerechnet werden würden. Eine mögliche Lösung für dieses Problem ist die Auswahl einer Wertekombination mit vergleichsweise hohen Werten aus der vorliegenden Gesamtstichprobe von Wertekombinationen für die epistemischen Variablen. Wenn die Werte von einigen epistemischen Variablen um Größenordnungen geringer ausfallen als die der anderen epistemischen Variablen sollten zusätzlich die höheren Werte dieser Variablen höher gewichtet werden.

2.2 Betriebsart zur Behandlung von klassischen Monte-Carlo-Simulationen

Mit MCDET können u. a. auch klassische Monte-Carlo-Simulationen durchgeführt werden. D. h. einmal gestartete Simulationsläufe werden bis zum vorgegebenen Ende gerechnet, ohne dass weitere Simulationspfade mit alternativen Systemzuständen von ihnen abzweigen. Ein Ereignisbaum besteht in diesem Fall also nur aus einem einzigen Simulationslauf. Für die Monte-Carlo-Simulation müssen gleich zu Beginn einer Simulation alle `nrun`-Wertekombinationen der unsicheren Parameter mit MCDET ausgespielt bzw. aus vorliegenden Dateien eingelesen werden. Anschließend müssen die Werte jeder Kombination den jeweiligen Größen des Rechencodes zugeordnet und der entsprechende Simulationslauf gestartet werden.

Die Spezifikation für das Ausspielen von Werten erfolgt in sog. Sampling-Fenstern. Diese sind durch einen Trigger (Zeit und/oder Systemzustand eines Simulationslaufs) für das Auslösen einer Zufallsauswahl sowie durch die Angabe der betroffenen Variablen und der zugehörigen Verteilungen, aus denen ausgespielt werden soll, gekennzeichnet. Der Trigger für den Anfangszustand des ersten Simulationslaufs kann ganz einfach mit dem Schlüsselwort `init` angegeben werden.

Die Zuordnung von neuen Werten (Zuständen) zu entsprechenden Größen des Rechencodes wird in sog. Transition-Fenstern festgelegt. Diese Fenster sind ebenfalls durch

einen Trigger für das Initiieren einer Zustandsänderung (Transition) gekennzeichnet. Zusätzlich werden die betroffenen Größen des Rechencodes, ihre neuen Werte (Zustände) sowie die Wahrscheinlichkeiten für die neuen Werte angegeben. Für die klassische Monte-Carlo-Simulation wird der Trigger genau wie beim Sampling-Fenster mit `init` angegeben. Als neuer Wert einer Rechencode-Größe wird der Name der zugeordneten Variablen angegeben, die die möglichen Werte der Größe bereitstellt. Die Wahrscheinlichkeit für den neuen Wert beträgt 1.

Der `init`-Trigger in einem Fenster bewirkt, dass gleich zu Beginn des ersten Simulationslaufs einer MCDET-Analyse die mit ihm verbundenen Aktionen (Zufallsauswahl von Werten oder Zustandsänderungen von Rechencode-Größen) durchgeführt werden. Sollen die Zustände von Rechencode-Größen geändert werden, wird zunächst überprüft, ob die neuen Werte der betreffenden Rechencode-Größe mit Variablen übereinstimmen, für die vorher `n` Werte zufällig ausgespielt bzw. aus einer vorliegenden Datei eingelesen wurden. In diesem Fall werden gleich zu Beginn einer MCDET-Simulation `nrun` neue Simulationsläufe angelegt. Jeder dieser Simulationsläufe berücksichtigt eine ausgespielte Wertekombination für die betroffenen Rechencode-Größen und hat eine bedingte Wahrscheinlichkeit von 1 (Bedingung ist das Eintreten der Wertekombination für die betroffenen Rechencode-Größen, mit der der Simulationspfad gestartet wurde). Der Simulationslauf mit den ursprünglichen Werten (z. B. Best-Estimate-Werte, konservative Werte) der Rechencode-Größen wird nicht fortgesetzt, weil seine bedingte Wahrscheinlichkeit 0 beträgt.

Nachdem die Simulationsläufe einer klassischen Monte-Carlo-Simulation gestartet wurden, ist keine weitere Aktion (Zufallsauswahl von Werten oder Zustandsänderungen von Rechencode-Größen) auf Seiten von MCDET mehr durchzuführen. Die Simulationsläufe können ohne Unterbrechung bis zum vorgegebenen Ende durchlaufen. Eine Evaluierung des Simulationszustand zu jedem Zeitpunkt - wie sonst üblich bei einer MCDET-Simulation - ist nicht erforderlich. Sie kann deshalb für den weiteren Verlauf der Simulation abgeschaltet werden. Dafür wird gleich zu Beginn überprüft, ob im Eingabedatensatz von MCDET ausschließlich Fenster mit einem `init`-Trigger vorkommen. Wenn ja, wird die Evaluierung deaktiviert, andernfalls wird der Simulationszustand auch weiterhin zu jedem Zeitpunkt bewertet. Durch diese Erweiterung können also zahlreiche Bewertungsschritte eingespart werden, was sich, besonders bei vielen parallel zu bewertenden Simulationen, günstig auf die Laufzeit des Schedulers auswirkt. Auch die Abschaltung des getakteten Simulationslaufs wäre dadurch optional möglich, was die Läufe nochmal deutlich beschleunigen würde. Diese Option (`direct_run`) ist aber nur in

Ausnahmefällen relevant, da hierdurch auch das zeitschrittgenaue Schreiben des Simulationszustands in die MCDET-Ausgabedatei (HDF5) nicht mehr möglich ist und für den weiteren Simulationsverlauf nur noch die Ausgabedateien des Rechenprogramms zur Verfügung stehen.

2.3 Reproduzierbarkeit einer MCDET-Analyse durch konsistente Berücksichtigung pfad-spezifischer Zufallszahlen

Mit dem in RS 1198 und RS 1529 neu entwickelten Scheduling-System ist die Reproduzierbarkeit einer MCDET-Analyse nicht immer gewährleistet. Bei diesem Scheduling-System werden alternative Simulationspfade parallel berechnet und diesen je nach Abarbeitung unterschiedliche Zahlen des Pseudozufallszahlengenerators (Pseudo Random Number Generator (PRNG)) zugeordnet. Dadurch kann bei einer wiederholten MCDET-Analyse eine andere Zuordnung von Simulationspfaden zu den generierten PRNG-Zahlen stattfinden, was im Folgenden zu veränderten Abläufen und damit zu unterschiedlichen MCDET-Ergebnissen führen kann. Das war mit dem früheren Scheduling-System, bei dem eine sequenzielle Abarbeitung alternativer Simulationspfade nach dem Prinzip last-in-first-out erfolgte und das deshalb relativ ineffizient war, nicht möglich, weshalb die Problematik der Zuordnung von Zufallszahlen hier nicht extra behandelt werden musste.

Um auch für das neue Scheduling-System die Zuordnung eines Simulationspfades zu der vom PRNG gelieferten Zahlensequenz eindeutig und damit eine MCDET-Analyse reproduzierbar zu machen, wurde folgender Ablauf im Fortran-Kern von MCDET umgesetzt:

- Für den ersten Simulationspfad wird ein Seed-Wert für den PRNG im Eingabedatensatz vorgegeben. Dieser wird in der Datamap des ersten Simulationspfades in der Größe Seed abgespeichert (z. B. Seed = 123457). Zusätzlich wird in der Datamap die Zahl der neuen Simulationszweige abgespeichert, die bis einschließlich des aktuellen Simulationszeitpunkts vom aktuellen Simulationspfad abgezweigt sind (SeedAdder). Ihr Anfangswert beträgt 0 (d. h. SeedAdder = 0).
- Wenn im aktuellen Simulationspfad der SeedAdder-Wert hochgezählt wird, weil neue Simulationszweige generiert werden, wird die Größe SeedAdder in der zugehörigen Datamap entsprechend aktualisiert.

- Wenn sich im aktuellen Simulationspfad der PRNG-Zustand ändert, weil ein zufälliger Wert aus einer Wahrscheinlichkeitsverteilung ausgespielt wird, wird die Seed-Größe in der zugehörigen Datamap auf den aktuellen PRNG-Zustand gesetzt.

- Für jeden neuen Simulationszweig wird in Abhängigkeit vom aktuellen Seed- und SeedAdder-Wert im aktuellen Simulationspfad, von dem abgezweigt wird, ein neuer Seed- und SeedAdder-Anfangswert festgelegt und in der Datamap für den neuen Simulationszweig abgespeichert. Der Seed-Anfangswert für den neuen Simulationszweig ist die Summe aus dem aktuellen Seed- und SeedAdder-Wert (Seed = Seed + SeedAdder). Der SeedAdder-Anfangswert für den neuen Simulationszweig ist das Produkt aus dem aktuellen SeedAdder-Wert und dem im Fortran-Kern vorgegebenen SeedFactor-Parameter (SeedAdder = SeedAdder * SeedFactor). SeedFactor entspricht der angenommenen maximalen Zahl von Pfaden, die von einem anderen Simulationspfad abzweigen können, und ist aktuell gleich 50 gesetzt.

2.3.1 Reproduzierbarkeit nach vollständiger Umstellung des MCDET-Kerns auf Python

Es ist mittelfristig vorgesehen, dass der MCDET-Kern vollständig auf Python umgestellt wird. Dann kann die Reproduzierbarkeit einer MCDET-Analyse über frei verfügbare Werkzeuge in den Python-Bibliotheken numpy /HAR 20/ und scipy gewährleistet werden.

Als PRNG empfiehlt sich aktuell der Bitgenerator PCG-64 von numpy (Permuted Congruential Generator (64-bit, PCG64) — NumPy v1.22 Manual). Wie alle Bitgeneratoren generiert er eindeutige und reproduzierbare Zufallszahlen. Wenn ein Anfangswert (Integer) vorgegeben wird (z. B. Seed = 123457), dann wird dieser zunächst in einen hochwertigen internen Anfangszustand des Bitgenerators umgewandelt. Diese Aufgabe wird bei jedem Bitgenerator von numpy.random.SeedSequence durchgeführt. Wenn kein Anfangswert vorgegeben wird, dann wird ein unvorhersehbarer Entropie-Wert abhängig vom Betriebssystem ausgewählt. In diesem Fall ist davon auszugehen, dass die Ergebnisse nicht reproduzierbar sind.

Die folgenden zwei möglichen Python-Anweisungen zur Instanziierung eines PCG-64 mit eindeutigen Zufallszahlen sind aktuell:

- from numpy.random import default_rng
  prng = default_rng (Seed)

- from numpy.random import Generator, PCG64
  `prng` = Generator (PCG64 (Seed))

Die Anweisung zur Generierung von n zufälligen Werten aus einer Standardnormalverteilung bei Anwendung von numpy.random lautet (`prng` steht dabei für den vorher definierten PRNG)

- `prng`.standard_normal(n).

Bei Anwendung von scipy.stats lautet die Anweisung

- norm.rvs(size = n, random_state = `prng`).

Die Reproduzierbarkeit einer MCDET-Analyse kann wie folgt gewährleistet werden:

- Für den ersten Simulationspfad wird ein Seed-Wert im Eingabedatensatz vorgegeben (z. B. Seed = 123457). Dieser wird verwendet, um einen PRNG mit eindeutigen Zufallszahlen für den ersten Simulationspfad zu instanziieren (z. B. `prng` = default_rng(Seed)). Die Größe `prng` mit dem Verweis auf die PRNG-Instanz wird in der Datamap des ersten Simulationspfads abgespeichert.
- Wenn im aktuellen Simulationspfad k neue Simulationszweige (childs) generiert werden, werden k neue Anfangszustände für die Instanziierung der PRNGS in den Simulationszweigen wie folgt generiert:
  - seq = `prng`.bit_generator._seed_seq
    child_seeds = seq.spawn(k)

  Die unterschiedlichen PRNGs der neuen Simulationszweige werden wie folgt instanziiert:

  - `prngs` = [default_rng(s) for s in child_seeds]

Die neuen PRNG-Instanzen für die Simulationszweige sind unabhängig und sehr wahrscheinlich nicht überlappend. Jede Größe `prngs[i]` aus der Liste `prngs` verweist auf eine neue PRNG-Instanz und wird in der zugehörigen Datamap des neuen Simulationszweigs `i, i = 1, …,` k, abgespeichert.

2.4 Methode zur quantitativen Bewertung von Einflussgrößen auf Ergebniscluster (ASPIC)

2.4.1 Motivation und Zielsetzung

Bisher durchgeführte MCDET-Analysen haben gezeigt, dass die in einer MCDET-Analyse berechnete Vielzahl unterschiedlicher Sequenzen oftmals in Gruppen (Cluster) eingeteilt werden können, wobei bestimmte Eigenschaften von Prozessgrößen innerhalb der Sequenzen eines Clusters relativ ähnlich und zwischen den Clustern eher unterschiedlich sind. Die Fragestellung, die durch die Methode zur quantitativen Bewertung von Einflussgrößen auf Ergebniscluster zu beantworten ist, soll am nachfolgenden Beispiel verdeutlicht werden.

Eine Veranschaulichung einer Clusterbildung von Ergebnissen liefern z. B. die Temperaturverläufe des Kabelinnenmantels in Abb. 2.3. Diese Temperaturverläufe wurden im Rahmen einer MCDET Analyse eines Brandszenarios unter Berücksichtigung von Brandbekämpfungsmaßnahmen /PES 14/ erzeugt.

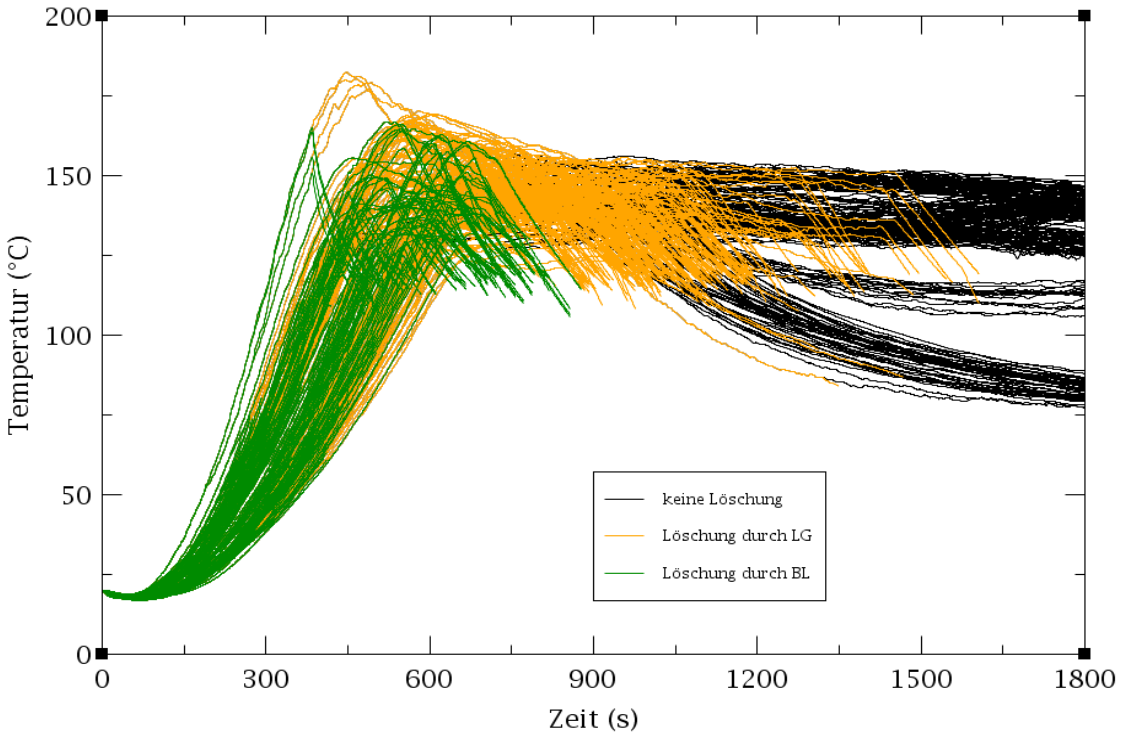


Abb. 2.3 Temperaturverlauf der Kabelmantelinnenwand in Abhängigkeit aleatorischer und epistemischer Unsicherheiten

Bei den Temperaturverläufen in Abb. 2.3 lassen sich verschiedene Sequenzcluster erkennen. Zwei deutlich voneinander unterscheidbare Sequenzcluster bilden die grün und

orange gekennzeichneten Temperaturverlaufe. Diese farblich gekennzeichneten Sequenzen haben alle die Eigenschaft, dass sie während der Rechenzeit zwischen 0 und 1800 s abbrechen. Der Abbruch einer Sequenz erfolgt zum Zeitpunkt, wann die Löschung des Brandes stattgefunden hat. In der Abbildung wird deutlich, dass die grünen Sequenzen in der Mehrzahl früher abbrechen als die orangenen Sequenzen. Der Hauptgrund, der zu den grünen und orangenen Sequenzclustern geführt hat, ist in der Legende der Abb. 2.3 ersichtlich. Bei den grünen Sequenzen wurde die Löschung durch den Brandläufer (BL), bei den orangenen Sequenzen durch die Löschgruppe (LG) bzw. Feuerwehr durchgeführt. Wenn die Situation gegeben ist, dass der BL die Löschung durchführen kann, erfolgt dies in der Regel schneller, als wenn die LG der Feuerwehr die Löschung vornimmt, da der BL den Brandraum eher erreicht als die LG der Feuerwehr.

Bei den schwarzen Sequenzen, die bis zum Rechenzeitende von 1800 s verlaufen, wurde keine Brandlöschung durchgeführt. Obwohl keine Brandlöschung durchgeführt wurde, kann man in Abb. 2.3 auch in diesen Fällen mindestens drei verschiedene Sequenzcluster erkennen. Während die Erklärung, was zu den grünen und orangenen Sequenzclustern geführt hat, relativ einfach angegeben werden konnte, ist die Erklärung, welche Ursache der Clusterbildung bei den schwarzen Sequenzen zugrunde liegt, nicht so offensichtlich.

Die dazu entwickelte Methodik zielt auf die Beantwortung folgender Fragen ab:

- Welche Parameter tragen am meisten zur Erklärung eines Ergebnisclusters bei, bzw. welche Parameter liefern die meiste Information zur Erklärung eines Clusters?
- Durch welche Werte bzw. Wertekombinationen der als relevant identifizierten Parameter können die Ergebniscluster möglichst gut erklärt werden bzw. welche Kombinationen von Parameterwerten können möglichst zuverlässig einem Cluster zugeordnet werden?
- Welche Fehlerwahrscheinlichkeiten sind mit der Zuordnung der Wertekombinationen auf die Ergebniscluster verbunden?

Die Fragestellungen der letzten beiden Punkte sind mit der einer Diskriminanzanalyse vergleichbar, in der verschiedene Gruppen auf Unterschiede ihrer Merkmale untersucht und die relevanten Merkmale zur Unterscheidung identifiziert werden, so dass eine Zuordnung auf die Cluster mit möglichst kleiner Fehlerwahrscheinlichkeit durchgeführt werden kann.

Entgegen der zur Verfügung stehenden Methoden der Diskriminanzanalyse, die oftmals an bestimmte Voraussetzungen gebunden ist (z. B. bestimmte Verteilungsannahmen), wurde hier ein Verfahren entwickelt, das allgemein für eine Analyse von MCDET-Ergebnissen angewendet werden kann.

Bei der entwickelten Methode geht es in erster Linie darum, die Ursachen zu ermitteln, die für die Clusterbildung von Ergebnissen verantwortlich sind und somit eine Erklärung zu finden, aufgrund welcher Einflüsse sich diese Cluster ergeben haben. Dies kann zu einem Kenntnisgewinn bzgl. der Zusammenhänge von Einflussparametern und Prozessentwicklungen beitragen. Es wird ausdrücklich darauf hingewiesen, dass die Methode nicht als Klassifikationsverfahren entwickelt wurde.

Als Ergebniscluster sollen nicht nur diejenigen betrachtet werden, die wie in Abb. 2.3 visuell durch die grafische Darstellung der Sequenzen leicht erkennbar sind. Es sollen auch solche Cluster betrachtet werden, die vom Anwender durch eine fachlich begründete Einteilung des Ergebnisbereichs erzeugt werden. Beispielsweise wenn der Anwender wissen möchte, welche Eingabekombination von Parameterwerten dazu führen, dass eine Ergebnisgröße y

- einen bestimmten Wert y_1 unterschreitet,
- einen bestimmten Wert y_2 überschreitet oder
- in einem bestimmten Intervall $y_1 \leq y \leq y_2$ liegt.

Um eventuelle Cluster in einem Datensatz zu erkennen, könnte auch das multivariate statistische Verfahren der Clusteranalyse eingesetzt werden.

In den Abschnitten 2.4.2 und 2.4.3 werden die grundlegenden Ideen und die Herleitung der Methode ASPIC zur quantitativen Bewertung von Einflussgrößen auf Ergebniscluster beschrieben. Die Methode basiert auf dem Konzept der Entropie und dem daraus abgeleiteten Informationsgewinn (Information-Gain) bzgl. der jeweiligen berücksichtigten Parameter.

Zur besseren Nachvollziehbarkeit wird die Methode ASPIC in Abschnitt 2.4.4 anhand eines einfachen Anwendungsbeispiels veranschaulicht.

2.4.2 Einleitende Überlegungen zur Verwendung des Entropie-Maßes

Wenn sich Sequenzen in einem Cluster mit hoher Wahrscheinlichkeit durch bestimmte Wertekombinationen von Einflussgrößen erklären lassen, dann kann davon ausgegangen werden, dass diese Wertekombinationen von Einflussgrößen auch zur Prognose bzw. zur Klassifikation von Ergebnissen verwendet werden können. Dies erinnert an das Klassifikationsverfahren eines Entscheidungsbaumes (Decision Tree), der durch die Anwendung bestimmter Algorithmen Regeln aufstellt, um Beobachtungen anhand von Merkmalsausprägungen (Parameterwerten) möglichst genau einer bestimmten Klasse zuordnen zu können.

Ein Konzept, das in einem Entscheidungsbaum zur Bildung von Entscheidungsregeln verwendet wird, beruht auf dem Entropie-Maß und dem Informationsgewinn (Information-Gain), der durch die Entscheidungsregel bzgl. eines Parameters erzielt wird. Die Entropie ist ein Maß, das die Ungleichheit, Unordnung oder Unsicherheit von Elementen in einem System angibt. In dem hier vorliegenden Problem bezieht sich die Entropie auf die ungleiche Verteilung von Werten eines Parameters auf die verschiedenen Cluster.

Die allgemeine Definition der Entropie ist durch Gleichung (1.1) gegeben:

$$E = -\sum_{i=1}^{h} p_i \cdot \log_h(p_i) \qquad (2.1)$$

Dabei bezeichnet h die Anzahl der Klassen (Cluster), p_i ist der Anteil (relative Häufigkeit) der Werte des betrachteten Parameters, die zur Klasse i gehören und $\log_h$ bezeichnet den Logarithmus zu Basis h. Durch die Abhängigkeit der Logarithmus-Basis von der Anzahl der Cluster wird gewährleitet, dass die maximale Entropie den Wert 1 erhält und somit eine einheitliche feste Bezugsgröße der maximalen Entropie gegeben ist.

Wie die Kenngrößen Entropie und Informationsgewinn in diesem Kontext zusammenhängen, kann folgendermaßen anhand der Wahrscheinlichkeitsverteilung einer Zufallsvariablen X verdeutlicht werden:

- Bei einer gleichverteilten Zufallsgröße zwischen a und b haben alle Werte zwischen a und b die gleiche Wahrscheinlichkeit, in einer Zufallsstichprobe ausgespielt zu werden. Teilt man den Wertebereich zwischen a und b in gleich große Intervalle ein, lässt sich kein Intervall angeben, aus dem ein Wert bei einer Zufallsziehung bevorzugt – d. h. mit höherer Wahrscheinlichkeit - gezogen wird. D. h., alle Intervalle, aus denen der Zufallswert gezogen werden könnte, haben die gleiche Wahrschein-

lichkeit. Damit ist die Unsicherheit bzw. Entropie bzgl. einer Vorhersage, aus welchem Intervall der Zufallswert gezogen wird, bei einer Gleichverteilung am größten. Dies kann auch so interpretiert werden, dass man bei einer Gleichverteilung keinen Informationsgewinn darüber erhält, welches Intervall bei einer Zufallsziehung bevorzugt ist.

- Sind die Werte einer Zufallsvariablen X über den Bereich a und b, der wie zuvor in gleich große Intervalle eigeteilt ist, gemäß einer Normalverteilung mit geringer Varianz verteilt, liefert uns die Verteilung von X wichtige Informationen darüber, aus welchem Intervall ein Zufallswert mit hoher Wahrscheinlichkeit gezogen wird. D. h., die Prognose, aus welchem Intervall der Zufallswert gezogen wird, ist mit weniger Unsicherheit bzw. kleinerer Entropie verbunden. In diesem Fall erhält man durch die Verteilung der Zufallsvariablen X einen Informationsgewinn, mit dem man mit höherer Wahrscheinlichkeit eine korrekte Prognose, aus welchem Intervall ein zufällig ausgespielter Wert stammt, abgeben kann.

Der Zusammenhang zwischen Entropie und Informationsgewinn besteht somit darin, dass, je kleiner (bzw. größer) die Entropie der Verteilung ist, desto größer (bzw. kleiner) ist der Informationsgehalt der Verteilung. Diese Beziehung zwischen Entropie und Informationsgewinn wird in der hier vorgeschlagenen Methode verwendet, um die Bedeutung der einzelnen Parameter für die jeweiligen Sequenz-Cluster quantifizieren zu können. Der Informationsgewinn (Information-Gain) gibt dabei an, wieviel Information durch die Verteilung eines Parameters gewonnen werden kann bzw. um wieviel die Unsicherheit bzgl. einer Prognose reduziert werden kann, zu welchem Cluster ein bestimmter Wert des Parameters zuzuordnen ist.

Um die Größen Entropie und Informationsgewinn für die entwickelte Methode verwenden zu können, müssen folgende Voraussetzungen gelten:

- Definition der interessierenden Cluster
- Für jeden Parameter ist eine Einteilung seines Wertebereichs in disjunkte Intervalle durchzuführen. Für diskrete Parameter sind die Klassen durch die diskreten Merkmalswerte definiert, die der Parameter annehmen kann.

Zunächst wird eine Zuordnung der Beobachtungen in die definierten Cluster durchgeführt. Dazu wird angenommen, dass eine Stichprobe von n Beobachtungen einer Funktion G gegeben ist. Eine Beobachtung besteht aus den Werten von m Einflussgrößen

$x = (x_1, \ldots, x_m)$ und dem zugehörigen Funktionswert $y = G(\mathbf{x})$. Die Verteilungsfunktionen $F_1, \ldots, F_m$ der m Parameter $X_1, \ldots, X_m$ werden als bekannt vorausgesetzt.

Um eine Klassifizierung von Beobachtungen auf die jeweiligen Cluster durchzuführen zu können, müssen zunächst die interessierenden Cluster für die Ergebnisgröße spezifiziert werden. Erfolgt dies durch den Nutzer, z. B. aufgrund von fachspezifischen Gegebenheiten, muss er die entsprechenden Werte angeben, die die Cluster voneinander trennen. Ist man an h Cluster interessiert, müssen zur Definition der Cluster die entsprechenden Intervallgrenzen $L_1, \ldots, L_{h-1}$ zwischen den Clustern angegeben werden. Mit der Angabe der Werte $L_1, \ldots, L_{h-1}$ sind die h Cluster definiert durch

Cluster 1 = $\{ y: y \leq L_1 \}$
Cluster 2 = $\{ y: L_1 < y \leq L_2 \}$
:
Cluster h-1 = $\{ y: L_{h-2} < y \leq L_{h-1} \}$
Cluster h = $\{ y: y > L_{h-1} \}$

y bezeichnet die Funktionswerte der Beobachtungen. Besteht nur Interesse an zwei Clustern, muss derjenige Wert L_1 angegeben werden, der die beiden Cluster trennt. Cluster 1 = $\{y: y \leq L_1\}$ beschreibt somit die Menge der Ergebniswerte, für die $y \leq L_1$ gilt.

Nach der Definition der Cluster können die Beobachtungen der Stichprobe genau einem Cluster zugeordnet werden. Falls der Nutzer keine fachlichen Gründe hat, um die Cluster zu definieren, und auch die graphische Darstellung der Sequenzen keine offensichtlichen Hinweise auf Clusterbildungen liefert, können zur Definition der Cluster auch die statistischen Verfahren der Clusteranalyse angewendet werden.

Wenn die Cluster definiert worden sind und die Beobachtungen einer Stichprobe vom Umfang n den Clustern zugeordnet wurden, ergibt sich eine bestimmte Verteilung der Beobachtungen der Stichprobe auf die Cluster. Sei $n_1, \ldots, n_h$ die Anzahl der Beobachtungen, die in den jeweiligen Clustern 1, …, h liegen, dann sind durch $\frac{n_i}{n}$ mit $\sum_{i=1}^{h} n_i = n$ die relativen Häufigkeiten gegeben, mit denen eine Beobachtung im Cluster i, i=1, …, h, liegt. Der Informationsgehalt der Verteilung der Beobachtungen auf die Cluster wird dadurch bestimmt, wie eindeutig sich die Ergebnisse den jeweiligen Clustern zuordnen lassen bzw. wie eindeutig sich die Beobachtungen über die Cluster verteilen.

Der geringste Informationsgehalt bzw. die maximale Entropie würde dann auftreten, wenn sich die Beobachtungen der Stichprobe mit gleicher Häufigkeit über alle Cluster verteilen würden, d. h. wenn sich $n_1 = n_2 = \ldots = n_h$ ergibt. Wie oben bereits diskutiert, liefert die Gleichverteilung die geringste Information bzgl. der Entscheidung, in welches Cluster eine neue Beobachtung, deren Ergebnis man noch nicht kennt, fallen wird. Im Fall einer Gleichverteilung besteht die Information darin, dass die Beobachtung mit der gleichen Wahrscheinlichkeit allen h Clustern zugeordnet werden kann. Die Wahrscheinlichkeit einer korrekten Prognose einer Beobachtung zu einem Cluster beträgt dann 1/h.

Der größte Informationsgehalt der Verteilung wäre dann gegeben, wenn alle Beobachtungen genau einem Cluster zugeordnet werden können. In diesem Fall wäre die Entropie minimal und man kann sich in Abhängigkeit von der Größe des Stichprobenumfanges mehr oder weniger sicher sein, dass eine künftige Beobachtung ebenfalls in das Cluster j fällt. Um die Wahrscheinlichkeit einer korrekten Zuordnung zu ermitteln, ist die Größe der Stichprobe und der damit zusammenhängende Stichprobenfehler zu berücksichtigen. Darauf wird im Abschnitt 2.4.3.2 im Zusammenhang mit der Berechnung von Fehlerwahrscheinlichkeiten für Zuordnungsregeln (Diskriminanzregeln) eingegangen.

Die Entropie der Verteilung der Beobachtungswerte eines Parameters auf die h Cluster wird durch die Gleichung (2.2) berechnet:

$$E_0 = - \sum_{i=1}^{h} \frac{n_i}{n} \cdot \log_h \left(\frac{n_i}{n} \right) \qquad (2.2)$$

Der Entropie-Wert E_0 beschreibt den Ausgangswert der Entropie, der sich durch die Verteilung der Beobachtungswerte auf die Cluster ergibt. Durch die Methode ASPIC sollen Diskriminanzregeln gefunden werden, mit denen der Ausgangswert E_0 der Entropie möglichst stark reduziert werden kann. Die Herleitung der Methode ASPIC erfolgt im nachfolgenden Abschnitt 2.4.3.

2.4.3 Herleitung der Methode ASPIC

Durch die Methode ASPIC (Quantitative Assessment of Parameter Influence on Value Clusters) sollen folgende Aufgaben bearbeitet werden:

i) Reduzierung der Entropie bzgl. der einzelnen Parameter (Einflussgrößen) und Quantifizierung des Informationsgewinns durch die Entropie-Reduktion.

ii) Identifizierung des Parameters, bei dem ein maximaler Informationsgewinn erzielt werden kann.

iii) Erzeugung einer univariaten Diskriminanzregel (Zuordnungsregel) für jeden Parameter, und Quantifikation der Fehlerwahrscheinlichkeit der Diskriminanzregel.

iv) Quantifizierung des Beitrags, den die jeweiligen Parameter zur Erklärung der Ergebniscluster liefern.

v) Erweiterung von univariaten auf multivariate Diskriminanzregeln und Quantifizierung des Beitrags, mit dem sich die Fehlerwahrscheinlichkeit einer Zuordnung zu den Ergebnisclustern verringert.

Zunächst wird in Abschnitt 2.4.3.1 das Verfahren beschrieben, mit der eine Entropie-Reduktion und der damit zusammenhängende Informationsgewinn gegenüber E_0 der Gleichung (1.2) bzgl. der einzelnen Parameter quantifiziert werden kann. Der Informationsgewinn der jeweiligen Parameter wird verwendet, um den Parameter zu ermitteln, mit dem ein maximaler Informationsgewinn für die Clusterbildung erzielt werden kann. In Abschnitt 2.4.3.2 werden für den Parameter mit dem maximalen Informationsgewinn Diskriminanzregeln (Zuordnungsregeln) und damit verbundene Fehlerwahrscheinlichkeiten ermittelt. In Abschnitt 2.4.3.3 wird beschrieben, wie in ASPIC der Beitrag von Parametern zur Erklärung der Cluster ermittelt wird. Der Abschnitt 2.4.3.4 behandelt die Erweiterung von univariaten auf multivariate Diskriminanzregeln.

2.4.3.1 Quantifizierung des Informationsgewinns durch Entropie-Reduktion

Ausgangspunkt sind die Beobachtungen einer Stichprobe. Wie oben bereits erwähnt, sind Beobachtungen einer Stichprobe gegeben durch einen Vektor $x = (x_1, \ldots, x_m)$ der Werte der m Einflussgrößen (Parameter) und dem zugehörigen Funktionswert $y = G(x)$. Die Funktion G kann dabei eine einfache Funktion darstellen oder ein komplexes Rechenmodell, mit dem die Funktionswerte berechnet werden.

Da die Quantifizierung des Informationsgewinns für jeden der m Parameter separat durchgeführt wird, erfolgt die Beschreibung des Verfahrens beispielhaft für einen beliebigen Parameter X. Für die restlichen Parameter wird das Verfahren analog angewendet.

Zunächst wird der Wertebereich des Parameters in disjunkte Intervalle bzw. Klassen eingeteilt:

- Handelt es sich um einen Parameter mit einem stetigen Definitionsbereich, so wird die Spannweite der Stichprobenwerte des jeweiligen Parameters in k (z. B. k = 5, 10, 15) Intervalle eingeteilt. Die Anzahl der Intervalle kann vom Nutzer optional gewählt werden. Je größer die Anzahl der Intervalle, desto genauer (zumindest bis zu einem Grad) kann die Quantifizierung des Informationsgewinns für den Parameter durchgeführt werden. Die Untersuchung, welchen Einfluss die Anzahl der Intervalle auf die Ergebnisse haben, wird hier nicht weiterverfolgt und müsste ggf. zu einem späteren Zeitpunkt durchgeführt werden.

  Die Einteilung des Wertebereichs in Intervalle erfolgt für einen Parameter X mit stetiger zugrundeliegender Verteilungsfunktion FX dermaßen, dass jedes Intervall des Parameterbereichs die gleiche Wahrscheinlichkeit aufweist.

  Wenn z. B. der Wertebereich eines Parameters in fünf Intervalle aufgeteilt wird, ergeben sich folgende Intervalle:

  $$I_1 = [\xi_0 , \xi_1],\ I_2 = [\xi_1 , \xi_2],\ I_3 = [\xi_2 , \xi_3],\ I_4 = [\xi_3 , \xi_4],\ I_5 = [\xi_4 , \xi_5] \tag{2.3}$$

  Die Wahrscheinlichkeit, dass eine Beobachtung x des Parameter X in dem Intervall I_j liegt, ist gegeben durch

  $$P(I_j) = P(\xi_{j-1} \leq x < \xi_j) = \int_{\xi_{j-1}}^{\xi_j} f(x)\, dx \text{ für } j = 1,\cdots, 5,$$

  wobei f(x) die Dichte der Verteilungsfunktion FX des Parameters X beschreibt.

  Zur Festlegung der Intervallgrenzen kann als einfaches Kriterium zugrunde gelegt werden, dass die Intervalle die gleiche Wahrscheinlichkeit aufweisen. Um zu gewährleisten, dass die Intervalle die gleiche Wahrscheinlichkeit P(I1) = P(I2) = … = P(I5) haben, sind für die Intervallgrenzen ξj die entsprechenden Quantile der Verteilungsfunktion FX in Abhängigkeit der Anzahl der Intervalle einzusetzen. Bei den in (1.3) gewählten fünf Intervallen wird für ξ1 das 20 %-Quantil, für ξ2 das 40 %-Quantil, für ξ3 das 60 %-Quantil und für ξ4 das 80 %-Quantil der Verteilung FX gewählt. Aus praktischen Gründen wird für ξ0 das 0.01 %-Quantil und für ξ5 das 99.99 %-Quantil der Verteilung eingesetzt, da bei stetigen Verteilungen von Parametern das Minimum oder das Maximum oftmals durch -∞ oder ∞ definiert ist. Im Folgenden werden die Quantile, die für die Intervallgrenzen der fünf Intervalle eingesetzt werden, mit ξ20 %, ξ40 %, ξ60 % und ξ80 % bezeichnet.

Die Intervallgrenzen für die einzelnen Parameter sind in der Regel unterschiedlich, da die Quantile von den zugrunde liegenden Verteilungsunktionen Fi der jeweiligen Parameter i, i=1, …, m, abhängen.

- Handelt es sich um einen diskreten Parameter, werden die Intervalle durch die diskreten Merkmalswerte des Parameters definiert.

Für einen Parameter wird die Struktur der Klassifikation der Beobachtungen einer Stichprobe vom Umfang n auf die jeweiligen Cluster und die fünf in (1.3) definierten Intervalle durch die Tab. 2.1 dargestellt.

Tab. 2.1 Klassifikation der Beobachtungsdaten eines Parameters X zur Quantifizierung der Entropie-Reduktion und des Informationsgewinns

Cluster	$\mathbf{I_1}$	$\mathbf{I_2}$	$\mathbf{I_3}$	$\mathbf{I_4}$	$\mathbf{I_5}$	$\sum$
1	$n_{1,1}$	$n_{2,1}$	$n_{3,1}$	$n_{4,1}$	$n_{5,1}$	n_1
2	$n_{1,2}$	$n_{2,2}$	$n_{3,2}$	$n_{4,2}$	$n_{5,2}$	n_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮
h	$n_{1,h}$	$n_{2,h}$	$n_{3,h}$	$n_{4,h}$	$n_{5,h}$	n_h

In Tab. 2.1 werden die Beobachtungen der Stichprobe für den Parameter X auf die definierten Intervalle I_1, …, I_5 und Cluster 1, …, h aufgeteilt. Die Zuordnung der Beobachtungen zu den Clustern erfolgt über die Ergebniswerte y und die Zuordnung in die Intervalle über die Stichprobenwerte (x_1, … , x_n) des Parameters X. Gemäß Tab. 2.1 ist die Verteilung der Beobachtungswerte des Parameters X bzgl. Cluster 1 über die definierten Intervalle gegeben durch

$n_{1;1}$ - Anzahl der Stichprobenwerte von Parameter X, die im Intervall I_1 liegen
$n_{2;1}$ - Anzahl der Stichprobenwerte von Parameter X, die im Intervall I_2 liegen,
⋮
$n_{5;1}$ - Anzahl der Stichprobenwerte von Parameter X, die im Intervall I_5 liegen.

Die $n_{j,k}$ bezeichnen somit die Anzahl der Beobachtungen, deren Ergebniswerte y in den Cluster k (k = 1, …, h) fallen und deren Stichprobenwerte für den Parameter X im Intervall I_j (j = 1, …, 5) liegen. Dabei ist $\sum_{j=1}^{5} n_{j,k} = n_k$ die Anzahl der Beobachtungen, die im Cluster k (k = 1, …, h) liegen. D. h., die Summe der Stichprobenwerte des Parameters X über die Intervalle bzgl. des Clusters k ist gleich der Anzahl der Beobachtungen n_k, die

in das Cluster k fallen. Die Summe der Beobachtungen über die Intervalle und Cluster ergibt die gesamte Anzahl der Beobachtungen in der Stichprobe, d. h.
$\sum_{j=1}^{5} \sum_{k=1}^{h} n_{j,k} = n.$

Die Anzahl n der Beobachtungen in der Stichprobe und die Anzahl der Beobachtungen in den jeweiligen Clustern ist für jeden Parameter $X_1, \ldots, X_m$ gleich. Bzgl. der unterschiedlichen Parameter unterscheiden sich die Beobachtungen somit nur in der Verteilung der Stichprobenwerte über die Intervalle.

Der Informationsgehalt der Verteilung der n Beobachtungen eines Parameters auf die Cluster wird durch die Entropie E_0 in Gleichung (2.2) ermittelt. Um zu berechnen, welche Entropie-Reduktion sich gegenüber E_0 erreichen lässt, wird die Entropie bzgl. verschiedener Trennungspunkte x_{split} berechnet. Dazu werden die Stichprobenwerte $(x_1, \ldots, x_n)$ des Parameters X in die Bereiche $x \leq x_{split}$ und $x > x_{split}$ aufteilt. Als Trennungspunkte x_{split} bieten sich mit Ausnahme des letzten Intervalls die oberen Klassengrenzen der Intervalle an. D. h. bei den in (2.3) angegebenen Intervallen werden für den Trennungspunkt die Werte $x_{split} = \xi_{20\%}, \xi_{40\%}, \xi_{60\%}$ und $\xi_{80\%}$ nacheinander eingesetzt. Der Trennungspunkt $x_{split} = \xi_{99.99\%}$ kommt nicht in Farge, da $\xi_{99.99\%}$ den maximalen Wert der Beobachtungen darstellt und damit keine Aufteilung des Wertebereichs in $x \leq x_{split}$ und $x > x_{split}$ mehr möglich ist.

Im Folgenden wird die Vorgehensweise zur Bestimmung der maximalen Entropie-Reduktion und des damit zusammenhängenden maximalen Informationsgewinns beschrieben. Die Berechnungen für die einzelnen Parameter werden analog zu der nachfolgen Beschreibung durchgeführt.

Im Folgenden soll die Frage beantwortet werden, welche Entropie-Reduktion sich für den Parameter X ergibt, wenn eine Aufteilung der Werte bei $x_{split} = \xi_{20\%}$ durchgeführt wird. Dazu werden die Entropien für die Bereiche $x \leq \xi_{20\%}$ und $x > \xi_{20\%}$ nach den Gleichungen (2.4) und (2.5) berechnet. Als Schätzwerte für p_i in Gleichung (2.1) werden die relativen Häufigkeiten eingesetzt, die sich aus den Klassifizierungen in Tab 2.1 ergeben:

$$E_{\leq \xi 20\%} = -\sum_{k=1}^{h} \frac{n_{1,k}}{S_1} \cdot \log_h \left(\frac{n_{1,k}}{S_1} \right) \qquad (2.4)$$

mit $S_1 = \sum_{k=1}^{h} n_{1,k}$

S_1 ist die Anzahl der Beobachtungen, bei denen die Stichprobenwerte von Parameter X im Intervall I_1 liegen

$$E_{> \xi 20\%} = - \sum_{k=1}^{h} \frac{\sum_{j=2}^{5} n_{j,k}}{S2} \cdot \log_h \left(\frac{\sum_{j=2}^{5} n_{j,k}}{S_2} \right) \quad (2.5)$$

$$\text{mit } S_2 = \sum_{k=1}^{h} \sum_{j=2}^{5} n_{j,k}$$

S_2 beschreibt die Anzahl der Beobachtungen, bei denen die Stichprobenwerte von Parameter X in den Intervallen I_2, …, I_5 liegen.

Die Gesamt-Entropie für den Trennungspunkt $\xi_{20\%}$ wird aus dem gewichteten Mittel der einzelnen Entropien $E_{\leq \xi 20\%}$ und $E_{> \xi 20\%}$ nach Gleichung (2.6) berechnet.

$$E_{\xi 20\%} = E_{\leq \xi 20\%} \cdot \frac{S_1}{n} + E_{> \xi 20\%} \cdot \frac{S_2}{n} \quad (2.6)$$

wobei $S_1 + S_2 = n$ die Anzahl der Beobachtungen in der Stichprobe ist.

Die Entropie-Reduktion $\Delta E_{\xi 20\%}$, die sich durch die Trennung des Wertebereichs bei $\xi_{20\%}$ ergibt, wird durch die Gleichung (2.7) ermittelt:

$$\Delta E_{\xi 20\%} = E_0 - E_{\xi 20\%} \quad (2.7)$$

Wie bereits in Abschnitt 2.4.2 diskutiert wurde, ist mit einer kleineren Entropie eine höhere Information der Verteilung verbunden. Mit zunehmender Information ist auch eine Zunahme der Wahrscheinlichkeit zu erwarten, mit der eine Beobachtung einem Cluster korrekt zugeordnet werden kann. Wenn die Entropie-Reduktion als proportional zu dem erzeugten Informationsgewinn betrachtet wird, kann der Informationsgewinn bei $\xi_{20\%}$ bzgl. der ursprünglichen Entropie E_0 durch Gleichung (2.8) berechnet werden:

$$IG_{\xi 20\%} = \frac{\Delta E_{\xi 20\%}}{E_0} = 1 - \frac{E_{\xi 20\%}}{E_0} \quad (2.8)$$

Die Berechnung der Entropien für den Trennungspunkt $x_{split} = \xi_{40\%}$ und $x_{split} = \xi_{60\%}$ erfolgen analog, wobei sich nur die Indizes der Summationen ändern. Ausführlich beschrieben werden noch die Gleichungen für $x_{split} = \xi_{80\%}$.

Die Berechnung der Entropien und des Informationsgewinns für den Trennungspunkt $x_{split} = \xi_{80\%}$ erfolgt durch:

$$E_{\leq \xi 80\%} = -\sum_{k=1}^{h} \frac{\sum_{j=1}^{4} n_{j,k}}{S_1} \cdot \log_2 \left(\frac{\sum_{j=1}^{4} n_{j,k}}{S_1} \right)$$

$$\text{mit } S_1 = \sum_{k=1}^{h} \sum_{j=1}^{4} n_{j,k}$$

$$E_{> \xi 80\%} = -\sum_{k=1}^{h} \frac{n_{5,k}}{S2} \cdot \log_2 \left(\frac{n_{5,k}}{S_2} \right)$$

$$\text{mit } S_2 = \sum_{k=1}^{h} n_{5,k}$$

$$E_{\xi 80\%} = E_{\leq \xi 80\%} \cdot \frac{S_1}{n} + E_{> \xi 80\%} \cdot \frac{S_2}{n}$$

$$\Delta E_{\xi 80\&} = E_0 - E_{\xi 80\%}$$

$$IG_{\xi 80\%} = \frac{\Delta E_{\xi 80\%}}{E_0} = 1 - \frac{E_{\xi 80\%}}{E_0}$$

Für den Parameter X ergibt sich die maximale Entropie-Reduktion $\Delta E_{max}(X)$ bzw. der maximale Informationsgewinn $IG_{max}(X)$ bei dem Trennungspunkt x_{split}, bei dem die Entropie-Reduktion bzw. der Informationsgewinn am größten ist, d. h.

$$\Delta E_{max}(X) = \max (\Delta E_{\xi 20\%}, \Delta E_{\xi 40\%}, \Delta E_{\xi 60\%}, \Delta E_{\xi 80\%}) \quad (2.9)$$

$$IG_{max}(X) = \max (IG_{\xi 20\%}, IG_{\xi 40\%}, IG_{\xi 60\%}, IG_{\xi 80\%}) \quad (2.10)$$

Durch die vollständige Abhängigkeit des Informationsgewinns von der Entropie-Reduktion, wird $\Delta E_{max}(X)$ und $IG_{max}(X)$ beim gleichen Trennungspunkt x_{split} erzielt.

Für alle Parameter X_1, …., X_m werden die Berechnungen analog durchgeführt und für jeden Parameter i (i = 1, …, m) wird die maximale Entropie-Reduktion $\Delta E_{max}(X_i)$ bzw. der maximale Informationsgewinn $IG_{max}(X_i)$ ermittelt.

2.4.3.2 Erzeugung von Zuordnungsregeln zu einem Ergebniscluster und Berechnung der Fehlerwahrscheinlichkeit der Zuordnungsregel

Die Entropie-Reduktion bzw. der Informationsgewinn eines Parameters wird durch die Verteilung der Daten auf die Ergebniscluster und die gewählten Intervalle bzgl. verschiedener Trennungspunkte des Parameters ermittelt. Durch den Vergleich des Informationsgewinns an den verschiedenen Trennungspunkten wird der maximale Informationsgewinn $IG_{max}(X_i)$ bzgl. der einzelnen Parameter X_i, i = 1, …, m bestimmt.

In diesem Abschnitt wird beschrieben, wie der maximale Informationsgewinn $IG_{max}(X)$ eines Parameters X verwendet werden kann, um die Bedeutung eines Parameters zur Clusterbildung zu ermitteln. Um die Clusterbildung von Funktionswerten möglichst gut durch die jeweiligen Parameter erklären zu können, besteht die grundlegende Idee darin, für jeden Parameter eine Regel zu finden, für die folgende beiden Kriterien gelten:

i) Die herzuleitende Diskriminanzregel eines Parameters soll mit einer möglichst kleinen Fehlerwahrscheinlichkeit verbunden sein.

ii) Es soll ein möglichst großer Anteil von Beobachtungen durch die Regel erfasst werden.

Je geringer die Fehlerwahrscheinlichkeit einer Diskriminanzregel bzgl. eines Parameters X ist, desto genauer lassen sich die Beobachtungen des Parameters auf die entsprechenden Cluster zuordnen. Aus der Fehlerwahrscheinlichkeit und dem Anteil der durch die Regel erfassten Beobachtungen kann somit der Anteil der korrekt zugeordneten Beobachtung für die betreffende Regel ermittelt werden. Je größer der Anteil der korrekt zugeordneten Beobachtungen für einen Parameter ist, desto besser kann die Clusterbildung durch den entsprechenden Parameter erklärt werden.

In diesem Abschnitt wird zunächst die Erzeugung einer Diskriminanzregel für den zugrundeliegenden Parameter und die Berechnung der mit der Diskriminanzregel verbundenen Fehlerwahrscheinlichkeit beschrieben, bevor im nachfolgenden Abschnitt 2.4.3.3 erläutert wird, wie der Beitrag des Parameters zur Clusterbildung berechnet wird.

Die unter Verwendung der Gleichungen (2.9) und (2.10) berechneten Größen der maximalen Entropie-Reduktion $\Delta E_{max}(X_i)$ bzw. des maximalen Informationsgewinns $IG_{max}(X_i)$ bzgl. der Parameter X_i, i=1, …, m, kann derjenige Parameter X bestimmt werden, bei dem der größte Informationsgewinn erzielt werden kann. Gleichzeitig ist bekannt, an welchem Trennungspunkt x_{split} der maximale Informationsgewinn bei diesem Parameter auftritt. Diese Informationen können nun verwendet werden, um bestimmte Zuordnungsregeln für den Parameter X zu erstellen. Da die Maße der maximalen Entropie-Reduktion und des maximalen Informationsgewinns gleichermaßen verwendet werden können, bezieht sich die weitere Beschreibung der Einfachheit halber nur noch auf den maximalen Informationsgewinn IG_{max}, da dieses Maß leichter zu interpretieren ist als das Maß der Entropie-Reduktion.

Mit der Kenntnis, bei welchem Trennungspunkt x_{split} der maximale Informationsgewinn eines Parameters erzielt werden konnte, ist der Wert des jeweiligen Parameters festgelegt, der als Trennungspunkt x_{split} in der aufzustellenden Regel zu verwenden ist. Durch den gegebenen Trennungspunkt x_{split} wird der Parameter in die Wertebereiche $x \leq x_{split}$ und $x > x_{split}$ aufgeteilt. Die Herleitung einer Zuordnungsregel für den Parameter X mit dem maximalen Informationsgewinn erfolgt in zwei Schritten:

Schritt 1: Im ersten Schritt wird die Zuordnungsregel für den Bereich erzeugt, für den der Anteil der korrekt zugeordneten Beobachtungen auf ein Cluster am größten ist. Um diesen Bereich zu erkennen, werden die Entropien $E_{\leq xsplit}$ und $E_{>xsplit}$ des Parameters betrachtet. Die zu erstellende Zuordnungsregelregel konzentriert sich dabei zunächst auf den Bereich mit der kleineren Entropie, da dort die Unsicherheit der Verteilung der Beobachtungen auf die Cluster geringer ist und somit einen höheren Informationsgewinn liefert. Mit x seien im Folgenden die Beobachtungswerte des Parameters X bezeichnet, bei dem der maximalen Informationsgewinn festgestellt wurde.

Im Fall $E_{>xsplit} < E_{\leq xsplit}$ wird folgende Zuordnungsregel (Z_X) aufgestellt:

(Z_X): Wenn $x > x_{split}$ gilt, dann ordne die Beobachtung dem Cluster zu, der die meisten Beobachtungen bzgl. des Bereichs $x > x_{split}$ enthält.

Im Fall $E_{\leq xsplit} < E_{>xsplit}$ lautet die Zuordnungsregel:

(Z_X): Wenn $x \leq x_{split}$ gilt, dann ordne die Beobachtung x dem Cluster zu, der die meisten Beobachtungen bzgl. des Bereichs $x \leq x_{split}$ enthält.

Schritt 2: Im zweiten Schritt wird die Zuordnungsregel für den Komplementärbereich erzeugt. Das ist der Bereich, der in der Zuordnungsregel (Z_X) im Schritt 1 nicht berücksichtigt wurde.

Bezieht sich die Zuordnungsregel (Z_X) auf den Bereich $x > x_{split}$, dann lautet die Regel (Z_X^C) für den Komplementärbereich:

(Z_X^C): Ordne alle Beobachtungen, für die der Wert x des Parameters $X \leq x_{split}$ ist, demjenigen Cluster zu, der die meisten Beobachtungen bzgl. des Bereichs $x \leq x_{split}$ enthält.

Bezieht sich die Zuordnungsregel (Z_X) auf den Bereich $x > x_{split}$, dann lautet die Regel (Z_X^C) für den Komplementärbereich:

(Z_X^C): Ordne alle Beobachtungen für die der Wert x des Parameters X > x_{split} ist, demjenigen Cluster zu, der die meisten Beobachtungen bzgl. des Bereichs x > x_{split} enthält.

Zur Veranschaulichung sei angenommen, dass der maximale Informationsgewinn bzgl. des Parameters X beim Trennungspunkt $x_{split} = \xi_{40\%}$ berechnet worden ist, wobei $\xi_{40\%}$ das 40 %-Quantil der Verteilung des Parameters X bezeichne. Weiter sei angenommen, dass $E_{\leq \xi 40\%} < E_{> \xi 40\%}$ gilt.

Da $E_{\leq \xi 40\%} < E_{> \xi 40\%}$ wird im 1. Schritt folgende Zuordnungsregel (Z_X) für die Beobachtungswerte x des Parameters X aufgestellt:

(Z_X): Falls $x \leq \xi_{40\%}$ ->, ordne Beobachtung x dem Cluster zu, der bzgl. des Bereichs $x \leq \xi_{40\%}$ die meisten Beobachtungen enthält. Angenommen, dies sei der Cluster 2 (C_2).

Anhand der Verteilung der Stichprobenwerte des Parameters X auf die Cluster und Intervalle (vgl. Tab. 2.1) kann nun die Fehlerwahrscheinlichkeit der Zuordnungsregel (Z_X) geschätzt werden. Nach der Regel wird jede Beobachtung x des Parameters X, deren Wert $x \leq \xi_{40\%}$ ist, dem Cluster C_2 zugeordnet. Eine richtige Zuordnung erfolgt somit für die Beobachtungen, für die $x \leq \xi_{40\%}$ gilt und die sich im Cluster C_2 befinden. Gemäß Tab. 2.1 sind das die Beobachtungen der Intervalle I_1 und I_2, die zum Cluster C_2 gehören. Damit ist die Anzahl der durch die Regel (Z_X) richtig zugeordneten Beobachtungen gegeben durch

$$N_{richtig} = n_{12} + n_{22} \tag{2.11}$$

Die Anzahl der durch die Regel (Z_X) falsch zugeordneten Beobachtungen ist gegeben durch

$$N_{falsch} = \sum_{j=1}^{2} \sum_{k=1}^{h} n_{j;k} - N_{richtig} = (n_{11} + n_{21}) + \sum_{j=1}^{2} \sum_{k=3}^{h-1} n_{j;k} \tag{2.12}$$

wobei $\sum_{j=1}^{2} \sum_{k=1}^{h} n_{j;k}$ die gesamte Anzahl der Beobachtungen im Bereich $x \leq \xi_{40\%}$ (d. h. in den Intervallen I_1 und I_2) ist.

Die falschen zugeordneten Beobachtungen bzgl. (Z_X) sind somit die Beobachtungen des Parameters X, bei denen der Parameter X einen Wert $x \leq \xi_{40\%}$ aufweist, aber die Beobachtung zu einem anderen Cluster als C_2 gehört.

Da die Zuordnungsregel und die damit verbundene Fehlerwahrscheinlichkeit auf der Grundlage einer Stichprobe von Beobachtungen ermittelt wird, muss man von mehr oder

weniger großen Unsicherheiten bei der Schätzung der Fehlerwahrscheinlichkeit ausgehen. Zur Berücksichtigung von Schätzunsicherheiten werden die Schätzungen der Fehlerwahrscheinlichkeiten unter Verwendung eines Bayes'schen Verfahrens mit nichtinformativer a-priori-Verteilung durchgeführt. Die Herleitung des Bayes'schen Verfahrens ist z. B. in /PES 95/ beschrieben. Mit diesem Verfahren erhält man als a-posteriori-Verteilungen der Fehlerwahrscheinlichkeit eine Beta(α,β)-Verteilung, mit der die Unsicherheiten der Schätzung ausgedrückt werden. Die Parameter der Betaverteilung sind gegeben durch $\alpha = N_{falsch} + 0.5$ und $\beta = N_{richtig} + 0.5$.

Unter der Bedingung der Zuordnungsregel (Z_X) ergibt sich somit als a-posteriori-Verteilung $\mathcal{F}$ der Fehlerwahrscheinlichkeit p_f eine Beta(N_{falsch} +0.5 ; $N_{richtig}$ + 0.5) -Verteilung, d. h.

$$\mathcal{F}(p_f \mid Z_X) = \frac{\Gamma(N_{falsch}+N_{richtig}+1)}{\Gamma(N_{falsch}+0.5)\cdot\Gamma(N_{richtig}+0.5)} \cdot p_f^{N_{falsch}-0.5} \cdot (1-p_f)^{N_{richtig}-0.5} \quad (2.13)$$

wobei die Anzahlen $N_{richtig}$ und N_{falsch} der richtigen bzw. falsch zugeordneten Beobachtungen durch die Gleichungen (2.11) und (2.12) ermittelt werden. Der Mittelwert (Erwartungswert) der a-posteriori Verteilung $\mathcal{F}(p_f \mid Z_X)$ ist gegeben durch

$$E(p_f \mid Z_X) = \frac{N_{falsch}+0.5}{N_{falsch}+N_{richtig}+1} \quad (2.14)$$

Die Verteilung $\mathcal{F}(p_f \mid Z_X)$ drückt die Unsicherheit bzgl. der Schätzung der Fehlerwahrscheinlichkeit p_f aus, wenn die Zuordnungsregel (Z_X) angewendet wird. Analog zur Verteilung der Fehlerwahrscheinlichkeit der Regel Z_X kann auch die a-posteriori-Verteilung $\mathcal{F}(p_r \mid Z_X)$ der Wahrscheinlichkeit der korrekten Zuordnungen bzgl. der Regel Z_X über das Bayes'sche Verfahren ermittelt werden. Die Verteilung lautet:

$$\mathcal{F}(p_r \mid Z_X) = \frac{\Gamma(N_{falsch}+N_{richtig}+1)}{\Gamma(N_{falsch}+0.5)\cdot\Gamma(N_{richtig}+0.5)} \cdot p_f^{N_{richtig}-0.5} \cdot (1-p_f)^{N_{falsch}-0.5}$$

Durch die Angabe von x_{split} und dem Bereich, auf den sich eine Zuordnungsregel Z_X bezieht, kann die Wahrscheinlichkeit geschätzt werden, mit der eine Beobachtung durch die Regel Z_X erfasst wird. Im obigen Beispiel wurde angenommen, dass sich der maximale Informationsgewinn beim Trennungspunkt $x_{split} = \xi_{40\%}$ ergeben hat, wobei $\xi_{40\%}$ das 40 %-Quantil der Verteilung des Parameters X bezeichnet. Des Weiteren wurde angenommen, dass sich die Zuordnungsregel Z_X auf den Bereich $x \le \xi_{40\%}$ bezieht und die Beobachtungen $x \le \xi_{40\%}$ dem Cluster C_2 zugeordnet werden.

Wenn die Werte der Funktionsparameter unabhängig voneinander ausgespielt werden, kann durch die Kenntnis des Bereichs, auf den sich die Zuordnungsregel Z_X bezieht, die Wahrscheinlichkeit bestimmt werden, mit der eine Beobachtung des Parameters durch die Regel Z_X erfasst wird. Wenn sich die Regel Z_X auf den Bereich $x \leq x_{split}$ bezieht, beträgt die Wahrscheinlichkeit $p_{erf}(Z_X)$, mit der eine Beobachtung durch Z_X erfasst wird $P(x \leq x_{split}) = q$. Bezieht sich die Regel auf den Bereich $x > x_{split}$, beträgt die Wahrscheinlichkeit $P(x > x_{split})$. Diese Wahrscheinlichkeiten können über die gegebene Verteilung F_X des Parameters X ermittelt werden. Wenn die Werte von x_{split} durch bestimmte Quantile $\xi_{q\%}$ der jeweiligen Parameter definiert sind, lassen sich die Wahrscheinlichkeiten einfach durch $p_{erf}(Z_X) = P(x \leq \xi_{q\%}) = q/100$ bzw. $p_{erf}(Z_X) = P(x > \xi_{q\%}) = 1 - q/100$ bestimmen.

Anhand der Verteilung bzgl. der Schätzung der Fehlerwahrscheinlichkeit $\boldsymbol{\mathcal{F}}(p_f \mid Z_X)$ in Gleichung (2.13) und der Wahrscheinlichkeit $p_{erf}(Z_X)$, dass eine Beobachtung von der Regel Z_X erfasst wird, kann die Wahrscheinlichkeit $p_{erf+falsch}$ berechnet werden, dass eine Beobachtung durch die Regel Z_X erfasst und einem falschen Cluster zugeordnet wird. Diese Wahrscheinlichkeit erhält man durch die Gleichung (2.15), je nachdem, ob sich die Regel (Z_X) auf den Bereich $x \leq \xi_{q\%}$ oder $x > x_{split}$ bezieht.

$$\boldsymbol{\mathcal{F}}_Z(p_{erf+falsch}) = \boldsymbol{\mathcal{F}}(p_f \mid Z_X) \cdot P(x \leq x_{split}) \text{ bzw.}$$

$$\boldsymbol{\mathcal{F}}_Z(p_{erf+falsch}) = \boldsymbol{\mathcal{F}}(p_f \mid Z_X) \cdot P(x > x_{split}) \qquad (2.15)$$

Die sich ergebende Verteilung in Gleichung (2.15) drückt die Unsicherheit bzgl. der Schätzung der Wahrscheinlichkeit $p_{erf+falsch}$ aus.

Analog kann die Verteilung $\boldsymbol{\mathcal{F}}(p_{erf+richtig})$ bzgl. der Wahrscheinlichkeit $p_{erf+richtig}$ berechnet werden, dass eine Beobachtung von der Regel Z_X erfasst wird und einem bestimmten Cluster korrekt zugeordnet wird.

$$\boldsymbol{\mathcal{F}}_{Z_X}(p_{erf+richtig}) = [1 - \boldsymbol{\mathcal{F}}(p_f \mid Z_X)] \cdot P(x \leq x_{split}) \text{ bzw.}$$

$$\boldsymbol{\mathcal{F}}_{Z_X}(p_{erf+richtig}) = [1 - \boldsymbol{\mathcal{F}}(p_f \mid Z_X)] \cdot P(x > x_{split})$$

Die Mittelwerte der Verteilungen $\boldsymbol{\mathcal{F}}_{Z_X}(p_{erf+falsch})$ und $\boldsymbol{\mathcal{F}}_{Z_X}(p_{erf+richtig})$ liefern die Grundlage zur Bewertung der Diskriminanzregel. Die Mittelwerte werden im Folgenden mit $\bar{p}_{erf+falsch}(Z_X)$ und $\bar{p}_{erf+richtig}(Z_X)$ bezeichnet.

In die Bewertung einer Zuordnungsregel Z gehen somit die Wahrscheinlichkeiten ein, mit denen durch die Regel Z_X eine Beobachtung falsch oder richtig einem Cluster zugeordnet wird, sowie die Wahrscheinlichkeit, mit der eine $_{Beobachtung}$ durch Z_X erfasst wird.

Im nachfolgenden Abschnitt 2.1.3.3 wird ein Verfahren zur Bewertung der Qualität einer Zuordnungsregel sowie zu einer Abschätzung des Beitrags vorgeschlagen, den ein Parameter zur Erklärung eines Clusters liefert.

2.4.3.3 Quantifizierung der Qualität der Zuordnungsregel und des Beitrags, den die jeweiligen Parameter zur Erklärung der Ergebniscluster liefern

Für die nachfolgende Herleitung wird von folgender Überlegung ausgegangen: Je geringer die Fehlerwahrscheinlichkeit einer Zuordnungsregel bzgl. eines Parameters X ist, desto genauer lassen sich die Beobachtungen des Parameters einem bestimmten Cluster zuordnen. Aus der Fehlerwahrscheinlichkeit bzw. der Wahrscheinlichkeit der korrekten Zuordnungen und dem Anteil der durch die Regel erfassten Beobachtungen kann somit die Wahrscheinlichkeit ermittelt werden, dass die Regel zur Anwendung kommt und damit eine korrekte Zuordnung auf ein bestimmtes Custer erfolgt. Ein Schätzwert dieser Wahrscheinlichkeit ist durch $\bar{p}_{erf+richtig}$ gegeben.

Um die Qualität einer Zuordnungsregel abschätzen zu können, reicht die alleinige Betrachtung der Wahrscheinlichkeit einer korrekten Zuordnung nicht aus. So kann beispielsweise der Fall eintreten, dass die Wahrscheinlichkeit einer korrekten Zuordnung $\bar{p}_{erf+richtig}(Z_{X1})$ durch die Regel (Z_{X1}) kleiner ist als die Wahrscheinlichkeit einer korrekten Zuordnung $\bar{p}'_{erf+richtig}(Z_{X2})$ durch die Regel(Z_{X2}), d. h. $\bar{p}_{erf+richtig}(Z_{X1}) < \bar{p}'_{erf+richtig}(Z_{X2})$. Wenn allerdings auch die Fehlerwahrscheinlichkeit $\bar{p}_{erf+falsch}(Z_{X1}) < \bar{p}'_{erf+falsch}(Z_{X2})$ ist, muss zur Beurteilung der Qualität von Zuordnungsregeln sowohl die Wahrscheinlichkeit einer korrekten als auch die Wahrscheinlichkeit einer fehlerhaften Zuordnung berücksichtigt werden.

Ein einfaches Verfahren, beide Wahrscheinlichkeiten zur Beurteilung der Qualität einer Zuordnungsregel zu berücksichtigen, besteht darin, beide Wahrscheinlichkeiten ins Verhältnis zu setzen, $\frac{\bar{p}_{erf+richtig}(Z_X)}{\bar{p}_{erf+falsch}(Z_X)}$. Dieses Verhältnis wird in der Statistik als ‚Odds' bezeichnet und liefert ein Maß wie oft ein Ereignis im Verhältnis zu einem Komplementärereignis

auftritt. Das Verhältnis Odds = $\frac{\bar{p}_{erf+richtig(Z_X)}}{\bar{p}_{erf+falsch}(Z_X)}$ liefert somit ein Maß dafür, wie oft bei einer Zuordnungsregel eine korrekte im Vergleich zu einer falschen Zuordnung erfolgt. Je größer das Verhältnis einer Zuordnungsregel ist, desto besser beschreibt die Regel den Wertebereich des jeweiligen Parameters X, mit dem die Werte dem entsprechenden Cluster korrekt zugeordnet werden, bei gleichzeitig geringerem Anteil der mit der Regel verbundenen Fehlerwahrscheinlichkeit.

Der Bereich der Zuordnungsregel mit dem höchsten Odds-Wert wird somit als derjenige interpretiert, mit dem die beste Zuordnung zu dem betreffenden Cluster erfolgt. Das bedeutet aber auch, dass der Parameter, dessen Zuordnungsregel den höchsten Odds-Wert aufweist, der mit dem größten Beitrag zum damit verbundenen Cluster ist.

2.4.3.4 Erweiterung auf multivariate Zuordnungsregeln

In Abschnitt 2.4.3.2 wurde eine Methode zur Erzeugung von Zuordnungsregeln beschrieben. Damit konnte festgestellt werden, mit welchem Parameter die Zugehörigkeit zu einem bestimmten Cluster mit der geringsten Fehlerwahrscheinlichkeit prognostiziert werden kann. Auch wenn durch die Zuordnungsregel des Parameters mit dem maximalen Informationsgewinn die geringste Fehlerwahrscheinlichkeit eingehalten wird, so werden generell nicht alle Beobachtungen des entsprechenden Parameterbereichs durch die Zuordnungsregel korrekt zugeordnet.

Durch die Erweiterung der Diskriminanzregel von einem auf mehrere Parameter wird versucht, die Fehlerwahrscheinlichkeit der Diskriminanzregel noch weiter zu minimieren. Dazu werden die bedingten Verteilungen der Parameter ermittelt, die sich unter der Bedingung der Regel (Z_X) ergeben. Durch einen Suchalgorithmus wird dann für jeden Parameter anhand der berechneten bedingten Verteilungsfunktionen ein Diskriminanzwert bestimmt, der eine möglichst gute Unterscheidung liefert, welchem Cluster eine Beobachtung zugeordnet werden muss. Anhand der ermittelten Diskriminanzwerte für die jeweiligen X_i können dann die Zuordnungen zu den Clustern erfolgen, die durch die ursprüngliche Regel (Z_X) nicht erfasst werden.

Für die erzeugten multivariaten Diskriminanzregeln wird berechnet, in welchem Ausmaß die Fehlerwahrscheinlichkeiten gegenüber der ursprünglichen univariaten Zuordnungsregel verringert werden können. Dieses eher heuristische Verfahren wird anhand eines Anwendungsbeispiels in Abschnitt 2.4.4.2 näher erläutert.

2.4.4 Demonstrationsbeispiel zur Anwendung der Methode ASPIC

Zur Veranschaulichung der in Abschnitt 2.4.3 beschriebenen Methodik zur quantitativen Bewertung von Einflussgrößen auf Ergebniscluster, soll der einfacheren Darstellung halber, eine Funktion mit nur drei Parametern (Einflussgrößen) angenommen werden. Die Methode ist jedoch analog auch für beliebig viele Parameter anwendbar.

Das Anwendungsbeispiel beruht auf einer Stichprobe von n = 100 Beobachtungen einer Funktion $G(x_1,x_2,x_3)$ mit drei Parametern X_1, X_2 und X_3. Die Funktion G(x) und die Verteilungen der Parameter sind gegeben durch:

$$G(x) = (X_1/X_2)^2 – X_3 \cdot X_2/X_1$$

mit

F_{X1} ~ Normal (200 , 30)
F_{X2} ~ Normal (150 , 40) und
F_{X3} ~ Uniform (1 , 5)

Anhand einer Stichprobe von 100 Zufallswerten, die aus den Verteilungen F_{X1}, F_{X2} und F_{X3} zufällig ausgespielt wurden, sind die Funktionswerte y der zugrundeliegenden Funktion G(x) berechnet worden. Die Spannweite der Funktionswerte der Stichprobe liegt zwischen -6 und 7.4. Als interessierende Ergebniscluster seien die Bereiche

$C_1 : y \leq -4,$

$C_2 : -4 < y \leq 2$ und

$C_3 : y > 2$

definiert. Die Werte der ersten zehn Beobachtungen sind nachfolgend aufgeführt (Tab. 2.2):

Tab. 2.2 Die ersten 10 Zufallswerte, die aus den Verteilungen Fx1, Fx2 und Fx3 ausgespielt wurden sowie die zugehörigen Funktionswerte G(x) und das jeweils zugeordnete Cluster

Beobachtung	X_1	X_2	X_3	G(x)	Cluster
1	195.07	232.77	4.06	-4.14	1
2	133.96	209.02	2.70	-3.81	2
3	199.67	124.70	2.48	1.01	2
4	215.84	81.65	1.65	6.36	3
5	170.88	166.00	3.88	-2.71	2
6	189.77	167.92	2.34	-0.79	2
7	196.31	154.15	2.30	-0.18	2
8	182.91	201.01	1.98	-1.34	2
9	205.29	282.96	3.87	-4.81	1
10	250.84	91.01	2.15	6.82	3
:	:	:	:	:	:

Die Verteilung der Funktionswerte auf die drei Cluster ist in Abb. 2.4 veranschaulicht, wobei sowohl auf der x-Achse als auch auf der y-Achse die Funktionswerte G(x) aufgetragen sind, so dass die Cluster auf der sich ergebenden Diagonalen besser veranschaulicht werden können. Aus Abb. 2.4 ist ersichtlich, dass die meisten Funktionswerte der Stichprobe im Cluster C_2 liegen.

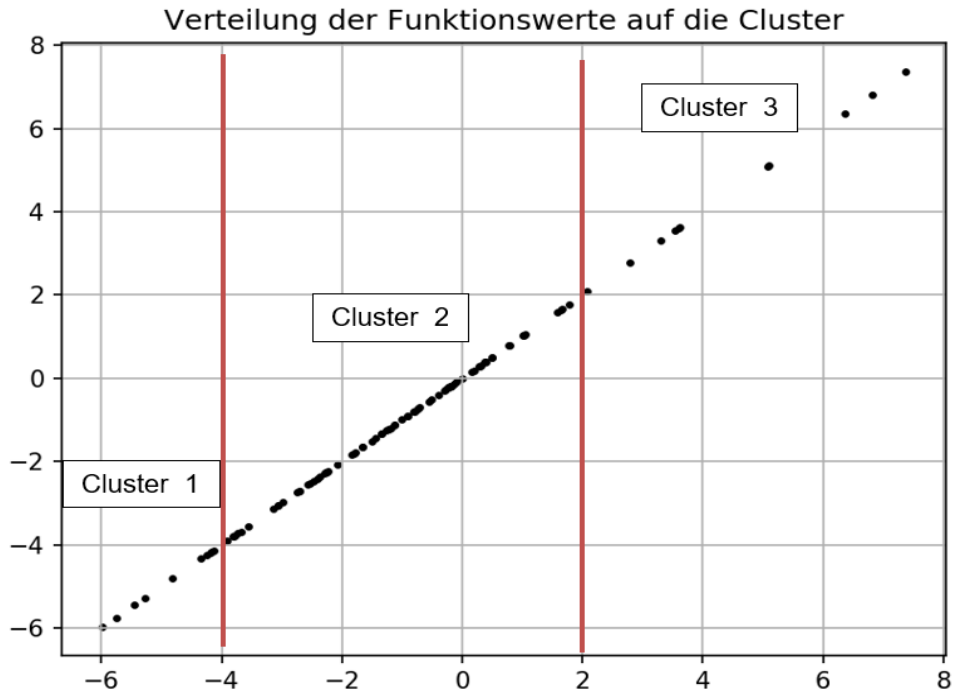


Abb. 2.4 Verteilung der Funktionswerte G(x) auf die drei definierten Cluster

In Abb. 2.5 werden die Funktionswerte in Abhängigkeit der Stichprobenwerte für jeden Parameter einzeln dargestellt. Aus der alleinigen Betrachtung der Beobachtungen der Stichprobe für jeden Parameter lässt sich kaum erkennen, welche Wertekombinationen zur Bildung der einzelnen Cluster führen. Was man aus der Betrachtung der Verteilung der Beobachtungswerte über die einzelnen Parameter erkennen kann, ist, dass bzgl. Parameter X_2 der größte Zusammenhang zwischen den Funktionswerten und den x-Werten besteht. Diesen Eindruck bestätigen die Pearson-Korrelationskoeffizienten zwischen Funktionswerten und den x-Werten der einzelnen Parameter. Für Parameter X_1, X_2 und X_3 ergeben sich die Korrelationskoeffizienten 0.35, -0.76 und -0.46.

Da bzgl. Parameter 2 betragsmäßig die größte Korrelation besteht, ist zu vermuten, dass Parameter 2 auch am ehesten dazu verwendet werden kann, um anhand der x-Werte die Zugehörigkeit zu einem Cluster zu prognostizieren. Anhand der Darstellung in Abb. 2.5 lässt sich allerdings nur schwer eine entsprechende Zuordnungsregel ableiten.

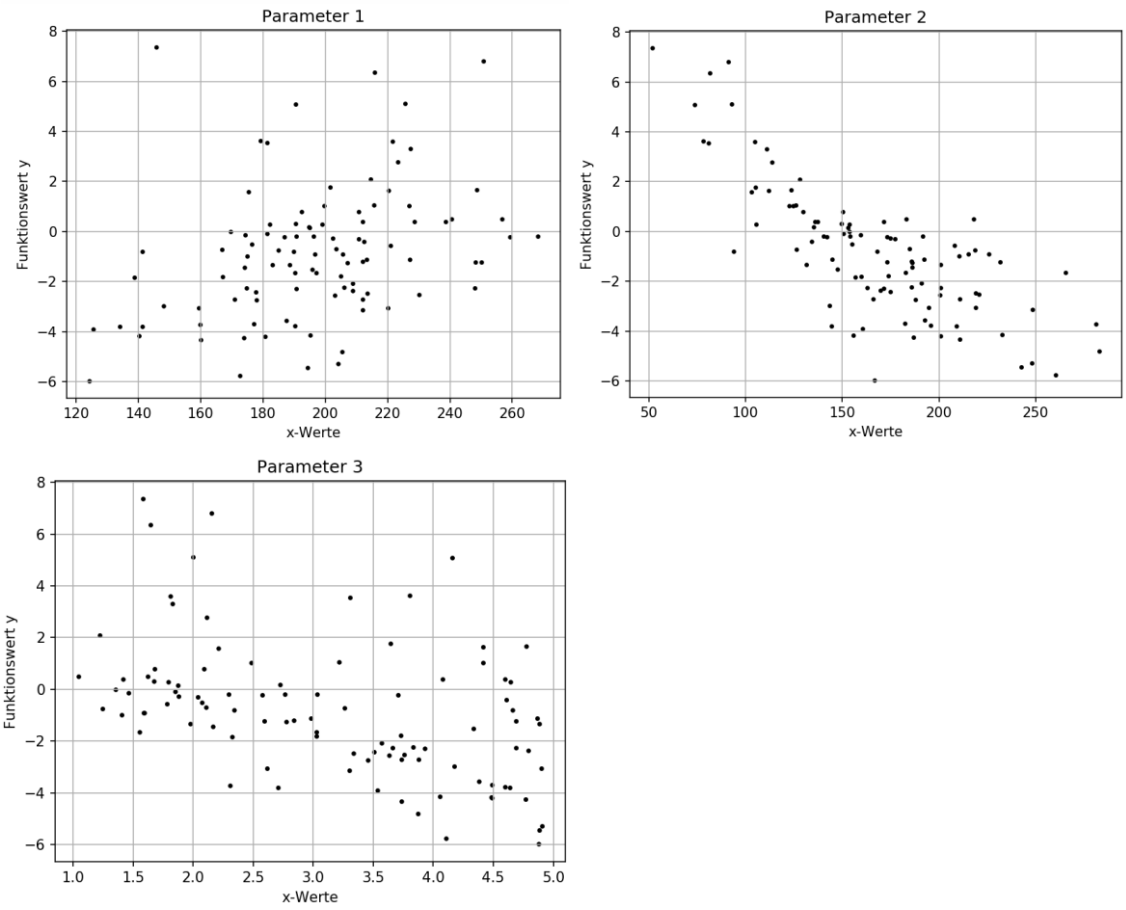


Abb. 2.5 Ergebnisse y in Abhängigkeit der x-Werte bzgl. der Parameter X_1, X_2 und X_3

Mit der Anwendung der entwickelten Methode ASPIC soll nun die Frage beantwortet werden, welchen Einfluss die Parameter auf die Verteilung der Ergebnisse auf die Cluster haben bzw. durch welche Wertekombinationen der Parameter die Ergebniscluster am besten erklärt werden können. Wie in Abschnitt 2.4.3 beschrieben, werden zur Erzeugung einer ersten univariaten Diskriminanzregel als maßgebliche Größen die Entropie-Reduktion sowie der damit verbundene Informationsgewinn verwendet. Anhand der erstellten Diskriminanzregel für jeden Parameter wird die damit verbundene Fehlerwahrscheinlichkeit und der Odds-Wert ermittelt. Anhand des Vergleichs der Odds-Werte wird derjenige Parameter bestimmt, der einen bestimmten Cluster am besten erklärt bzw. am meisten zur Erzeugung des Clusters beiträgt.

2.4.4.1 Ermittlung der maximalen Entropie-Reduktion bzw. des maximalen Informationsgewinns für die jeweiligen Parameter

Im ersten Schritt werden die Beobachtungsdaten der Stichprobe für jeden Parameter nach der in Tab. 2.1 (siehe Abschnitt 2.4.3) dargestellten Struktur klassifiziert. In dem

Anwendungsbeispiel werden die Beobachtungen in fünf Intervalle $I_1 = [\xi_0 , \xi_1]$, $I_2 = [\xi_1 , \xi_2]$, $I_3 = [\xi_2 , \xi_3]$, $I_4 = [\xi_3 , \xi_4]$, $I_5 = [\xi_4 , \xi_5]$ wobei ξ_1 das 20 %-Quantil des jeweiligen Parameters, ξ_2 das 40 %, ξ_3 das 60 % und ξ_4 das 80 %-Quantil des jeweiligen Parameters ist. In Tab. 2.3 sind die Verteilungen der Beobachtungen über die Intervalle I_1, …, I_5 und Ergebniscluster C_1, C_2 und C_3 separat für jeden Parameter angegeben.

Tab. 2.3 Verteilung der 100 Beobachtungen der Stichprobe auf die drei definierten Cluster und fünf Intervalle I_1, …, I_5 bzgl. der Parameter 1 - 3

	Intervallgrenzen bzgl. Parameter X_1						
Cluster	57.4 – 174.75	174.75 – 192.4	192.4 – 207.6	207.6 – 225.25	225.25 - 327.9	∑	Funktions-werte y
1	5	1	4	0	0	10	y ≤ -4
2	15	19	17	15	13	79	-4 < y ≤ 2)
3	1	3	0	4	3	11	y > 2

	Intervallgrenzen bzgl. Parameter X_2						
Cluster	-87.7 – 107.92	107.92 – 137.3	137.3 – 162.67	162.67 – 192.1	192.1 – 363.2	∑	Funktions-werte y
1	0	0	1	2	7	10	y ≤ -4
2	4	12	18	23	22	79	-4 < y ≤ 2)
3	8	3	0	0	0	11	y > 2

	Intervallgrenzen bzgl. Parameter X_3						
Cluster	1.0 – 1.8	1.8 – 2.6	2.6 – 3.4	3.4 – 4.2	4.2 – 5.0	∑	Funktions-werte y
1	0	0	0	4	6	10	y ≤ -4
2	14	17	14	16	18	79	-4 < y ≤ 2)
3	3	5	1	2	0	11	y > 2

Wie aus Tab. 2.3 ersichtlich, weisen zehn Beobachtungen der Stichprobe Funktionswerte ≤ -4 auf und liegen somit im Cluster C_1. 79 Beobachtungen weisen Funktionswerte -4 < y ≤ 2 auf und liegen im Cluster C_2. Der Cluster C_3 enthält elf Beobachtungen mit Funktionswerten y > 2.

Aus dieser Verteilung der Beobachtungen auf die Cluster C_1, C_2 und C_3 wird die vorliegende Basis-Entropie E_0 berechnet durch:

$$E_0 = - (0.1 * \log_3(0.1) + 0.79 * \log_3(0.79) + 0.11 * \log_3(0.11)) = 0.6$$

Da drei Cluster definiert sind, wird bei der Entropie-Berechnung der Logarithmus zur Basis 3 verwendet. Damit können die Entropie-Werte auf das Intervall zwischen 0 und 1 normiert werden, was die Einordnung und den Vergleich der Entropie-Werte erleichtert.

Die maximale Entropie würde sich ergeben, wenn die Beobachtungen gleichmäßig auf die drei Cluster verteilt wären, d. h. jeder Cluster 33.33 Beobachtungen enthielte. Die maximale Entropie wäre dann durch $E_{max} = 1.0$ gegeben. Maximale Entropie bedeutet in diesem Fall, dass uns die Verteilung der Beobachtungen auf die Cluster keine Informationen liefert, die unseren Kenntnisstand gegenüber völliger Unwissenheit bzgl. einer Prognose verbessert. D. h., wenn wir prognostizieren müssten, zu welchem Cluster eine Beobachtung gehört, liefert die Verteilung der Beobachtungen bei maximaler Entropie keine Vorteile gegenüber der Situation, als wenn die Prognose rein zufällig getroffen würde. In diesem Fall wäre die Fehlerwahrscheinlichkeit der Prognose mit ca. 67 % am größten.

Durch die Verteilung Funktionswerte auf die Cluster erhält man gegenüber der maximalen Entropie eine Entropie-Reduktion von $E_{max} – E_0 = 1. – 0.6 = 0.4$ bzw. einen Informationsgewinn von $(E_{max} – E_0)/E_{max} = 40$ %. Wird die Prognose nach der Strategie durchgeführt, dass jede neue Beobachtung automatisch dem Cluster zugeordnet wird, der die meisten Beobachtungen der Stichprobe enthält (d. h. im gegebenen Beispiel C_2), dann wäre nach der Verteilung der Stichprobenergebnisse auf die Cluster eine mittlere Fehlerwahrscheinlichkeit von 21 % zu erwarten. Gegenüber der Fehlerwahrscheinlichkeit von 67 % bei maximaler Entropie wäre dies eine Reduktion von 46 %.

Durch die Anwendung der Methode ASPIC soll nun untersucht werden, in welchem Maß eine weitere Optimierung des Informationsgewinns für jeden der drei Parameter erzielt werden kann. Aus diesen Informationen wird dann der Parameter bestimmt, der den maximalen Informationsgewinn erzielt. Für den Parameter mit dem maximalen Informationsgewinn wird dann eine erste Zuordnungsregel abgeleitet und die Fehlerwahrscheinlichkeit der erzeugten Zuordnungsregel berechnet. Die Zuordnungsregel beschreibt den Wertebereich des Parameters, der mit hoher Wahrscheinlichkeit einem bestimmten Cluster zugeordnet werden kann.

Für die Basisentropie der gegebenen Verteilung der Funktionswerte der Stichprobe auf die Cluster C_1, C_2 und C_3 wurde oben der Wert $E_0 = 0.6$ berechnet. Durch die Anwendung Methode ASPIC soll nun eine Optimierung der Entropie-Reduktion bzw. des Informationsgewinns durchgeführt werden. Dazu wird die Klassifizierung der 100 Beobachtungen

der Stichprobe gemäß Tabelle 2.2 verwendet, wobei für jeden Parameter separat der optimale Trennungspunkt x_{split} gesucht wird, bei dem der maximale Informationsgewinn erzielt werden kann. Zum besseren Verständnis wird das Vorgehen im Nachfolgenden ausführlich für den Parameter 1 demonstriert.

Für den Parameter X_1 werden nacheinander die Entropien für die Trennungspunkte $\xi_1 = 174.75$, $\xi_2 = 192.4$, $\xi_3 = 207.6$ und $\xi_4 = 225.5$ berechnet. Die einzelnen Trennungspunkte beschreiben jeweils das 20 %-, 40 %-, 60 %- und 80 %-Quantil der Verteilung des Parameters X_1.

Trennungspunkt bei $x_{split} = 174.75$:

Für Parameter X_1 wird der erste Split beim Trennungspunkt $x_{split} = 174.75$ durchgeführt. D. h., es wird die Verteilung derjenigen Werte von Parameter X_1 betrachtet, deren Werte ≤ 174.75 bzw. > 174.75 sind. Gemäß Tab. 2.3 liegen 21 Beobachtungen vor, bei denen der Parameter X_1 Werte aufweist, die ≤ 174.75 sind. Von den 21 Beobachtungen gehören fünf zum Cluster C_1, 15 zum Cluster C_2 und eine Beobachtung zum Cluster C_3. Bei 79 Beobachtungen liegen Werte von Parameter X_1 vor, die > 174.75 sind. Von den 79 Beobachtungen gehören fünf zum Cluster C_1 und 64 zum Cluster C_2 und zehn zum Cluster C_3.

Damit ergeben sich gemäß Gleichung (2.4) und (2.5) mit h = 3 für den Trennungspunkt $x_{split} = 174.75$ bzgl. Parameter X_1 folgende Entropien für die Bereiche $x \leq 174.75$ und $x > 174.75$:

$$E_{\leq 174.75} = -\left(\frac{5}{21} \cdot \log_3\left(\frac{5}{21}\right) + \frac{15}{21} \cdot \log_3\left(\frac{15}{21}\right) + \frac{1}{21} \cdot \log_3\left(\frac{1}{21}\right)\right) = 0.6617 \text{ und}$$

$$E_{> 174.75} = -\left(\frac{5}{79} \cdot \log_3\left(\frac{5}{79}\right) + \frac{64}{79} \cdot \log_3\left(\frac{64}{79}\right) + \frac{10}{79} \cdot \log_3\left(\frac{10}{79}\right)\right) = 0.5524$$

Um die Entropie $E_{\xi 1}$ beim Trennungspunkt $x_{split} = 174.75$ zu berechnen, werden die Teilentropien $E_{\leq \xi 1}$ und $E_{> \xi 1}$ mit der relativen Häufigkeit der Beobachtungen von Parameter X_1 gewichtet, deren Werte ≤ 174.75 bzw. > 174.75 sind, d. h.:

$$E_{174.75} = 0.6617 \frac{21}{100} + 0.5524 \frac{79}{100} = 0.5754$$

Durch den Split beim Trennungspunkt $x_{split} = 174.75$ kann die anfängliche Entropie E_0 um den Betrag von $E_0 - E_{174.74} = 0.6 - 0.5754 = 0.0246$ reduziert werden. Diese Entropie-Reduktion wird zur Berechnung des relativen Informationsgewinns $IG_{\xi 1}$ verwendet:

$$IG_{174.75} = \frac{(E_0 - E_{174.75})}{E_0} = \frac{0.0246}{0.6} = 0.041$$

Durch die Trennung der Werte bei $x_{split} = 174.75$ kann bzgl. Parameter X_1 ein Informationsgewinn von 4.1 % gegenüber der Information E_0 der ursprünglichen Stichprobenverteilung auf die Cluster erzielt werden.

Trennungspunkt bei $x_{split} = 192.4$:

Der nächste Trennungspunkt bzgl. Parameter X_1 wird bei $x_{split} = 192.4$ gesetzt. In diesem Fall weisen 44 Beobachten von Parameter X_1 Werte ≤ 192.4 und 56 Beobachtungen Werte > 192.4 auf. Von den 44 Beobachtungen mit Werten ≤ 192.4 gehören sechs zum Cluster C_1, 34 zum Cluster C_2 und vier Beobachtungen zum Cluster C_3. Von den 56 Beobachtungen mit x-Werten > 192.4 gehören vier zum Cluster C_1, 45 zum Cluster C_2 und sieben zum Cluster C_3.

Für den Trennungspunkt $x_{split} = 192.4$ ergeben sich bzgl. Parameter X_1 für die Bereiche $x \leq 192.4$ und $x > 192.4$ folgende Teil-Entropien:

$$E_{\leq 192.4} = -\left(\frac{6}{44} \cdot \log_3\left(\frac{6}{44}\right) + \frac{34}{44} \cdot \log_3\left(\frac{34}{44}\right) + \frac{4}{44} \cdot \log_2\left(\frac{4}{44}\right)\right) = 0.627 \text{ und}$$

$$E_{> 192.4} = -\left(\frac{4}{56} \cdot \log_3\left(\frac{4}{56}\right) + \frac{45}{56} \cdot \log_3\left(\frac{45}{56}\right) + \frac{7}{56} \cdot \log_3\left(\frac{7}{56}\right)\right) = 0.568$$

Für den Trennungspunkt $x_{split} = 192.4$ ergibt sich als Entropie:

$$E_{192.4} = 0.627\,\frac{44}{100} + 0.568 \cdot \frac{56}{100} = 0.594$$

Damit erhält man eine Entropie-Reduktion von

$$ER_{192.4} = E_0 - E_{192.4} = 0.6 - 0.594 = 0{,}006$$

und einen Informationsgewinn von

$$IG_{192.4} = \frac{(E_0 - E_{192.4})}{E_0} = \frac{0.006}{0.6} = 0.01$$

Durch die Trennung der Werte bei $x_{split} = 192.4$ kann bzgl. Parameter X_1 ein Informationsgewinn von 1 % gegenüber der Information E_0 der ursprünglichen Stichprobenverteilung (siehe Gleichung (2.2)) auf die Cluster erzielt werden.

Die Berechnungen bzgl. der Trennungspunkte x_{split} = 207.6 und x_{split} = 225.25 erfolgen analog. Die berechneten Werte des Informationsgewinns bei den verschiedenen Trennungspunkten werden im Folgenden verwendet, um den Trennungspunkt mit dem maximalen Informationsgewinn zu bestimmen.

Der maximale Informationsgewinn $IG_{max}(X_1)$ für Parameter X_1 wird nach Gleichung (2.10) aus dem Maximum der einzelnen Werte des Informationsgewinns ermittelt, die bzgl. der Trennungspunkte $\xi_1 = 174.75$, $\xi_2 = 192.4$, $\xi_3 = 207.6$ und $\xi_4 = 225.5$ berechnet wurden. Damit ergibt sich für den maximalen Informationsgewinn der Wert:

$$IG_{max}(X_1) = \max (IG_{\xi 1}, IG_{\xi 2}, IG_{\xi 3}, IG_{\xi 4}) = \max(0.041, 0.01, 0.093, 0.033) = 0.093$$

Bzgl. Parameter X_1 wird der größte Informationsgewinn beim Trennungspunkt $\xi_3 = 207.6$ erreicht. Zur Herleitung der Zuordnungsregel werden die damit verbundenen Teilentropien $E_{\leq \xi 3}$ und $E_{> \xi 3}$ herangezogen. Beim Trennungspunkt $\xi_3 = 207.6$ ergeben sich für die Teilentropien die Werte $E_{\leq \xi 3} = 0.592$ und $E_{> \xi 3} = 0.456$. Da die kleinere Entropie (d. h. größerer Informationsgewinn) bei $E_{> \xi 3}$ vorliegt, bezieht sich die Zuordnungsregel Z_{X1} bzgl. Parameter X_1 auf den Bereich $x_1 > 207.6$. Die Zuordnungsregeln für die einzelnen Parameter sind in Tab. 2.7 angegeben.

Für die beiden anderen Parameter werden die entsprechenden Berechnungen analog durchgeführt. In den Tabellen Tab. 2.4 bis Tab. 2.6 werden die berechneten Entropie-Werte der einzelnen Trennungspunkte für die jeweiligen Parameter zusammenfassend angegeben.

Tab. 2.4 Entropie, Entropie-Reduktion (ER) und Informationsgewinn (IG) an den Trennungspunkten von Parameter X_1

x_{split}	Entropie	$E_{\leq xsplit}$	$E_{>xsplit}$	ER	IG
$\xi_1 = 174.75$	0.575	0.6617	0.5524	0.025	0.041
$\xi_2 = 192.4$	0.594	0.627	0.568	0.006	0.01
$\xi_3 = 207.6$	0.544	0.592	0.456	0.056	0.093
$\xi_4 = 225.25$	0.58	0.607	0.44	0.02	0.033

Tab. 2.5 Entropie, Entropie-Reduktion (ER) und Informationsgewinn (IG) an den Trennungspunkten von Parameter X_2

x_{split}	Entropie	$E_{\leq xsplit}$	$E_{>xsplit}$	ER	IG
$\xi_1 = 107.9$	0.469	0.581	0.454	0.131	0.218
$\xi_2 = 137.3$	0.433	0.616	0.364	0.168	0.28
$\xi_3 = 162.7$	0.494	0.591	0.41	0.107	0.178
$\xi_4 = 192.1$	0.533	0.546	0.504	0.067	0.112

Tab. 2.6 Entropie, Entropie-Reduktion (ER) und Informationsgewinn (IG) an den Trennungspunkten von Parameter X_3

q_{split}	Entropie	$E_{\leq qsplit}$	$E_{>qsplit}$	ER	IG
$\xi_1 = 1.8$	0.58	0.426	0.612	0.02	0.033
$\xi_2 = 2.6$	0.533	0.463	0.577	0.068	0.113
$\xi_3 = 3.4$	0.512	0.412	0.63	0.088	0.147
$\xi_4 = 4.2$	0.546	0.557	0.513	0.054	0.09

Aus Tab. 2.5 ist abzulesen, dass bzgl. Parameter X_2 der größte Informationsgewinn $IG\text{-}_{max}(X_2)$ beim Trennungspunkt $\xi_2 = 137.3$ erzielt werden kann. Der dort erreichte Informationsgewinn gegenüber der Basisentropie E_0 der Ausgangsverteilung beträgt 28 %.

Bzgl. Parameter X_3 wird der größte Informationsgewinn $IG_{max}(X_3)$ beim Trennungspunkt $\xi_3 = 3.4$ erzielt (vgl. Tab. 2.6). Der dort erreichte Informationsgewinn gegenüber der Basisentropie E_0 der Ausgangsverteilung beträgt 14.7 %.

Im Nachfolgenden wird gezeigt, wie aus den gegebenen Informationen eine univariate Diskriminanzregel mit minimaler Fehlerwahrscheinlichkeit erstellt werden kann. Für jeden Parameter wurde der maximale Informationsgewinn sowie der Trennungspunkt ermittelt, bei dem der maximale Informationsgewinn erreicht wurde. Aus den für die drei Parameter berechneten Werten $IG_{max}(X_1) = 9.3$ %, $IG_{max}(X_2) = 28$ % und $IG_{max}(X_3) = 14.7$ % ergibt sich, dass sich der maximale Informationsgewinn über alle Parameter bzgl. Parameter 2 erzielen lässt, d. h.:

$$IG_{max} = \max (IG_{max}(X_1), IG_{max}(X_2), IG_{max}(X_3)) = IG_{max}(X_2) = 0.28$$

Der bzgl. Parameter X_2 erzielte maximale Informationsgewinn von ca. 28 % kann erreicht werden, wenn der Wertebereich des Parameters X_2 beim Trennungspunkt $\xi_2 = 137.3$ geteilt wird.

Um die Clusterbildung von Funktionswerten möglichst gut durch die jeweiligen Parameter erklären zu können, besteht die grundlegende Idee darin, eine Zuordnungsregel zu finden, mit der die Zuordnung der Beobachtungen auf die Cluster mit möglichst kleiner Fehlerwahrscheinlichkeit durchgeführt werden kann. Je geringer die Fehlerwahrscheinlichkeit einer Zuordnungsregel für einen Parameter ist, desto zuverlässiger kann der Wertebereich des Parameters bestimmt werden, der zum Cluster gehört und desto besser ist der Parameter geeignet, die Clusterbildung zu erklären.

Da der Parameter X_2 derjenige ist, bei dem der maximale Informationsgewinn erzielt wurde, bezieht sich die erste zu erzeugende Diskriminanzregel auf den Parameter X_2 und auf den Trennungspunkt $\xi_2 = 137.3$, bei dem sich der maximale Informationsgewinn ergeben hat. Zur Erstellung der Diskriminanzregel werden die Teilentropien $E_{\leq \xi 2} = 0.616$ und $E_{> \xi 2} = 0.364$ herangezogen (siehe Tab. 2.5). Die erste Regel wird für denjenigen Teilbereich mit der kleineren Entropie erstellt, da für diesen Teilbereich die Zuordnung die kleinere Unsicherheit und damit die geringere Fehlerwahrscheinlichkeit aufweist. Da $E_{> \xi 2} < E_{\leq \xi 2}$ bezieht sich die Zuordnungsregel auf den Teilbereich $x_2 > 137.3$, wobei x_2 die Parameterwerte von X_2 bezeichnen.

Die univariate Zuordnungsregel Z_{X2} bzgl. des Parameters X_2 lautet demnach:

Z_{X2} : if $x_2 > 137.3$ -> C_2

D. h., wenn der Wert x_2 des Parameters X_2 größer als 137.3 ist, dann ordne Beobachtung mit dem Wert x_2 dem Cluster C_2 zu. Die Stichprobenverteilung der Beobachtungen des Parameters X_2 (vgl. Tab. 2.3) zeigt, dass insgesamt 73 Beobachtungen vorliegen, für die $x_2 > 137.3$ gilt. Von diesen 73 liegen zehn Beobachtungen im Cluster C_1, 63 Beobachtungen im Cluster C_2 und null Beobachtungen im Cluster C_3.

Würden alle Beobachtungen der zugrundeliegenden Stichprobe mit $x_2 > 137.3$ gemäß der erstellten Diskriminanzregel dem Cluster C_2 zugeordnet, ergibt sich nach Gleichung (2.14) für die erwartete Fehlerwahrscheinlichkeit p_{falsch}

$$E(p_f \mid Z_{X2}) = \frac{10.5}{74} = 0.142$$

Würden die Beobachtungen der Stichprobe generell dem Cluster C_2 zugeordnet, d. h. ohne Berücksichtigung der Diskriminanzregel Z_{X2}, würde die Fehlerwahrscheinlichkeit ca. 21 % betragen. Durch diesen Vergleich wird deutlich, in welchem Ausmaß durch die Zuordnungsregel Z_{X2} eine Verringerung der erwarteten Fehlerwahrscheinlichkeit erzielt werden konnte. Die Unsicherheit, die sich für die Schätzung der Fehlerwahrscheinlichkeit bzgl. der Diskriminanzregel Z_{X2} ergibt, wird nach Gleichung (2.13) berechnet. Die 5 %-, 50 %- und 95 %-Quantile der a-posteriori-Verteilung $\boldsymbol{\mathcal{F}}$ (p_f | Z_{X2}) der Fehlerwahrscheinlichkeit p_f betragen 0.082, 0.139 und 0.213.

Durch die Kenntnis des Bereichs, auf den sich die Zuordnungsregel Z_{X2} bezieht, kann auch die Wahrscheinlichkeit bestimmt werden, mit der für den Parameter X_2 ein Wert vorliegt, der in den entsprechenden Bereich der Regel fällt und somit durch die Regel erfasst wird. Da sich in dem Beispiel Z_{X2} auf den Bereich $x_2 > 137.3$ bezieht und der Wert 137.3 das 40 %-Quantil der Verteilung des Parameters X_2 ist, kann damit die Wahrscheinlichkeit $p_{erf}(Z_{X2})$ bestimmt werden, mit der eine beliebige Beobachtung durch die Regel Z_{X2} erfasst wird. Aus der Kenntnis, dass 137.3 das 40 %-Quantil der Verteilung des Parameters X_2 ist, folgt $P(x_2 \leq 137.3) = 0.4$ und damit $p_{erf}(Z_{X2}) = P(x_2 > 137.3) = 0.6$.

Anhand der Verteilung der Fehlerwahrscheinlichkeit $\boldsymbol{\mathcal{F}}$ (p_f | Z_{X2}) aus Gleichung (2.13) und der Wahrscheinlichkeit $p_{erf}(Z_{X2})$, kann mittels Gleichung (2.15) die Verteilung $\boldsymbol{\mathcal{F}}_{Z_{X2}}$ ($p_{erf+falsch}$) der Wahrscheinlichkeit berechnet werden, dass eine Beobachtung durch die Regel Z_{X2} erfasst wird und dem Cluster C_2 falsch zugeordnet wird. Der Mittelwert der Verteilung beträgt $\bar{p}_{erf+falsch}(Z_{X2}) = 0.085$. Unter Berücksichtigung der Schätzunsicherheiten ergeben sich aus der a-posteriori-Verteilung $\boldsymbol{\mathcal{F}}_{Z_{X2}}$ ($p_{ferf+falsch}$) die Quantile [5 %, 50 %, 95 %] = [0.049, 0.083, 0.128].

Analog dazu kann die Verteilung $\boldsymbol{\mathcal{F}}_{Z_{X2}}(p_{erf+richtig})$ der Wahrscheinlichkeit berechnet werden, dass eine Beobachtung durch die Regel Z_{X2} erfasst wird und dem Cluster C_2 korrekt zugeordnet wird. Die mittlere Wahrscheinlichkeit beträgt $\bar{p}_{erf+richtig}(Z_{X2}) = 0.515$. Unter Berücksichtigung der Schätzunsicherheiten ergeben sich aus der a-posteriori-Verteilung $\boldsymbol{\mathcal{F}}_{Z_{X2}}(p_{erf+richtig})$ die Quantile [5 %, 50 %, 95 %] = [0.472, 0.517, 0.551].

Zusammenfassend sind in Abb. 2.7 die Ergebnisse der Methode ASPIC für jeden der drei Parameter separat dargestellt.

Tab. 2.7 Zusammenfassende Ergebnisse der Methode ASPIC bzgl. der Parameter X_1, X_2 und X_3

	Parameter X_1	**Parameter X_2**	**Parameter X_3**
Maximaler Informationsgewinn: Trennungspunkt:	0.093 207.6	0.28 137.3	0.147 3.4
Univariate Diskriminanzregel Z_{Xi}:	$Z(X_1)$: if $x_1 > 207.6$ -> C_2	$Z(X_2)$: if $x_2 > 137.3$ -> C_2	$Z(X_3)$: if $x_3 \leq 3.4$ -> C_2
Fehlerwkt: P(Fehler| Z_{Xi}) Mittelwert: 5 % - 50 % - 95 % Quantile:	 0.208 0.108, 0.203, 0.327	 0.142 0.082, 0.139, 0.213	 0.173 0.097, 0.169, 0.262
Wahrscheinlichkeit, dass Regel $Z(X_i)$ zur Anwendung kommt:	0.4	0.6	0.6
Wahrscheinlichkeit, dass Beobachtung durch Regel Z erfasst und falsch zugeordnet wird: Mittelwert: 5 % - 50 % - 95 % Quantile:	 0.083 0.04, 0.081, 0.15	 0.085 0.049, 0.083, 0.128	 0.104 0.055, 0.101, 0.17
Wahrscheinlichkeit, dass Beobachtung durch Regel Z erfasst und richtig zugeordnet wird: Mittelwert: 5 % - 50 % - 95 % Quantile:	 0.317 0.227, 0.319, 0.419	 0.515 0.472, 0.517, 0.551	 0.496 0.394, 0.498, 0.601
ODDS	3.82	6.06	4.77
Beitrag zur Erklärung des Clusters: Mittelwert: 5 % - 50 % - 95 % Quantile (in %):	C_2 35.6 % [27.1, 35.5, 44.6]	C_2 79.4 % [71.6, 79.6, 86.3]	C_2 56.9 % [47.7, 56.9, 65.8]

Aus Tab. 2.7 ist zu erkennen, dass der Parameter mit dem maximalen Informationsgewinn auch derjenige ist, bei dem die abgeleitete Zuordnungsregel die geringste mittlere Fehlerwahrscheinlichkeit (0.142) aufweist. Da sich die Zuordnungsregel Z nur auf einen bestimmten Bereich des entsprechenden Parameters bezieht, wird eine Beobachtung durch die Regel nur mit einer bestimmten Wahrscheinlichkeit erfasst. Das Produkt dieser Wahrscheinlichkeit mit der Wahrscheinlichkeit $P(p_f \mid Z_X)$ liefert eine Schätzung der Wahrscheinlichkeit, dass eine Beobachtung durch die Regel erfasst wird und dem durch die Regel ausgewiesenen Cluster falsch zugeordnet wird. Analog gilt dies für die Wahrscheinlichkeit, dass eine Beobachtung durch die Regel erfasst wird und dem durch die Regel ausgewiesenen Cluster richtig zugeordnet wird. In dem Beispiel beträgt die Wahrscheinlichkeit, dass eine Beobachtung von der Regel Z erfasst wird und dem Cluster C_2 richtig zugeordnet wird, $\bar{p}_{erf+richtig}(Z_{X1}) = 0.317$ für Parameter X_1, $\bar{p}_{erf+richtig}(Z_{X2}) = 0.515$ für Parameter X_2 und $\bar{p}_{erf+richtig}(Z_{X3}) = 0.496$ für Parameter X_3.

Es ist darauf zu achten, dass eine alleinige Betrachtung von $\bar{p}_{erf+falsch}(Z_X)$ oder $\bar{p}_{erf+richtig}(Z_X)$ zu einer verzerrten Einschätzung führen kann. Dies wird an dem Vergleich der Werte zwischen Parameter X_1 und X_2 ersichtlich. Obwohl der Parameter X_1 den geringsten Informationsgewinn aufweist, ist die Wahrscheinlichkeit, dass eine Beobachtung durch die abgeleitete Zuordnungsregel erfasst und falsch zugeordnet wird mit 0.083 ebenfalls am geringsten. Dagegen ist die Wahrscheinlichkeit $\bar{p}_{erf+falsch}(Z_{X2})$ für den Parameter X_2, bei dem der größte Informationsgewinn ermittelt wurde, mit 0.085 etwas größer, was den eigentlichen Erwartungen widerspricht. Dies liegt daran, dass durch die Zuordnungsregel Z_{X1} im Mittel nur ca. 40 % des Wertebereiches von Parameter X_1 erfasst werden, während die Zuordnungsregel $Z(X_2)$ im Mittel ca. 60 % des Wertebereiches von Parameter X_2 abdeckt. Die Wahrscheinlichkeiten, mit denen eine Beobachtung durch die Regel erfasst wird, tragen dazu bei, dass die alleinige Betrachtung von $\bar{p}_{erf+falsch}(Z_X)$ zu einem verzerrten Eindruck führen kann. Betrachtet man zusätzlich die Wahrscheinlichkeit, dass eine Bobachtung durch die Zuordnungsregel erfasst und dem Cluster C_2 richtig zugeordnet wird, weist Parameter X_1 mit $\bar{p}_{erf+richtig}(Z_{X1}) = 0.317$ eine erheblich geringere Wahrscheinlichkeit auf als der Parameter X_2 mit $\bar{p}_{erf+richtig}(Z_{X2}) = 0.515$.

Zur Bewertung der Qualität einer Zuordnungsregel sowie zu einer Abschätzung des Beitrags, den ein Parameter zur Erklärung eines Clusters liefert, müssen beide Wahrscheinlichkeiten $\bar{p}_{erf+falsch}(Z_X)$ und $\bar{p}_{erf+richtig}(Z_X)$ betrachtet werden. Dazu werden beide Wahrscheinlichkeiten ins Verhältnis $\frac{\bar{p}_{erf+richtig}(Z_X)}{\bar{p}_{erf+falsch}(Z_X)}$ gesetzt. Dieses Verhältnis wird als ‚Odds' bezeichnet und liefert ein Maß dafür, wie oft bei einer Zuordnungsregel eine korrekte im Vergleich zu einer falschen Zuordnung erfolgt (vgl. Abschnitt 2.4.3.3). Die Zuordnungsregel des Parameters mit dem höchsten Odds-Wert wird somit als diejenige betrachtet, mit der die beste Zuordnung zu dem betreffenden Cluster erfolgt. Das bedeutet aber auch, dass derjenige Parameter, dessen Zuordnungsregel den höchsten Odds-Wert aufweist, derjenige mit dem größten Beitrag zum damit verbundenen Cluster ist.

Gemäß Tab. 2.7 weist der Parameter X_2 den höchsten Odds-Wert auf, bei dem auch der maximale Informationsgewinn ermittelt wurde. Wird die Diskriminanzregel Z_{X2} auf die Stichprobe angewendet, werden 63 Beobachtungen der Stichprobenergebnisse dem Cluster 2 korrekt zugeordnet. Insgesamt sind im Cluster C_2 79 Beobachtungen der Stichprobe enthalten. Der Beitrag, den Parameter X_2 durch die Zuordnungsregel Z_{X2}: if $x_2 > 137.3$ -> C_2 zur Erklärung des Clusters C_2 liefert, beträgt somit im Mittel 79.4 %. Diese

Schätzung ergibt sich aus dem Mittelwert der a-posteriori-Verteilung, die über den Bayes'schen Ansatz berechnet wurde. Die Beiträge der restlichen Parameter sowie die Quantile der a-posteriori-Verteilung können der Tab. 2.7 entnommen werden.

2.4.4.2 Methode zur Erweiterung auf multivariate Diskriminanzregeln

Mit der Erstellung der univariaten Diskriminanzregel Z_{X2} wurde der Wertebereich desjenigen Parameters definiert, der den größten Beitrag zur Erklärung des Clusters C_2 liefert und somit von allen drei Parametern die zuverlässigste Zuordnung zum Cluster C_2 erlaubt. Um Diskriminanzregeln herzuleiten, mit der auch eine Zuordnung zu den restlichen Clustern C_1 und C_3 mit möglichst kleiner Fehlerwahrscheinlichkeit durchgeführt werden kann, wird die bisher ermittelte univariate Diskriminanzregel Z_{X2} auf eine multivariate Diskriminanzregel erweitert. Dazu erfolgt zunächst eine weitere Untersuchung derjenigen Beobachtungen, die durch die Regel Z_{X2} erfasst werden. Das sind die Beobachtungen der Stichprobe, bei denen der Parameter X_2 die Werte $x_2 > 137.3$ annimmt. Zur Herleitung von multivariaten Diskriminanzregeln wird ein eher heuristisches Verfahren verwendet, das im Nachfolgenden beschrieben wird.

Werden nur diejenigen Beobachtungen betrachtet, für die $x_2 > 137.3$ gilt, erhält man 73 Beobachtungen, von denen 63 zum Cluster C_2 und zehn zum Cluster C_1 gehören. Keines der verbleibenden 73 Beobachtungen befindet sich im Cluster C_3. Die Aufgabe besteht nun darin, die Diskriminanzregel Z_{X2} so zu erweitern, dass Beobachtungen mit möglichst hoher Wahrscheinlichkeit dem richtigen Cluster zugewiesen werden können.

Um die Regel zu erweitern kann untersucht werden, wie sich die Struktur der Daten in den einzelnen Clustern unterscheidet. Die erste grobe Information erhält man dadurch, in dem die Mittelwerte der Parameter bzgl. der einzelnen Cluster verglichen werden. Da die 73 Beobachtungen nur in den Clustern C_1 und C_2 liegen, werden die Mittelwerte in Tab. 2.8 nur zwischen diesen beiden Clustern verglichen.

Tab. 2.8 Mittelwerte der Parameter bzgl. der Cluster C_1 und C_2

Parameter	Cluster C_1	Cluster C_2
X_1	175.0	196.5
X_2	218.7	184.1
X_3	4.42	2.9

Die Aufgabe besteht nun darin, für die jeweiligen Parameter diejenigen Wertebereiche zu finden, die eine möglichst genaue Zuordnung in den Cluster C_1 erlauben. Einen ersten Hinwies bekommt man aus dem Vergleich der Mittelwerte in Tab. 2.8. Dort erkennt man, dass der Mittelwert des Parameters X_1 bzgl. Cluster C_1 kleiner als der bzgl. Cluster C_2 ($175.0 < 196.5$) ist. Für den Parameter X_2 ist der Mittelwert bzgl. C_1 größer als der bzgl. C_2 ($218.7 > 184.1$) und für den Parameter X_3 ist der Mittelwert bzgl. C_1 ebenfalls größer als der bzgl. C_2. D. h., zur Erklärung des Clusters C_1 ist zu erwarten, dass bzgl. Parameter X_1 eher kleinere Werte und bzgl. der Parameter X_2 und X_3 eher größere Werte vorliegen. Im Nachfolgenden wird untersucht, für welche Wertebereiche der Parameter eine möglichst optimale Zuordnung in den Cluster C_1 möglich ist.

Da in der Regel Z_{X2} bereits eine Einschränkung des Wertebereichs bzgl. X_2 durch $x_2 > 137.3$ gegeben ist, sollen zunächst die Wertebereiche bzgl. X_1 und X_3 gefunden werden, die zu einer möglichst optimalen Zuordnung zum Cluster C_1 führen. In Abb. 2.6 sind die bedingten Verteilungen für die jeweiligen Parameter X_1, X_2 und X_3 angegeben, die sich unter der Bedingung ergeben, dass die Beobachtungen im Cluster C_1 bzw. C_2 liegen und $x_2 > 137.3$ gilt.

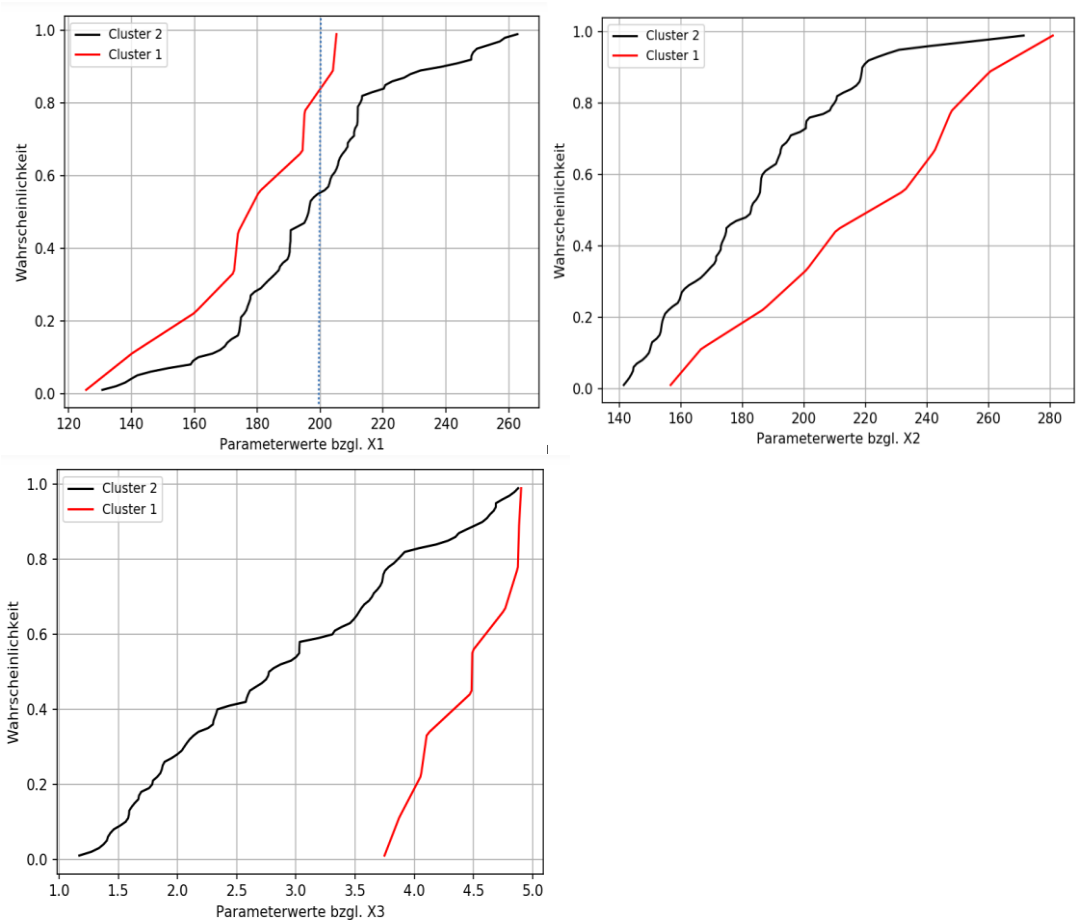


Abb. 2.6 Bedingte Verteilungen der Werte von Parameter X_1, X_2 und X_3 unter der Bedingung, dass die Beobachtungen im Cluster C_1 bzw. C_2 liegen und $x_2 > 137.3$ gilt

Die bedingten Verteilungen in Abb. 2.6 spiegeln die Situation wider, die bereits in Tab. 2.8 über die Mittelwerte angedeutet wurden. Die Werte bzgl. Parameter X_1 sind im Cluster C_1 geringer als im Cluster C_2. Die Parameter X_2 und X_3 weisen im Cluster C_1 dagegen höhere Werte auf als im Cluster C_2. Im Folgenden wird versucht, für jeden Parameter X_i einen Diskriminanzwert D_i, i=1,2,3, zu finden, der eine möglichst gute Unterscheidung liefert, ob die Beobachtung im Cluster C_1 oder C_2 liegt.

Im ersten Schritt soll ein Diskriminanzwert D_1 bzgl. Parameter X_1 gefunden werden, der eine möglichst genaue Zuordnung zum Cluster C_1 erlaubt. Da die Verteilungswerte bzgl. Parameter X_1 im Cluster C_1 kleiner sind als im Cluster C_2, hat die Regel zur Bestimmung des optimalen Diskriminanzwertes für den Parameter X_1 folgende Struktur:

$$\text{if } x_2 > 137.3 \text{ and } x_1 \leq D_1 \text{ -> } C_1 \qquad (2.16)$$

Die Suche nach einem optimalen Diskriminanzwert basiert auf folgender Idee: Durch die Wahl eines optimalen Diskriminanzwertes soll erreicht werden, dass durch die Regel $x_1 \leq D_1$ möglichst viele Beobachtungen der Stichprobe dem Cluster C_1 korrekt zugewiesen und möglichst wenig Beobachtungen falsch zugewiesen werden. Falsch zugewiesene Beobachtungen sind in diesem Fall diejenigen, die fälschlicherweise dem Cluster C_1 zugewiesen werden, obwohl sie zum Cluster C_2 gehören oder fälschlicherweise dem Cluster C_2 zugewiesen werden, obwohl sie zum Cluster C_1 gehören.

Als Kriterium zur Auswahl eines optimalen Diskriminanzwertes D_1 wird ein Verhältniswert R verwendet, der sich aus dem Quotienten zwischen der Anzahl der korrekt und der Anzahl der falsch zugeordneten Beobachtungen ergibt.

$R = n_r / (n_{f1} + n_{f2})$ wobei

n_r - die Anzahl der dem Cluster C_1 korrekt zugewiesenen Beobachtungen,

n_{f1} - die Anzahl der Beobachtungen, die dem Cluster C_1 falsch zugewiesen wurden,

n_{f2} – die Anzahl der Beobachtungen, die dem Cluster C_2 falsch zugewiesen wurden

bezeichnet.

Aus der Tab. 2.3 ist bekannt, dass im Cluster C_1 zehn Beobachtungen vorliegen. Wenn durch die Regel in Gleichung (2.16) n_r Beobachtungen dem Cluster C_1 korrekt zugewiesen werden, bedeutet dies, dass $n_{f2} = 10\text{-}n_r$ Beobachtungen fälschlicherweise dem Cluster C_2 zugewiesen werden. Wenn die Anzahl der Beobachtungen in C_1 $n' > 10$ ist, bedeutet dies, dass $n_{f1} = n' –$ zehn Beobachtungen dem Cluster C_1 falsch zugewiesen wurden. Die Gesamtzahl der durch die Gleichung (2.16) erfolgten falschen Zuordnungen beträgt somit $n_{f1} + n_{f2}$.

Der Bereich, in dem für Parameter X_1 ein optimaler Diskriminanzwert gesucht wird, ist der, wo die bedingten Verteilungen bzgl. der Cluster den gleichen Wertebereich abdecken. Gemäß Abb. 2.6 ist dieser Bereich für den Parameter X_1 ungefähr zwischen den Werten 135 und 210 gegeben. Aus diesem Bereich wird für verschiedene Werte von D_1 berechnet, wieviel Beobachtungen nach der Regel in Gleichung (2.16) korrekt bzw. falsch zugeordnet werden. Es wird derjenige Diskriminanzwert ausgewählt, bei dem der Verhältniswert R am größten ist. Liegen mehrere Diskriminanzwerte mit maximalem R vor, wird der optimale Diskriminanzwert in Anhängigkeit von der Form der Regel gewählt.

Da sich bei der Regel in Gleichung (2.16) die Zuordnung zum Cluster C_1 auf den Bereich $x_1 \leq D_1$ bezieht, wird derjenige Wert als optimaler Diskriminanzwert gewählt, bei dem der maximale R-Wert zum ersten Mal auftritt. Die Ergebnisse verschiedener Diskriminanzwerte sind in Tab. 2.9 angegeben.

Tab. 2.9 Verhältniswert R und Anzahl der Beobachtungen, die bzgl. verschiedener Diskriminanzwerte bzgl. Parameter X_1 dem Cluster C_1 richtig bzw. falsch zugeordnet wurden

Diskriminanzwert D_1	**Korrekte Zuordnung zu C_1**	**Falsche Zuordnungen**	**R**
140	1	12	0.083
150	2	13	0.154
160	2	15	0.133
170	3	16	0.1875
174	5	15	0.333
175	5	19	0.263
180	5	23	0.217
190	6	28	0.214
200	8	37	0.216

Zur Veranschaulichung liefert Tab. 2.9 die Ergebnisse einiger ausgewählter Diskriminanzwerte. Zur gesamten Übersicht ist der Verlauf der Verhältniswerte R für den Parameter X_1 über den Bereich der Diskriminanzwerte zwischen 135 und 240 in Abb. 2.7 dargestellt.

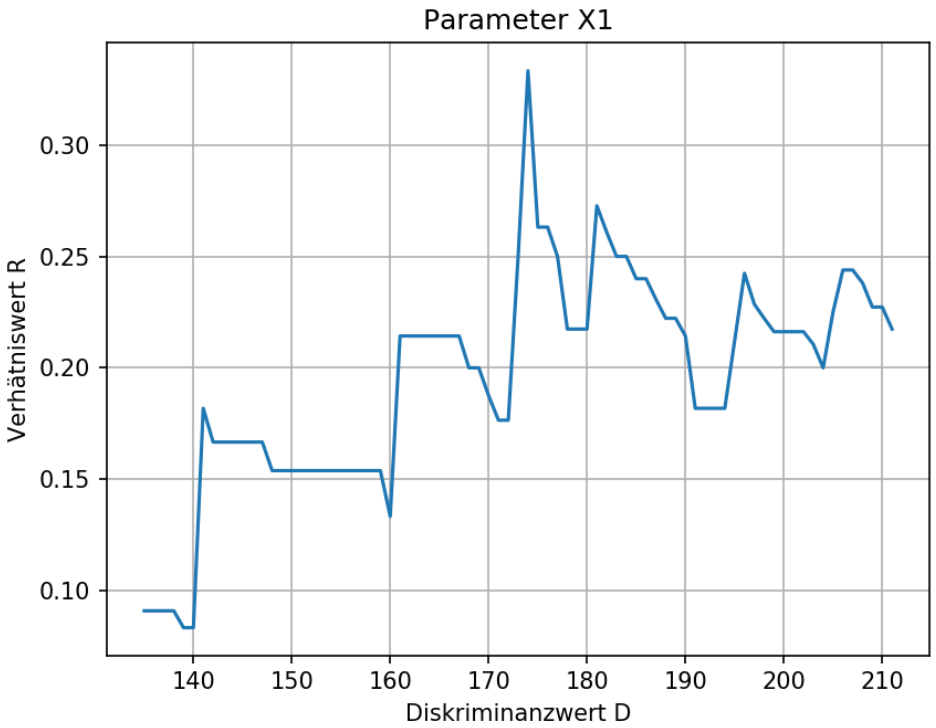


Abb. 2.7 Verlauf des Verhältniswertes R bzgl. Parameter X_1

Aus Abb. 2.7 und Tab. 2.9 ist zu erkennen, dass sich der maximale Verhältniswert R bei D=174 ergibt. Nach Erweiterung der Regel Z_{X2} um den optimalen Diskriminanzwert für den Parameter X_1 ergibt sich die Diskriminanzregel:

$Z_{X2,X1}$: if $x_2 > 137.3$ and $x_1 \leq 174$ -> C_1
if $x_2 > 137.3$ and $x_1 > 174$ -> C_2

Wendet man die Regel $Z_{X2,X1}$ auf die Beobachtungen der Stichprobe an, werden von den 73 Beobachtungen, für die $x_2 > 137.3$ gilt, 58 Beobachtungen den Clustern C_1 und C_2 korrekt und 15 Beobachtungen falsch zugeordnet. Dabei werden fünf Beobachtungen fälschlicherweise dem Cluster C_2 und zehn fälschlicherweise dem Cluster C_1 zugeordnet. Im Vergleich der Fehlerwahrscheinlichkeit von 0.142 unter der Regel Z_{X2} wäre mit der Regel $Z_{X2,X1}$ eine Erhöhung der Fehlerwahrscheinlichkeit auf ca. 0.21 verbunden.

Analog zu Parameter X_1 wird eine Regel für den Parameter X_3 bzgl. der Zuordnung zum Cluster C_1 hergeleitet. Aus Abb. 2.6 kann entnommen werden, dass die Werte von Parameter X_3 im Cluster C_1 größer sind als die Werte im Cluster C_2. Die Regel für die Zuordnung zum Cluster C_1 hat damit die Form if $x_3 > D_3$ -> C_1. Anhand des Verlaufs der bedingten Verteilungen in Abb. 2.6 beschränkt sich die Suche nach dem optimalen Diskriminanzwert für Parameter X_3 auf den Bereich zwischen 3.6 und 4.7. Die Tab. 2.10

zeigt die Ergebnisse verschiedener Diskriminanzwerte für den Parameter X_3. In Abb. 2.9 ist der Verlauf der Verhältniswerte R über den Bereich von 3.6 bis 4.7 dargestellt.

Tab. 2.10 Verhältniswerte R und Anzahl der korrekt, falsch zugeordneten Beobachtungen für verschiedene Diskriminanzwerte D_3 bzgl. Parameter X_3

Diskriminanzwert D_3	Korrekte Zuordnung zu C_1	Falsche Zuordnung	R
3.6	10	20	0.5
3.7	10	18	0.556
3.8	9	15	0.6
3.9	8	14	0.571
4.0	8	13	0.615
4.1	7	14	0.5
4.2	6	14	0.43

Abb. 2.8 Verlauf des Verhältniswertes R bzgl. Parameter X_3

Der maximale R-Wert stellt sich nach Abb. 2.9 und Tab. 2.10 bei dem Diskriminanzwert $D_3 = 4.0$ ein. Bzgl. des Parameters X_3 lautet die Regel für die Zuordnung zum Cluster C_1 somit:

if $x_2 > 137.33$ and $x_3 > 4.0$ -> C_1.

Nach Erweiterung der Regel Z_{X2} um die Regel für Parameter X_3 ergibt sich die Diskriminanzregel

$Z_{X2, X3}$: if $x_2 > 137.33$ and $x_3 > 4.0$ -> C_1

if $x_2 > 137.33$ and $x_3 \leq 4.0$ -> C_2

Wird die Regel $Z_{X2, X3}$ auf die Beobachtungen der Stichprobe angewendet, werden von den 73 Beobachtungen 55 korrekt und 18 Beobachtungen falsch zugeordnet. Die Fehlerwahrscheinlichkeit unter der Bedingung der Regel $Z_{X2, X3}$ beträgt damit 0.25, was gegenüber der Regel Z_{X2} ebenfalls eine Verschlechterung wäre.

Wenn die ursprüngliche Regel Z_{X2} um die Regeln für X_1 und X_3 zusammen erweitert wird, ergibt sich die Diskriminanzregel

$Z_{X2,X1,X3}$: if $x_2 > 137.3$ and $x_1 \leq 174$ and $x_3 > 4.0$ -> C_1

if $x_2 > 137.3$ and ($x_1 > 174$ or $x_3 \leq 4.0$) -> C_2

Wird die Regel $Z_{X2,X1,X3}$ auf die Beobachtungen der Stichprobe angewendet, werden von den 73 Beobachtungen, für die $x_2 > 137.3$ gilt, 65 Beobachtungen korrekt und acht Beobachtungen falsch zugeordnet. Mit der multivariaten Regel $Z_{X2,X1,X3}$ erhält man somit gegenüber der ursprünglichen Regel Z_{X2} eine etwas verringerte Fehlerwahrscheinlichkeit von 0.115.

Bisher wurde für den Parameter X_2 der Diskriminanzwert $x_2 > 137.3$, der bzgl. der univariaten Regel Z_{X2} ermittelt wurde, beibehalten. Im letzten Schritt wird noch ein Diskriminanzwert für den Parameter X_2 gesucht, um eine weitere Verringerung der Fehlerwahrscheinlichkeit zu erreichen. Die Suche nach dem optimalen Diskriminanzwert D_2 erfolgt nun aber in Kombination mit den Diskriminanzwerten, die für die Parameter X_1 und X_3 ermittelten wurden, d. h. $x_1 \leq 174$ und $x_3 > 4.0$.

Mit $x_1 \leq 174$ und $x_3 > 4.0$ hat die Suche für den optimalen Diskriminanzwert von X_2 den Wert $D_2 = 155$ ergeben, was der Abb. 2.9 entnommen werden kann.

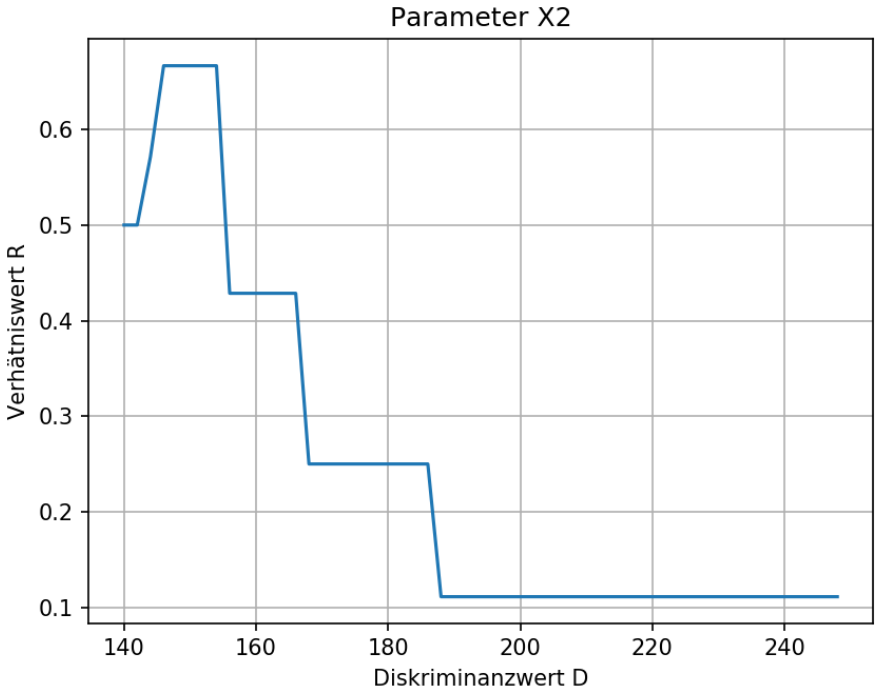


Abb. 2.9 Verlauf des Verhältniswertes R bzgl. Parameter X_3

Unter Verwendung der für die einzelnen Parameter ermittelten optimalen Diskriminanzwerte D_1, D_2 und D_3 ergibt sich die Diskriminanzregel:

$Z_{X1, X2, X3}$: if $x_1 \leq 174$ and $x_2 > 155$ and $x_3 > 4.0$ -> C_1
if $x_1 > 174$ or $137.3 \leq x_2 \leq 155$ or $x_3 \leq 4.0$ -> C_2

Bei Anwendung der Regel $Z_{X1, X2, X3}$ werden 67 Beobachtungen den jeweiligen Clustern korrekt zugeordnet. Sechs Beobachtungen, die zu C_2 gehören, werden mit der Regel dem Cluster C_1 falsch zugeordnet. Damit ergibt sich eine mittlere Fehlerwahrscheinlichkeit von 0.088, was gegenüber der mittleren Fehlerwahrscheinlichkeit von 0.142 bei Anwendung der univariaten Regel Z_{X2} eine Verringerung um ca. 62 % bedeutet.

Die geschätzte Verteilung $\mathcal{F}$ (p_{falsch} | $Z_{X1, X2, X3}$), die die Unsicherheit bzgl. der Fehlerwahrscheinlichkeit beschreibt, ist somit durch eine Beta(6.5 ; 67.5)-Verteilung gegeben. Die Quantile der geschätzten Verteilung $\mathcal{F}$ (p_{falsch} | $Z_{X1, X2, X3}$) betragen:

5 %-, 50 %-, 95 %-Quantil = [0.041, 0.084, 0.147].

Um die Qualität der Diskriminanzregel $Z_{X1, X2, X3}$ zu untersuchen, wird eine Simulationsstudie mit wachsender Anzahl von Simulationen durchgeführt. Durch die Simulationsstudie soll überprüft werden, ob die bei den Simulationen aufgetretenen Fehlerwahrscheinlichkeit im 90 %-Bereich (d. h. innerhalb des 5 %- und 95 %-Quantils) der oben

angegebenen Verteilungsschätzung liegt. Durch die wachsende Anzahl der Simulationsläufe soll das Konvergenzverhalten der Schätzungen überprüft werden. Neben den mittleren Fehlerwahrscheinlichkeiten werden für die Simulationen auch die Anteile der Beobachtungen geschätzt, die von der Diskriminanzregel erfasst werden. Die Ergebnisse der Simulationen sind in Tab. 2.11 angegeben.

Tab. 2.11 Fehlerwahrscheinlichkeit und Anteil der erfassten Beobachtung bei Anwendung der multivariaten Diskriminanzregel $Z_{X1,\ X2,\ X3}$ für unterschiedliche Stichprobenumfänge

Anzahl der Simulationen	Mittlere Fehlerwahrscheinlichkeit	Anteil der erfassten Beobachtungen
200	0.072	0.635
500	0.063	0.572
1 000	0.078	0.619
5 000	0.067	0.602
10 000	0.066	0.599
50 000	0.064	0.6
100 000	0.066	0.599
500 000	0.065	0.6

Die Tab. 2.11 veranschaulicht, dass sich die Schätzungen der mittleren Fehlerwahrscheinlichkeiten bzgl. der Diskriminanzregel $Z_{X1,\ X2,\ X3}$ sowohl bei den niedrigeren als auch bei den hohen Simulationsumfängen im 90 %-Bereich der geschätzten Verteilung $\mathcal{F}$ (p_{falsch} | $Z_{X1,\ X2,\ X3}$) liegen. Damit zeigt sich, dass die mit der hergeleiteten Diskriminanzregel ausgewiesene Fehlerwahrscheinlichkeit eine zuverlässige Schätzung darstellt. Der Anteil der durch die Regel erfassten Beobachtungen beträgt ca. 60 %.

Die multivariate Diskriminanzregel $Z_{X1,\ X2,\ X3}$ wurde für diejenigen Beobachtungen der Stichprobe hergeleitet, für die $x_2 > 137.3$ gilt. Damit bleibt noch die Herleitung der Regel für den Komplementärbereich durchzuführen, d. h. für diejenigen Beobachtungen, für die $x_2 \leq 137.3$ gilt und die nicht von der Diskriminanzregel $Z_{X1,\ X2,\ X3}$ erfasst werden.

2.4.4.3 Erzeugung der Diskriminanzregel für den Komplementärbereich

Da unter Verwendung der Zuordnungsregel $Z_{X1,\ X2,\ X3}$ 73 Beobachtungen erfasst und den Clustern C_1 und C_2 zugeordnet wurden, verbleiben im Komplementärbereich, für den denen der Parameter X_2 Werte $x_2 \leq 137.3$ aufweist, noch 27 Beobachtungen. Die

Verteilungen der Parameterwerte der 27 Beobachtungen auf die Cluster und Intervalle sind in Tab 2.11 angegeben. In der vorliegenden Stichprobe liegen für den im Komplementärbereich keine Beobachtungen vor, die im Cluster C_1 liegen. Bzgl. der Parameter X_1, X_2 und X_3 ergeben sich für den Komplementärbereich die in Tab. 2.12 dargestellte Verteilung der Werte auf die Cluster 2 und 3.

Tab. 2.12 Verteilung der 27 Beobachtungen des Komplementärbereichs ($x_2 \leq 137.3$) der Stichprobe auf die drei definierten Cluster und fünf Intervalle I_1, …, I_5 bzgl. der Parameter X_1, X_2 und X_3

	Intervallgrenzen bzgl. Parameter X_1						
Cluster	57.4 – 174.75	174.75 – 192.4	192.4 – 207.6	207.6 – 225.25	225.25 – 327.9	∑	Funktionswerte y
1	0	0	0	0	0	0	y ≤ -4
2	2	3	4	3	4	16	-4 < y ≤ 2)
3	1	3	0	4	3	11	y > 2

	Intervallgrenzen bzgl. Parameter X_2						
Cluster	-87.7 – 107.92	107.92 – 137.33	137.33 – 162.67	162.67 – 192.1	192.1 – 363.2	∑	Funktionswerte y
1	0	0	0	0	0	0	y ≤ -4
2	4	12	0	0	0	16	-4 < y ≤ 2)
3	8	3	0	0	0	11	y > 2

	Intervallgrenzen bzgl. Parameter X_3						
Cluster	1.0 – 1.8	1.8 – 2.6	2.6 – 3.	3.4 – 4.2	4.2 – 5.0	∑	Funktionswerte y
1	0	0	0	0	0	0	y ≤ -4
2	0	3	3	2	8	16	-4 < y ≤ 2)
3	3	5	1	2	0	11	y > 2

Die univariate Diskriminanzregel für den Komplementärbereich, die sich durch die im Abschnitt 2.4.3 beschriebene Methode ergibt, lautet.

Z^C_{X2}: if $x_2 \leq 137.3$ -> C_2

Mit der Regel Z^C_{X2} des Komplementärbereichs ergibt sich eine mittlere Fehlerwahrscheinlichkeit von $P(\text{Fehler} \mid Z^C_{X2}) = 0.41$. Analog zu der in Abschnitt 2.4.4.2 beschrieben Methodik, wird im Folgenden die Regel Z^C_{X2} auf eine multivariate Zuordnungsregel erweitert, um auch für den Komplementärbereich eine Minimierung der Fehlerwahrscheinlichkeit der Diskriminanzregel zu erreichen.

Aus der Tab. 2.12 ist ersichtlich, dass sich die 27 Beobachtungen des Komplementärbereichs auf die Cluster C_2 und C_3 verteilen. In Abb. 2.10 sind die bedingten Verteilungen der einzelnen Parameter für die Beobachtungen dargestellt, die sich in den Clustern C_2 bzw. C_3 befinden. Zusätzliche Bedingung ist, dass nur die Beobachtungen des Komplementärbereiches betrachtet werden, d. h. diejenigen Beobachtungen, für die $x_2 \leq 137.3$ gilt.

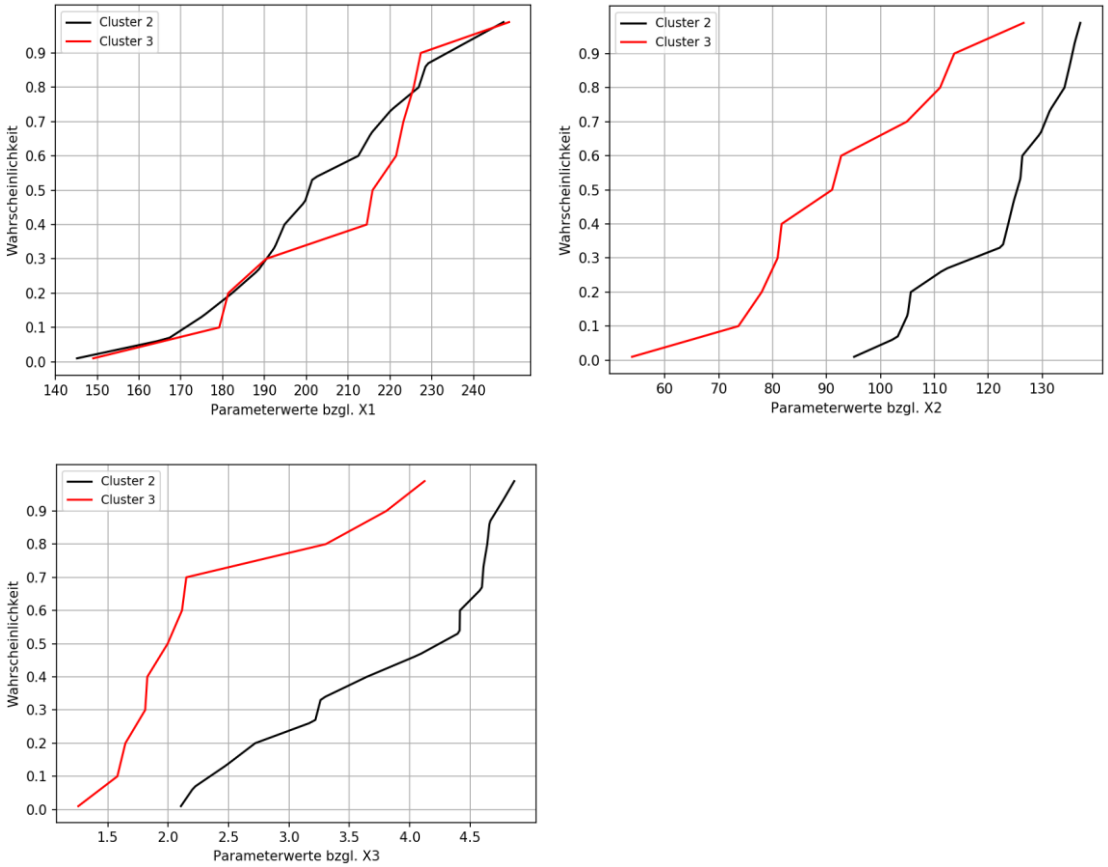


Abb. 2.10 Bedingte Verteilungen der Werte von Parameter X_1, X_2 und X_3 unter der Bedingung, dass die Beobachtungen im Cluster C_2 bzw. C_3 liegen und $x_2 \leq 137.3$ gilt

Zur Bestimmung der multivariaten Diskriminanzregel für den Komplementärbereich wird zunächst für jeden Parameter ein optimaler Diskriminanzwert nach der in Abschnitt 2.4.4.2 beschriebenen Methodik bestimmt. Die optimalen Diskriminanzwerte und die Anzahlen der damit verbundenen Fehler und korrekten Zuordnungen zum Cluster C_3 sind in Tab. 2.13 angegeben.

Tab. 2.13 Optimale Diskriminanzwerte der Parameter X_1 und X_3 und damit verbundene Anzahlen der korrekten und falschen Zuordnungen unter der Bedingung $x_2 \leq 137$

Parameter	Optimaler Diskriminanzwert D
1	$x_1 > 145$
2	$x_2 \leq 128$
3	$x_3 \leq 2.2$

Anhand der für die einzelnen Parameter ermittelten optimalen Diskriminanzwerte in Tab. 2.13 ergibt sich folgende multivariate Diskriminanzregel für den Komplementärbereich:

$Z^C_{X1,X2,X3}$:

if ($x_1 > 145$) and ($x_2 \leq 128$) and ($x_3 \leq 2.2$) -> C_3

if ($x_1 \leq 145$) or ($128 < x_2 \leq 137.33$) or ($x_3 > 2.2$) -> C_2

Bei Anwendung der Regel $Z^C_{X1,X2,X3}$ für den Komplementärbereich werden 24 von den 27 im Komplementärbereich befindlichen Beobachtungen den jeweiligen Clustern C_3 und C_2 korrekt zugeordnet. Drei Beobachtungen, die zu C_2 gehören, werden fälschlicherweise dem Cluster C_3 zugeordnet. Die Verteilung der Fehlerwahrscheinlichkeit $\mathcal{F}$ (p_{falsch} | $Z^C_{X1,X2,X3}$) folgt somit einer Beta(3.5 ; 24.5)-Verteilung. Für den Komplementärbereich ergibt sich damit eine mittlere Fehlerwahrscheinlichkeit von 0.125, was gegenüber der mittleren Fehlerwahrscheinlichkeit von 0.41 bei Anwendung der univariaten Regel Z^C_{X2} eine signifikante Verringerung darstellt. Die 5 %-, 50 %-, 95 %-Quantile der geschätzten Verteilung der Fehlerwahrscheinlichkeit $\mathcal{F}$ (p_{falsch} | $Z^C_{X1,X2,X3}$) betragen [0.041, 0.116, 0.239].

Die gemeinsame Anwendung der hergeleiteten Diskriminanzregeln $Z_{X1,\ X2,\ X3}$ und $Z^C_{X1,\ X2,\ X3}$ lautet:

if ($x_1 \leq 174$) and ($x_2 > 155$) and ($x_3 > 4.0$) -> C_1

if ($x_1 > 145$) and ($x_2 \leq 128$) and ($x_3 \leq 2.2$) -> C_3

if ($x_1 \leq 145$ or $x_1 > 174$) or ($128 < x_2 \leq 155$) or ($2.2 < x_3 \leq 4.0$) -> C_2

Für die Diskriminanzregel $Z_{X1,\ X2,\ X3}$ ist die Unsicherheiten bzgl. der Schätzung der Fehlerwahrscheinlichkeit durch die Verteilung $\mathcal{F}$ (p_{falsch} | $Z_{X1,\ X2,\ X3}$) ~ Beta(6.5, 67.5) gegeben,

für die Diskriminanzregel $Z^{C}_{X1,\ X2,\ X3}$ des Komplementärbereiches durch $\mathcal{F}\ (p_{falsch} \mid Z^{C}_{X1,\ X2,\ X3}) \sim$ Beta(3.5, 24.5).

Die Abschätzung der Fehlerwahrscheinlichkeit bei der gemeinsamen Anwendung der Diskriminanzregeln erfolgt über die Mischverteilung aus den beiden Verteilungen mit gleichen gewichten, d. h.

$$\mathcal{F}\ (p_{falsch}) = \mathcal{F}\ (p_{falsch} \mid Z_{X1,\ X2,\ X3}) * 0.5 + \mathcal{F}\ (p_{falsch} \mid Z^{C}_{X1,\ X2,\ X3}) * 0.5$$

Die sich daraus ergebenden 5 %-, 50 %-, und 95 %-Quantile der Fehlerwahrscheinlichkeit betragen [5 %, 50 %, 95 %] = [0.041, 0.096, 0.21]. Das 95 %-Konfidenzintervall der Mischverteilung beträgt 95 %-KI = (0.034 , 0.24)

Zusammenfassend sind die erzeugten multivariaten Diskriminanzregeln und deren zugehörige Fehlerwahrscheinlichkeiten in Tab. 2.14 angegeben.

Tab. 2.14 Multivariate Diskriminanzregeln und zugehörige Fehlerwahrscheinlichkeiten

Diskriminanzregel	**Fehlerwahrscheinlichkeit**			
	5 %	**50 %**	**95 %**	**Mean**
$Z_{X1,\ X2,\ X3}$: if $x_1 \leq 174$ and $x_2 > 155$ and $x_3 > 4.0$ -> C_1 else -> C_2	$\mathcal{F}\ (p_{falsch} \mid Z_{X1,\ X2,\ X3})$:			
	0.04	0.084	0.147	0.088
$Z^{C}_{X1,\ X2,\ X3}$: if $x_1 > 145$ and $x_2 \leq 128$ and $x_3 \leq 2.2$ -> C_3 else -> C_2	$\mathcal{F}\ (p_{falsch} \mid Z^{C}_{X1,\ X2,\ X3})$:			
	0.041	0.116	0.239	0.125
$Z_{X1,\ X2,\ X3} \cup Z^{C}_{X1,\ X2,\ X3}$: if $x_1 \leq 174$ and $x_2 > 155$ and $x_3 > 4.0$ -> C_1 if $x_1 > 145$ and $x_2 \leq 128$ and $x_3 \leq 2.2$ -> C_3 if ($x_1 \leq 145$ or $x_1 > 174$) or ($128 < x2 \leq 155$) or ($2.2 < x_3 \leq 4$) -> C_2	$\mathcal{F}\ (p_{falsch})$:			
	0.041	0.096	0.21	0.107
	95%-KI = (0.034 , 0.24)			

Um die Validität der Diskriminanzregel und der damit verbundenen Fehlerwahrscheinlichkeit zu überprüfen, wurde eine Simulation mit 1000000 zufällig ausgespielten Testdaten durchgeführt. Bei der Anwendung der Diskriminanzregeln $Z_{X1,\ X2,\ X3}$ und $Z^{C}_{X1,\ X2,\ X3}$ hat sich dabei eine Fehlerwahrscheinlichkeit von ca. 0.198 ergeben. Dieser Wert liegt im 95 %-KI der Mischverteilung $\mathcal{F}\ (p_{falsch})$ und ist etwas kleiner als das 95 %-Quantils.

Dass die Fehlerwahrscheinlichkeit im oberen Bereich der Schätzung durch $\mathcal{F}\ (p_{falsch})$ liegt, wird durch die begrenzte Stichprobe vom Umfang n = 100 verursacht, anhand der die Diskriminanzregeln abgeleitet wurden. Bei der Simulation hat sich gezeigt, dass sich ein Großteil der fehlerhaften Zuordnungen (ca. 85 %) durch falsche Zuordnungen in das

Cluster C_2 ergeben. D. h., Beobachtungen die eigentlich zu C_1 bzw. C_3 gehören, wurden durch die Diskriminanzregel fälschlicherweise dem Cluster C_2 zugeordnet. Insgesamt können jedoch durch die erzeugten Diskriminanzregeln diejenigen Parameterkombinationen angegeben werden, die mit einer relativ hohen Genauigkeit von ca. 80 % die jeweiligen Cluster erklären.

Als zusammenfassender Überblick ist in Abb. 2.11 ein Ablaufdiagramm der entwickelten Methode dargestellt.

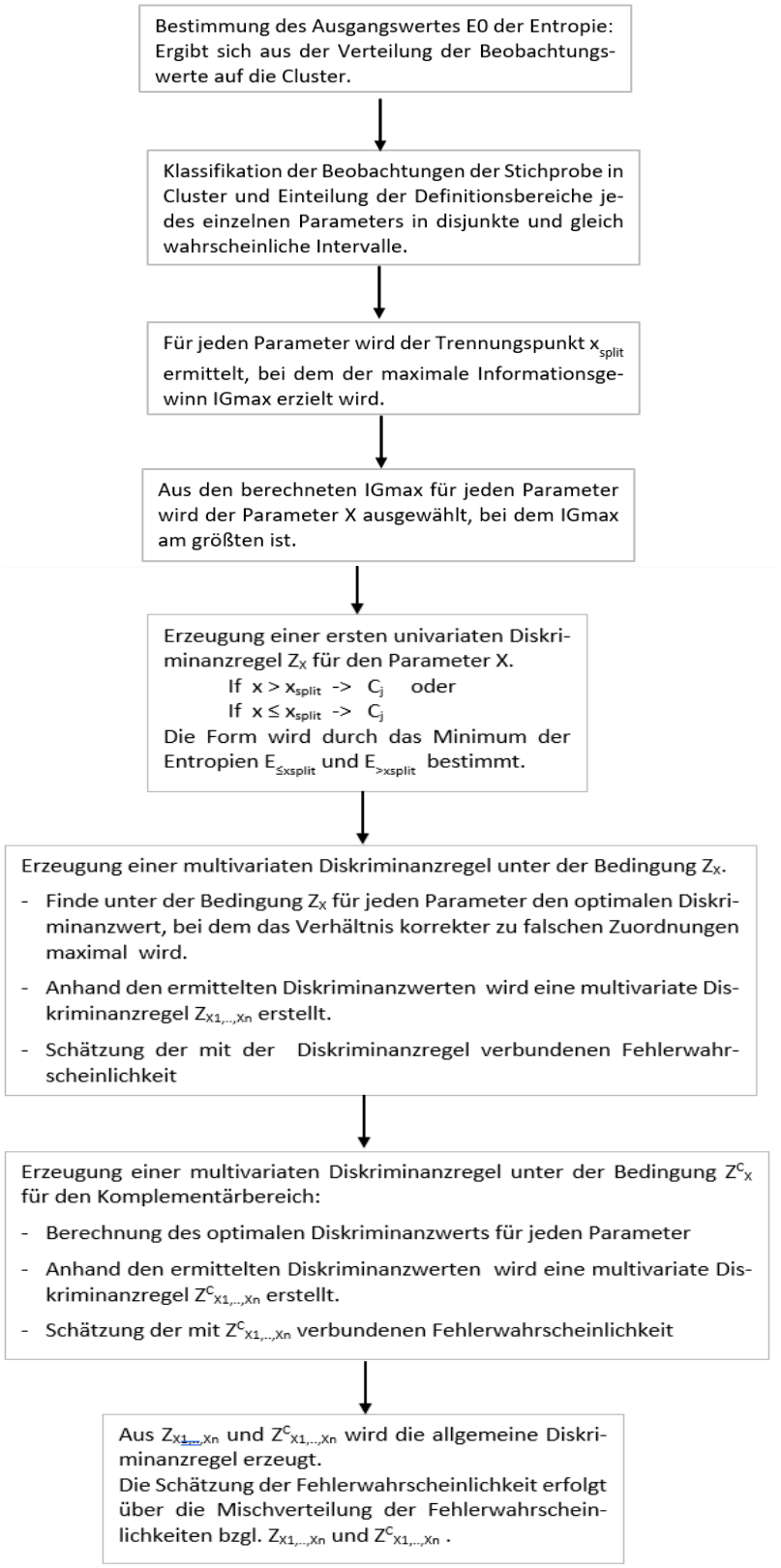


Abb. 2.11 Ablaufdiagramm der Methode ASPIC

2.5 Methode zur Identifikation von Primimplikanten aus Ergebnissen einer IDPSA mit MCDET

Primimplikanten sind ein Schlüsselkonzept der klassischen PSA. In einer PSA ist ein Primimplikant eines Szenarios das minimale Set an booleschen Bedingungen, das erfüllt sein muss, damit es zu diesem Szenario kommt. Diese Idee kann für eine IDPSA erweitert werden. In diesem Fall ist ein Primimplikant nicht länger ein boolesches Set von Bedingungen, sondern das minimale Set von diskreten charakteristischen Eigenschaften, die zu dem speziellen Szenario führen. Diese charakteristischen Eigenschaften können sowohl Wertebereiche für den Zeitpunkt von Ereignissen und Prozessparametern als auch diskrete System- und Komponentenzustände sein. Primimplikanten vereinfachen dadurch die Ergebnisse einer dynamischen PSA, so dass sie für einen aus dem Bereich der PSA kommenden Nutzer leichter verständlich sind. Für eine Analyse müssen die gesuchten Eigenschaften aus den von MCDET gespeicherten Zeitreihen extrahiert werden. Dies Methode wurde exemplarisch für eine im Jahr 2018 mit MCDET durchgeführte Analyse eines DEHEIRO (/PES 18/) angewandt. Im Folgenden soll zuerst die Methode vorgestellt und dann an einem Anwendungsbeispiel verdeutlicht werden.

2.5.1 Extraktion der relevanten Zeitreiheninformation

MCDET behandelt kontinuierliche aleatorische und epistemische Unsicherheiten durch eine Doppelschleife an ausgespielten Werten, die über ein Monte-Carlo-Verfahren gezogen werden. Das Resultat einer MCDET-Analyse sind hierarchisch strukturierte Zeitserien für jeden erzeugten Ereignisbaum. Im vorliegenden Vorhaben wurden die Möglichkeiten für das Berechnen von MCDET-Ergebnisdaten erweitert. Es ist nun möglich, die interessierenden diskreten Werte zu Beginn der Analyse zu definieren und dann über eine vordefinierte Funktionsgruppe in ein in der Datenanalyse gebräuchliches, handlicheres Tabellenformat (Python pandas Dataframe) zu extrahieren. Zu den üblicherweise extrahierten Eigenschaften gehören

- die Wahrscheinlichkeit einer Zeitreihe,
- die Dauer einer Zeitreihe,
- die Werte von beobachteten Variablen zu bestimmten Zeitpunkten im betrachteten Prozess,

- der erste und letzte Zeitpunkt, in dem der Wert einer Variablen eine definierte Schwelle über oder unterschreitet und
- der Wert von ausgespielten aleatorischen und epistemischen Unsicherheiten.

Die auf diese Weise extrahierten Werte können über verschiedene statistische Methoden, wie Korrelationskoeffizienten aber auch Methoden des maschinellen Lernens und diverse Visualisierungen analysiert werden.

2.5.2 Notwendigkeit zur Diskretisierung der extrahierten Variablen

Im Gegensatz zu verschiedenen anderen maschinellen Klassifikationsalgorithmen braucht eine Primimplikantenanalyse diskretisierte Werte. Frühere Studien mit Methoden der dynamischen PSA /DIM 15a/, /DIM 15b/ haben gezeigt, wie eine Primimplikantenanalyse erweitert werden kann, um kompatibel mit der mehrwertigen Information einer diskreten dynamischen PSA zu sein. In den in diesen Studien durchgeführten diskreten dynamischen PSA gab es nur einen Ereignisbaum, d. h. es wurden nicht Sets an kontinuierlichen aleatorischen und epistemischen Werten mittels Monte-Carlo-Verfahren ausgespielt. Alle kontinuierlichen Werte für aleatorische Unsicherheiten, wie z. B. der Zeitpunkt von Ereignissen, wurden bereits im Vorfeld diskretisiert. Der große Unterschied zwischen diesen früheren Studien und den hier präsentierten Studien mit MCDET liegt also darin, dass es für die Analyse einer MCDET-Studie nötig sein kann, die extrahierten Variablenwerte zu diskretisieren.

Im vorliegenden Vorhaben wurde ein maschineller Lernalgorithmus entwickelt, um eine geschickte Diskretisierung der extrahierten Variablen zu finden. Mit geschickt ist hier gemeint, dass die Wertebereiche für kontinuierliche Variablen so gewählt werden sollen, dass nur anhand der Zugehörigkeit zu den verschiedenen Wertebereichen eine möglichst gute Unterscheidung zwischen Zeitserien, die zum gesuchten Szenario führen, und anderen Zeitserien gemacht werden kann.

2.5.3 Einsatz von Entscheidungsbäumen und Random Forest Klassifikatoren für die Variablen Diskretisierung

Entscheidungsbäume sind eine überwachte Lernmethode für Klassifikation und Regression. Entscheidungsbäume bestehen aus sogenannten Knoten und Zweigen ausgehend von einem Basisknoten. An jedem Knoten wird eine Bedingung basierend auf den Eingangsvariablen ausgewertet. Bei einem binären Entscheidungsbaum wird ausgehend

von dieser Bedingung der eingehende Datensatz in zwei Teile geteilt. An jedem Knotenpunkt wird die Variable und der Schwellwert für diese Variable gesucht, die ein vorher bestimmtes Kriterium optimieren. Dieses Kriterium kann das Gini-Impurity-Maß in einem Knoten m sein:

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

mit

$$p_{mk} = \frac{1}{n_m} \sum_{y \epsilon Q_m} I(y = k)$$

p_{mk} ist hier der Anteil an Datenpunkten in Klasse k die im Knoten m liegen, oder die Entropie des Knotens:

$$H(Q_m) = -\sum_k p_{mk} \log(p_{mk})$$

In der vorliegenden Studie wurde zum Training für das maschinelle Lernen für einen Teil des Gesamtdatensatzes eine binäre Klassifizierung angewandt. Diese Klassifizierung wurde basierend auf der Frage durchgeführt, ob der Datenpunkt zum gesuchten Endzustand führt oder nicht. Die auf diese Weise für jede Variable gefundenen Schwellwerte eignen sich damit für die gewünschte Variablendiskretisierung. Dabei wird der Ereignisbaum auf dem Trainingsteil des vorhandenen Datensatzes trainiert und auf dem Rest getestet. Es ist allerdings zu beachten, dass Entscheidungsbaumalgorithmen vor allem bei vielen eingehenden Variablen zum sog. Overfitting neigen. Das heißt der entstandene Baum liefert eine sehr gute Klassifizierung für den zum Training verwandten Datensatz, aber eine relativ schlechte auf den Kontrolldatensatz. Die so gefundenen Schwellwerte wären damit stark abhängig von der Wahl des Trainingsdatensatzes. Um diese Problematik zu verringern, wird nicht nur ein Baum trainiert, sondern mit verschiedenen zufallsgezogenen Trainingsdatensätzen basierend auf dem Ursprungstrainingsdatensatz ein Wald von Entscheidungsbäumen (Random-Forest).

Als erster Schritt werden die extrahierten Variablen in kontinuierliche numerische Variablen und in kategorische Variablen unterschieden. Der genutzte Algorithmus, der Decision-Tree-Algorithmus in der scikit-learn Python-Bibliothek, kann nur mit

kontinuierlichen numerischen Variablen arbeiten. Die extrahierten kategorischen Variablen werden später zur Anwendung des Primimplikantenalgorithmus wieder hinzugefügt. Als nächstes wird ein Random-Forest-Algorithmus genutzt, um die Variablen auszuwählen, die zur besten Separation anhand eines der beiden oben definierten Kriterien führt. Das reduziert zum einen die Zahl der Variablen, die in einem Entscheidungsbaum genutzt werden können, und verringert so die Gefahr von Overfitting. Zum anderen sorgt es dafür, dass die wesentlichen Variablen auch möglichst in allen Bäumen betrachtet werden. Der Ablauf der Variablendiskretisierung ist in Abb. 2.12 dargestellt. Die Implementation dieses Diskretisierungsalgorithmus folgt der Python scikit-learn API und sollte so auch für Außenstehende einfach zu verwenden sein. Die Daten werden im gebräuchlichen Python-numpy-Format eingegeben und sind so nicht abhängig von der MCDET internen Darstellung der Zeitserien. Als Trainingsdaten dient ein zufällig ausgespielter Teil des Gesamtdatensatz. Wie bei einem klassischen scikit-learn-Transformer erhält der Nutzer die Möglichkeit, die gelernte Transformierung, also die Diskretisierung, über den Befehl transform auf neue Daten anzuwenden. Auch kann sich der Nutzer die gelernten Intervallgrenzen ausgeben lassen.

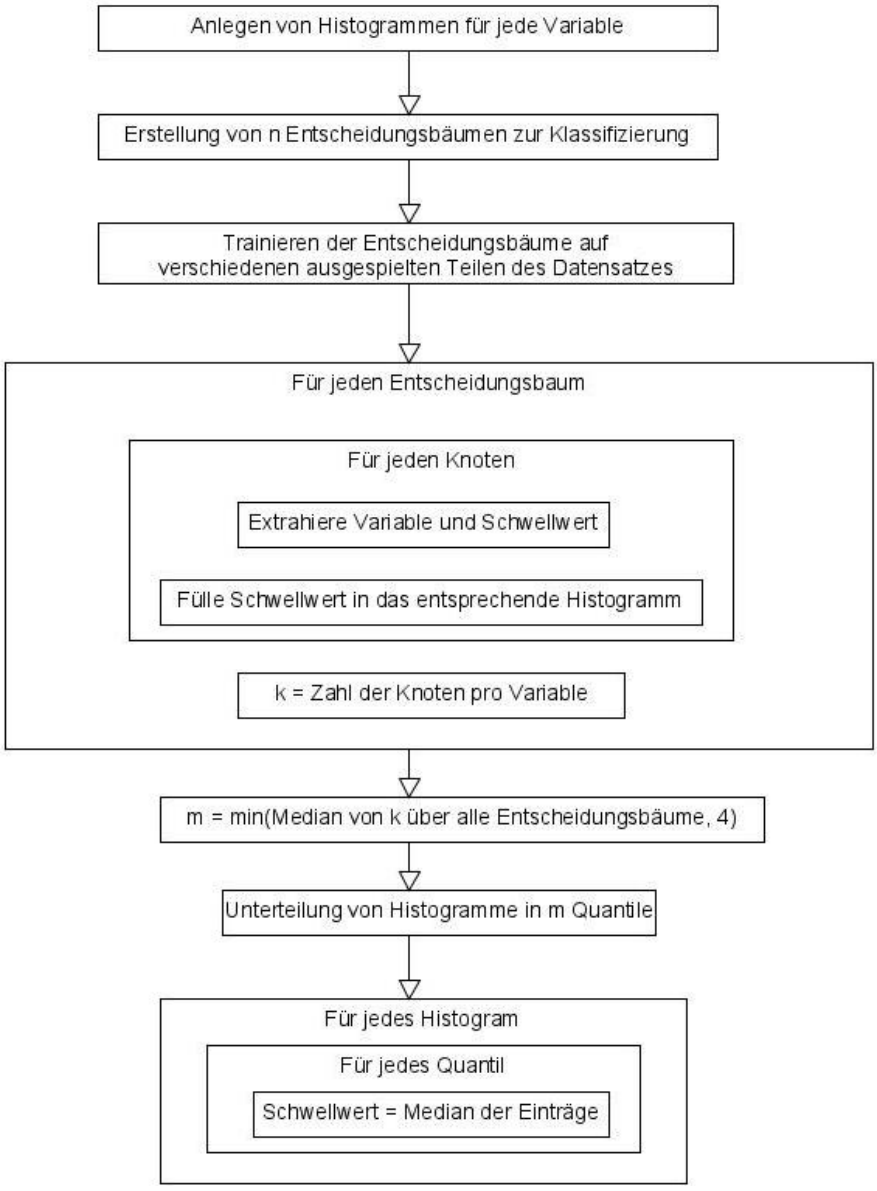


Abb. 2.12 Ablauf der Variablendiskretisierung und Ermittlung der verschiedenen Schwellwerte

2.5.4 Erweiterter Primimplikantenalgorithmus

Ergebnis des vorherigen Arbeitsschrittes ist eine Reihe von Implikanten für den gesuchten Prozesszustand. Diese Implikanten setzen sich aus den Werten für die kategorischen und die diskretisierten numerischen Variablen für die einzelnen gespeicherten Zeitserien zusammen. In diesem Abschnitt soll nun ein erweiterter Primimplikantenalgorithmus vorgestellt werden, der genutzt werden kann, um aus diesen Implikanten die Primimplikanten zu extrahieren. Der eingesetzte Primimplikantenalgorithmus ist nicht auf Performanz hin optimiert und dient nur zur Überprüfung des prinzipiellen Arbeitsablaufs. Das ganze Verfahren ist so modular aufgebaut, dass der Algorithmus in späteren

Weiterentwicklungen optional ausgetauscht werden kann. Der erweiterte Primimplikantenalgorithmus basiert auf dem Quine-McCluskey-Vereinfachungs- und Konsensalgorithmus /OGU 81/. Im Quine-McCluskey-Algorithmus gibt es die drei Schritte

- Absorption,
- Vereinigung und
- Reduktion.

Alle gefundenen Implikanten können Teil einer Entscheidungstabelle sein. Die Länge der Implikanten kann definiert werden als die Zahl der Eigenschaften, die für diesen Implikant festgelegt sein muss. Eigenschaften in einem Implikant, die nicht signifikant für den Zielwert sind, werden auf ‚don't care' gesetzt.

Absorption bedeutet dann, dass wenn zwei Implikanten in allen festgelegten Eigenschaften übereinstimmen, der längere von beiden im kürzeren absorbiert werden kann. Dies ist exemplarisch dargestellt in Abb. 2.13.

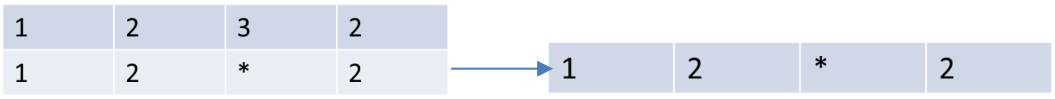


Abb. 2.13 Exemplarische Darstellung einer Absorption (Der Stern * steht für einen „don't care" Wert)

Vereinigung bedeutet für mehrwertige Variablen folgendes: Wenn n Implikanten sich bis auf eine Eigenschaft gleichen und diese Eigenschaft in allen Varianten vorkommt, dann können diese Implikanten zu einem Implikant vereinigt werden und der Wert für die Eigenschaft, die in allen Varianten vorkam, ist bei dem neuen Implikant auf ‚don't care' gesetzt. Der neue Implikant ist also um eine Eigenschaft kürzer als die Ursprungsimplikanten. Dies ist exemplarisch dargestellt in Abb. 2.14.

Abb. 2.14 Exemplarische Darstellung einer Vereinigung (Der Stern * steht für einen „don't care" Wert).

Reduktion bedeutet für mehrwertige Variablen folgendes: Wenn n Implikanten unterschiedlicher Länge sich bis auf eine Eigenschaft gleichen und diese Eigenschaft in allen Varianten vorkommt, so wird der längste Implikant ersetzt durch einen Implikanten, bei dem die Eigenschaft auf ‚don't care' gesetzt ist. Dies ist exemplarisch dargestellt in Abb. 2.15.

Dieser Algorithmus wurde implementiert und seine Anwendung in einem Jupyter Notebook exemplarisch vorgeführt.

Abb. 2.15 Exemplarische Darstellung einer Reduktion (Der Stern * steht für einen „don't care" Wert)

2.5.5 Interaktive Analyse der extrahierten Implikanten und Primimplikanten

Zusätzlich zum oben dargestellten Algorithmus wird dem Nutzer in einem Jupyter Notebook eine visuell interaktive Analysemöglichkeit vorgestellt. Dabei wird für jeden Implikant und Primimplikant ein Kostenwert berechnet. Dieser Kostenwert ist die Anzahl der Eigenschaften, die nicht auf einen ‚don't care' Wert gesetzt ist. Die diskretisierten Eigenschaftswerte sowie auch der Kostenwert werden für die ursprünglichen Implikanten und die gefundenen Primimplikanten in einem Parallelkoordinatendiagramm dargestellt. Hierzu werden die Python-Bibliotheken Hiplot und Plotly verwandt. Diese Darstellungsformate erlauben interaktives Selektieren und Deselektieren von Implikanten und Primimplikanten. Dadurch kann der Primimplikantenalgorithmus vom Nutzer interaktiv nachempfunden werden. Der folgende Algorithmus kann verwandt werden:

1. Auswählen der Implikanten mit den niedrigsten Kostenwerten. Diese sind automatisch auch Primimplikanten.
2. Entfernen dieser Primimplikanten und aller Implikanten, die von ihnen abgedeckt werden, aus dem Diagramm.
3. Wiederholung dieses Prozesses, bis alle Implikanten aus dem Diagramm entfernt wurden.

2.5.6 Einbindung der gefundenen Implikanten in eine klassische PSA

Es wurde ein Konzept erarbeitet, wie über die Nutzung des erweiterten GRS Software Werkzeuges pyRiskRobot /BER 16/, /BER 17/ die gefundenen Implikanten automatisiert in eine klassische PSA integriert werden können. Die gefundenen diskretisierten Eigenschaften werden dabei in die binäre Logik einer klassischen PSA übersetzt. Folgender Ablauf wird verwandt:

1. Die Eigenschaften werden nach ihrer in Abschnitt 2.5.3 bestimmten Wichtigkeit sortiert.

2. Für jede diskretisierte Eigenschaft mit n möglichen Werten werden n klassische PSA-Ereignisse erzeugt. Jedem Ereignis wird ein Schwellwert zugeordnet. Das Ereignis ist wahr, wenn der beobachtete Wert für diese Eigenschaft über dem Schwellwert liegt. Durch die Verknüpfung aller Ereignisse und gefundenen Ausgangsmöglichkeiten ergibt sich ein klassischer Ereignisbaum.

   Vorteil der automatisierten Übertragung der Ergebnisse einer dynamisch probabilistischen Sicherheitsanalyse in eine klassische Sicherheitsanalyse ist die Möglichkeit, sich zum einen die Primimplikanten über die in den klassischen Sicherheitsanalysen optimierten Methoden ausrechnen zu lassen, zum anderen eröffnet es die Möglichkeit die Ergebnisse der dynamischen Sicherheitsanalyse zielgerichtet in eine klassische Analyse einzubinden. Dies entspricht dem Grundgedanken der dynamischen Sicherheitsanalyse. Sie ist nicht Ersatz für die klassische Analyse, sondern erweitert diese an Punkten, an denen eine dynamische Betrachtung relevant ist.

2.5.7 Anwendung der Primimplikanten Methode auf eine MCDET DEHEIRO Analyse

Der ursprünglich durchgeführten MCDET-Analyse eines thermisch bedingten Dampferzeuger-Heizrohrlecks (DEHEIRO) /PES 18/ lag zusammengefasst folgendes Szenario zugrunde:

Ein totaler lang andauernder Stromausfall im Kraftwerk führte zu einer Hochdrucksituation mit eventuell folgender Kernschmelze. Der entstehende Druckunterschied zwischen primärem und sekundärem Kühlkreis zusammen mit den hohen Temperaturen begünstigen dann ein thermisch bedingtes DEHEIRO. Wichtige aleatorische Unsicherheiten, die in der durchgeführten dynamischen PSA berücksichtigt wurden, betreffen den

Vorschaden des Dampferzeugerheizrohres und die zyklischen Anforderungen des Druckhalter-Abblaseventils und der beiden Sicherheitsventile.

Für das Beispiel dieses DEHEIRO-Falls wurden die erzeugten Sequenzen anhand des Zeitpunktes klassifiziert, wann ein Heizrohrleck aufgetreten ist. So wurden Sequenzen, in denen das Heizrohrleck in weniger als 50 Minuten nach der Durchführung der Notfallmaßnahme ‚Sekundärseitiges Druckentlasten' eintritt, als ‚Fehlerzustand' klassifiziert. Alle anderen Sequenzen wurden als ‚kein Fehlerzustand' klassifiziert.

Folgende Variablen wurden als geeignet identifiziert, um diese Einstufung ohne Kenntnis der Zielgröße durchzuführen:

- Zeitpunkt des ersten Öffnens der Druckhalter (DH)-Sicherheitsventile 1 und 2
- Zeitpunkt des ersten Öffnens des DH-Abblaseventils
- Letzter Zeitpunkt an dem sich der Zustand der Sicherheitsventile 1 und 2 ändert
- Letzter Zeitpunkt an dem sich der Zustand des DH-Abblaseventils ändert
- Finaler Öffnungsquerschnitt der Sicherheitsventile 1 und 2
- Finaler Öffnungsquerschnitt des DH-Abblaseventils
- Reihenfolge in denen die letzten erfolgreichen Zustandsänderungen der drei obengenannten Ventile durchgeführt wurden
- Vorschaden des Dampferzeugerheizrohrs
- Zeitpunkt der Druckentlastung

Hierbei wurden die finalen Öffnungsquerschnitte mitberücksichtigt, da an ihnen zu erkennen ist, ob die Ventile in geöffnetem oder geschlossenem Zustand ausfallen.

Für dieses Beispiel wurde die oben beschriebene mit maschinellem Lernen unterstützte Diskretisierung durchgeführt.

In Abb. 2.16 sind die Zeiten für das letzte erfolgreichen Öffnen/Schließen von Sicherheitsventil 1 (startfinalcond_sv1_state x-Achse im Bild) und 2 (startfinalcond_sv2_state y-Achse im Bild) dargestellt, einmal für Sequenzen, die zu einem Fehlerzustand führen (orangene Punkte) und einmal für solche, die nicht zu einem Fehlerzustand führen (blaue Punkte). Ebenso sind die zugehörigen Intervallgrenzen in blau eingezeichnet. Die

gefundenen Intervallgrenzen stellen einen Kompromiss dar zwischen einer möglichst guten Trennung der erzeugten Sequenzen basierend auf der Klasse der Zielvariablen und einer einfachen Interpretierbarkeit der Grenzen im Sinne einer klassischen Primimplikantenanalyse.

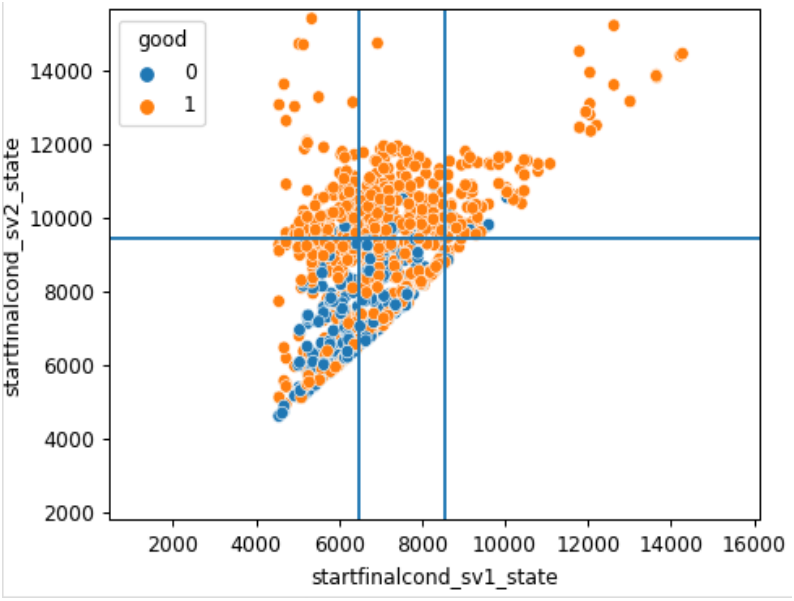


Abb. 2.16 Zeiten für das letzte erfolgreichen Öffnen/Schließen von Sicherheitsventil 1 und 2 und die zugehörigen Intervallgrenzen

Die gefundenen Schwellwerte für die verschiedenen Variablen sind in Tab. 2.15 aufgelistet.

Tab. 2.15 Schwellwerte die bei der Diskretisierung kontinuierlicher Variablen im DE-HEIRO-Beispiel genutzt wurden

Variable	Schwellwerte
Zeitpunkt des ersten Öffnens des Sicherheitsventils 1 (s)	6 740
Zeitpunkt des ersten Öffnens des Sicherheitsventils 2 (s)	6 870, 12 850
Zeitpunkt des ersten Öffnens des Dampfabblaseventils (s)	5 180
Zeitpunkt des letzten erfolgreichen Schaltens des Sicherheitsventils 1 (s)	6 490, 8 510
Zeitpunkt des letzten erfolgreichen Schaltens des Sicherheitsventils 2 (s)	310, 9 450
Zeitpunkt des letzten erfolgreichen Schaltens des Dampfabblaseventils (s)	6 050
Finaler Öffnungsquerschnitt des Sicherheitsventils 1 (%)	94

Variable	Schwellwerte
Finaler Öffnungsquerschnitt des Sicherheitsventils 2 (%)	86
Finaler Öffnungsquerschnitt des Abblaseventils (%)	91, 96
Vorschaden Dampferzeugerheizrohr	27, 47
Zeit Druckentlastung (s)	5 250

Basierend auf den so gefundenen diskreten Variablen wurde der oben beschriebene erweiterte Primimplikantenalgorithmus durchgeführt.

Die ursprünglichen Implikanten sowie die gefundenen Primimplikanten sind in Abb. 2.17 in einem Parallelkoordinatendiagramm dargestellt. Zur Veranschaulichung wurden einige Sequenzen selektiert. Die Farbe der selektierten Sequenzen wird dabei durch den Kostenwert bestimmt. Weitere Details zur Anwendung der Primimplikantenmethode für dieses DEHEIRO-Beispiel finden sich unter /ERA 22/.

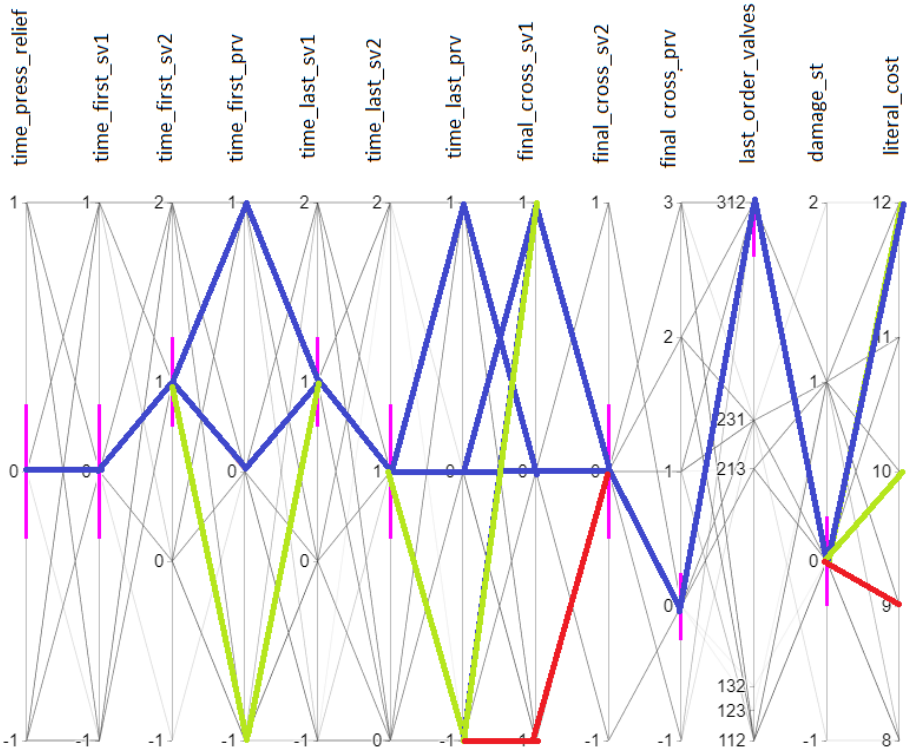


Abb. 2.17 Parallelkoordinatendiagram für die verschiedenen in /ERA 22// untersuchten diskretisierten Variablen der mit MCDET gefundenen Zeitserien (Gezeigt sind nur Sets von Variablen (Implikanten), die zum gesuchten Ereignis führen.)

3 Weiterentwicklung des Scheduling-Systems

3.1 Überarbeitung und technische Weiterentwicklung des MCDET-Kerns

3.1.1 Vermeidung von Zwillingspfaden

Es kann vorkommen, dass zwei oder mehrere unterschiedliche Trigger dieselbe Zustandsänderung bei Größen eines Rechencodes auslösen. Als Beispiel soll ein Ventil betrachtet werden, das in Abhängigkeit vom Füllstand in einem Tank entweder offen oder geschlossen ist. Wenn der Füllstand zu niedrig ist, wird das offene Ventil durch die Regelung automatisch geschlossen. Wenn der Füllstand zu hoch ist, wird das geschlossene Ventil automatisch geöffnet. Bzgl. des Ausfallverhaltens kann man z. B. annehmen, dass das Ventil mit der Wahrscheinlichkeit p1 aufgrund eines Fehlsignals offen (geschlossen) bleibt, wenn es schließen (öffnen) soll. Weiter kann man annehmen, dass aufgrund von Verschleiß wegen häufigem Öffnen und Schließen das Ventil mit der Wahrscheinlichkeit p2 offen (geschlossen) bleibt, wenn es eigentlich schließen (öffnen) soll. D. h. man hat z. B. bzgl. eines Ventilausfalls in Offen-Stellung zwei unterschiedliche Trigger. Trigger 1 tritt ein, wenn durch die Regelung das Ventil von offen auf zu gesetzt wird. Trigger 2 tritt ein, wenn das Ventil das n-te Mal angefordert wird zu schließen, wobei n zufällig aus einer geeigneten Verteilung (z. B. geometrische Verteilung) ausgespielt wird. Jeder der beiden Trigger bewirkt, dass vom aktuellen Simulationspfad, in dem das Ventil geschlossen wird, ein neuer Simulationspfad abzweigt, in dem das Ventil offenbleibt. Bei Trigger 1 hat der neue Simulationspfad die Eintrittswahrscheinlichkeit p1, bei Trigger 2 die Eintrittswahrscheinlichkeit p2. Treten beide Trigger gleichzeitig ein, würden zwei identische Simulationspfade (Zwillingspfade) gerechnet werden, weil dieselbe Zustandsänderung zum selben Zeitpunkt erfolgt. Dies ist überflüssig und kostet zudem viel Rechenzeit.

Um die Generierung von Zwillingspfaden zu vermeiden, wird jedes Mal, bevor ein neuer Simulationspfad angelegt wird, überprüft, ob zu diesem Zeitpunkt bereits ein identischer Simulationspfad mit derselben Zustandsänderung angelegt wurde. Wenn ja, wird der neue Simulationspfad nicht angelegt und seine Wahrscheinlichkeit auf die des bereits angelegten Simulationspfads addiert.

3.1.2 Zustandsänderungen für eine Gruppe von Rechencode-Größen

Bisher war es in den Transition-Fenstern von MCDET nur möglich, für jede Rechencode-Größe einzeln die alternativen Zustände (Werte) zu einem bestimmten Zustand und die zugehörigen Eintrittswahrscheinlichkeiten einzugeben. MCDET hat dann bei den Fällen, bei denen alternative Zustände für mehrere Rechencode-Größen angegeben wurden, alle möglichen Zustandskombinationen für diese Größen ermittelt, die zugehörigen Simulationspfade gestartet und diese mit den Eintrittswahrscheinlichkeiten der jeweiligen Kombinationen bewertet (siehe folgendes Beispiel 1).

Wenn für mehrere Rechencode-Größen nicht alle, sondern nur bestimmte Kombinationen von Zuständen als Alternativen möglich waren, mussten aufwändige und manchmal auch komplizierte Eingaben gemacht werden, um das gewünschte Ergebnis zu erhalten. Deshalb wurden die programmtechnischen Voraussetzungen geschaffen, die Eingabe für diese Fälle zu vereinfachen. Die neue MCDET-Version erlaubt es nun, eine Gruppe von Rechencode-Größen zusammen mit den alternativen Zustandskombinationen und deren Wahrscheinlichkeiten zu spezifizieren. MCDET generiert dann nur die Simulationspfade mit den spezifizierten alternativen Zustandskombinationen. Die Bewertung erfolgt mit den angegebenen Wahrscheinlichkeiten (siehe folgendes Beispiel 2).

Beispiel 1

Trigger:

- K1 = 0, K2 = 0,

Zustandsänderungen:

- K1 = 1 mit Wahrscheinlichkeit p1
- K2 = 1 mit Wahrscheinlichkeit p2

Aufgrund dieser Informationen legt MCDET drei neue Simulationspfade mit den folgenden Anfangszuständen und Wahrscheinlichkeiten an:

- K1 = 1, K2 = 0 mit Wahrscheinlichkeit p1*(1.-p2)
- K1 = 0, K2 = 1 mit Wahrscheinlichkeit (1.-p1)*p2
- K1 = 1, K2 = 1 mit Wahrscheinlichkeit p1*p2

Der aktuelle Simulationspfad mit K1 = 0, K2 = 0 wird mit der Wahrscheinlichkeit (1.-p1)*(1.-p2) bewertet.

Beispiel 2

Trigger:

- K1 = 0, K2 = 0

Zustandsänderungen:

- (K1 = 1, K2 = 1) mit Wahrscheinlichkeit p3

Aufgrund dieser Informationen legt MCDET nur einen neuen Simulationspfad mit dem Anfangszustand (K1 = 1, K2 = 1) an. Dieser wird mit der Wahrscheinlichkeit p3 bewertet. Der aktuelle Simulationspfad mit (K1 = 0, K2 = 0) wird mit der Wahrscheinlichkeit (1.-p3) bewertet.

3.1.3 Erweiterung für Simulationen mit unsicheren Startbedingungen

Die frühere auf serielle Abarbeitung ausgelegte Version des MCDET-Systems führte die Zustandsveränderungen in abgezweigten Simulationspfaden über eine automatisierte Modifikation der Rechencode-Eingabedateien durch. Diese Art von Zustandsanpassung war sehr spezifisch auf ATHLET-basierte Simulationen ausgelegt und wurde in den generisch einsetzbaren Treiberschnittstellen des Scheduling-Systems durch den Zugriff auf den Simulationszustand ersetzt. Die Eingabedateien des Rechencodes konnten so innerhalb eines Scheduler-Laufs als unveränderlich behandelt werden, was die Verwendung schreibgeschützter Rechencode-Submodelle und Parallelzugriffe durch gleichzeitig laufende Simulationsprozesse erlaubt.

Allerdings ergeben sich hieraus Einschränkungen bei der Modellierung spezieller Störfallszenarien, welche die Berücksichtigung potenziell unsicherer Startbedingungen (Epistemik) und Größen, wie z. B. physikalische Modellparameter erfordert. Diese werden üblicherweise in den Eingabedateien des Rechencodes fest vorgegeben und haben oft Einfluss auf das Verhalten (Kennlinie) von Modell-Komponenten wie Pumpen und Ventilen. Da aber im Simulationszustand meist keine zentral justierbare Größe existiert, können diese Parameter zwangsläufig nur über die Modifikation der Eingabedaten angepasst werden.

Ziel dieses Arbeitspunktes war die Nachrüstung der generischen Treiberschnittstelle, um auch Störfallszenarien mit epistemischen Unsicherheiten zu unterstützen, welche veränderliche Eingabedateien erfordern. Der im Berichtszeitraum erreichte Entwicklungsstand erlaubt zum einen die Verwendung automatisiert variierter Rechencode-Eingabedateien und zum anderen diese über Indizes identifiziert, parallelen Scheduler-Läufen zuzuordnen. Hiermit können die oben beschriebenen Simulationsvariationen bereits in allen

Startpunkten epistemischer Läufe durchgeführt werden. Der nächste Schritt wäre, entsprechende Anpassungen auch generisch in Abzweigpunkten zu erlauben, was jedoch einiger Änderungen im Treiberteil zur Abspaltung von Simulationen bedarf. Diese müssen auch mit der inzwischen parallelisierten Arbeitsweise von MCDET eine sichere Nebenläufigkeit der Simulationen gewährleisten. Die dafür notwendigen Anpassungen wurden ermittelt und ein Konzept für deren Implementierung erstellt. Dieses sieht die Erweiterung der Abspaltungsprozedur um eine automatisierte Anpassung von Eingabedateien und ein generisches Interface zur Änderungsspezifikation vor. Die weitere Ausarbeitung, die Implementierung in den jeweiligen Simulationstreibern und deren Integration in die Hauptentwicklungslinie könnten zu einem späteren Zeitpunkt umgesetzt werden.

3.1.4 Konzept zur Kopplung von MCDET mit verschiedenen Rechenprogrammen inklusive „closed source“-Rechencodes

Bei der Entwicklung der im aktuellen MCDET-Scheduler enthaltenen Schnittstellen zur Kopplung von Rechenprogrammen wurde sehr auf Modularität und generische Einsetzbarkeit geachtet. Hierdurch wird bereits eine sehr robuste Nutzbarkeit verschiedener Rechencodes möglich, selbst wenn diese nicht für die Anwendung mit MCDET konzipiert wurden oder speziell dafür gewartet werden. Durch die generische Schnittstelle FDE (Fortran Development Extensions) /FDE 22/ zur Ansteuerung und zum Datenaustausch ist die Anpassung neu einzubindender Rechencodes minimal, was anhand des Crew-Moduls und eines Rechenprogramms für ein einfaches Tanksystem exemplarisch gezeigt wurde. Durch die in der Schnittstelle vorbereiteten Methoden zum Datenzugriff und der Ablauf-Unterbrechung durch Callbacks können die zur Koppelbarkeit von Codes erforderlichen Eigenschaften meist sehr einfach nachgerüstet werden:

- **Getaktete Simulation:** Das Anhalten der Simulation und Warten auf einen Tick-Befehl zur Berechnung des nächsten Zeitschritts.
- **Datenzugriff (lesend)** auf alle relevanten Variablen des Simulationszustands. Diese werden zur Bewertung des Simulationszustands durch den Estimator herangezogen.
- **Datenzugriff (schreibend)** auf alle zur Steuerung relevanten Variablen des Simulationszustands. Hierdurch kommt es zur Modifikation des Simulationszustands abgespaltener Subsequenzen.

- **Vorzeitige Beendigung** der Simulation. Diese wird von MCDET genutzt, um die Simulation gezielt zu beenden, weil z. B. ein als sicher angenommener Zustand erreicht oder die Cut-off-Wahrscheinlichkeit unterschritten wurde.

Darüber hinaus muss der Rechencode die folgende Funktionalität aufweisen, um Abspaltungen zu ermöglichen:

- **Speichern des aktuellen Simulationszustands** als Speicherpunkt (Restart)
- **Fortsetzung der Simulation** als neuer Prozess unter Verwendung eines bestimmten Speicherpunkts (Restart)

Bereits anhand dieser Anforderungsliste kann die Kompatibilität eines Rechencodes mit MCDET beurteilt werden. Sie erlaubt somit auch die Abschätzung des Entwicklungsaufwand für dessen Anpassung. Aber auch wenn diese Änderungen ohne viel Aufwand im Rechencode nachrüstbar sind, erfordern sie dennoch den Zugriff auf dessen Code und die Neuübersetzung der Binärdateien, was für „Closed-Source“-Codes nicht möglich ist. Eine wichtige Zielsetzung war deshalb die Erarbeitung alternativer Ansätze, mit denen die oben genannten Anforderungen auch ohne Änderungen am Rechencode ermöglicht oder zumindest angenähert werden können. Unter Verwendung dieser Ansätze sollte exemplarisch ein Treiber für den Rechencode FDS (Fire Dynamics Simulator) implementiert und dieser somit an MCDET angebunden werden. Auch sollte durch die exemplarische Implementierung die Priorität der einzelnen Anforderungen ersichtlich werden und sich herausstellen, unter welchen Umständen und ggf. mit welchen Einschränkungen diese angenähert werden können.

Der FDS-Rechencode ist ein international weit verbreiteter CFD-Code (computational fluid dynamic) zur Simulation von Bränden. In diesem werden numerische Gleichungen gelöst, um vor allem den Rauch und Hitzetransport durch Feuer zu modellieren /MCG 13/. FDS unterteilt den Raum in rechtwinklige Gitter und schätzt die Gleichungslösungen für diese Gitterstrukturen ab. Jede FDS-Simulation wird über eine einzelne textbasierte Eingabedatei kontrolliert und kann über die Kommandozeile gestartet werden. Eine wichtige Eingabe ist die vorgesehene Endzeit (TE) der Simulation. Durch entsprechende Konfiguration kann FDS an bestimmten Stellen Restart-Files schreiben, allerdings wird dieser Restart Mechanismus im FDS-Benutzerhandbuch selbst als fragil beschrieben. Darauf muss bei der Umsetzung ein besonderes Augenmerk gelegt werden. Es stellt sich also die Frage, ob FDS stabil genug ist für die Anwendung unter den Bedingungen des aktuellen MCDET-Schedulers.

Die im Folgenden beschriebenen Überlegungen beziehen sich auf die Version FDS 6, welche vor der Einführung des neuen Scheduler-Systems bereits erfolgreich mit MCDET gekoppelt wurde /PES 14/. Die notwendigen Ansätze zur Anbindung an den aktuellen MCDET-Scheduler werden anhand der oben genannten Anforderungen zur Koppelbarkeit von Rechencodes beschrieben.

- **Getaktete Simulation:** Diese wird durch FDS nicht unterstützt und muss daher über ein Steuerprogramm (Treiber) durch schrittweise Berechnung angenähert werden. Hierzu modifiziert das Steuerprogramm vor dem FDS-Aufruf die in der Eingabedatei TE, indem eine angenommene Schrittweite dt auf die TE addiert wird. Dieses Verfahren hat allerdings den Nachteil, dass die Simulation nicht mit der Genauigkeit von FDS-Zeitschritten überwacht und bewertet werden kann, d. h. die für MCDET verfügbare zeitliche Auflösung ist geringer als die FDS-interne Auflösung. Ohne weitere Vorkehrungen, z. B. durch Festlegung spezieller Regeln im Input, führt die feste Vorgabe von dt zu atomaren Zeitschritten, und es kann vorkommen, dass MCDET auf spontane Ereignisse in der Simulation nicht reagieren kann.

- **Datenzugriff (lesend):** Dieser wird durch FDS nicht unterstützt und muss über das Steuerprogramm implementiert werden, indem die Daten nach dem Abschluss eines Simulationstakts aus den FDS-Ausgabedateien gelesen werden. Die verfügbaren Rechencode-Größen beschränken sich somit auf die im Plotvektor vorhandenen Daten.

- **Datenzugriff (schreibend):** Dieser wird durch FDS nicht unterstützt und muss über das Steuerprogramm implementiert werden, indem die Zustandsänderungen über Modifikation der Eingabedatei oder der Restart-Datei für den nächsten Zeitschritt in die Simulation injiziert werden. Diese Art der Zustandsmodifikation ist als automatisiertes Verfahren nur schwer stabil implementierbar, denn in beiden Fällen müssen die Einfügepositionen in den Dateien zuverlässig erkannt werden, und Wertänderungen dürfen keine Format-inkompatiblen Verschiebungen zur Folge haben.

- **Vorzeitige Beendigung** der Simulation. FDS bietet die Möglichkeit eine Simulation durch Ablegen einer Stop-Datei im Arbeitsverzeichnis zu beenden. Dadurch hat das Steuerprogramm die Möglichkeit, eine Anforderung zum Beenden an die Simulation weiterzuleiten. Ein Nachteil dieses Vorgehens ergibt sich durch die Nutzung des Filesystems zur Prozess-Kommunikation (IPC). Durch die fehlende Synchronisierung wird die Reproduzierbarkeit beeinträchtigt, da nicht garantiert werden kann, wann die Stop-Datei von FDS erkannt wird. Alternativ kann die Beendigung dadurch realisiert

werden, indem das Steuerprogramm den Abschluss des aktuellen Simulationstakts abwartet und die Simulation einfach als beendet markiert.

- **Speichern des aktuellen Simulationszustands** wird von FDS durch entsprechende Konfiguration in der Eingabedatei unterstützt.
- **Fortsetzung der Simulation** als neuer Prozess wird von FDS unterstützt und bildet die Grundlage des oben beschriebenen Vorgehens zur Implementierung der getakteten Simulation.

3.1.5 Umstrukturierung des CrewModuls

Ziel der Umstrukturierung des klassischen CrewModuls, dessen Benutzerführung in Kapitel 5 im Detail beschrieben ist, war es, einen Python-basierten Rechencode für menschliche Handlungen zu schaffen, der den Anforderungen an einen mit MCDET koppelbaren Rechencode genügt (siehe Abschnitt 3.1.5.2) und dadurch mit dem MCDET Scheduler interagieren kann.

Zusätzlich sollte der eingesetzte Rechencode in der Lage sein, die bereits existierenden CrewModul Eingabedaten in Form von Mind-Map Files zu lesen und die nötige Information zu extrahieren. Darüber hinaus sollte es möglich sein, aus dem im Zusammenspiel mit MCDET produzierten Output die gleichen Analyseergebnisse zu extrahieren wie aus dem bisherigen CrewModul Output. Der Rechencode sollte sowohl eigenständig als auch in Kombination mit MCDET verwendet werden können. Um all diese Anforderungen zu erfüllen, wurde der bisher in Fortran gehaltene CrewModul Code neu erarbeitet und modular aufgesetzt. Dies erleichtert spätere Erweiterungen und Verbesserungen. Im Folgenden soll die Struktur des neuen CrewModuls beschrieben werden, für Grundüberlegungen zur Modellierung menschlicher Handlungen wird auf Kapitel 5 verwiesen.

3.1.5.1 MindMap Eingabeformat

Wie im vorigen Abschnitt erläutert, soll das CrewModul in der Lage sein, mit denselben Eingabedaten wie das klassische CrewModul zu arbeiten. Das Eingabeformat ist im Detail in Abschnitt 5.2 beschrieben. Es basiert auf einer MindMap Struktur, die mit der frei verfügbaren Software FreeMind erstellt wird. Das neue CrewModul wurde so aufgesetzt, dass es sowohl mit diesem Format umgehen kann als auch im Falle von Formatänderungen in der Eingabe leicht adaptierbar ist.

3.1.5.2 Grundgedanken des neuen CrewModuls

Um das neue CrewModul zu gestalten, war es nötig, einige Grundannahmen zu treffen, auf denen die weitere Ausgestaltung basiert:

- Eine Handlung verändert den Systemzustand erst, wenn sie finalisiert wurde, also am Ende der Handlungsdauer. Dies ist eine notwendige Zuweisung, da eine graduelle Handlungsdurchführung schwer zu implementieren ist und auch nicht unbedingt realistischer ist. Graduell durchzuführende Handlungen lassen sich durch eine feinere Aufteilung in Einzelhandlungen realisieren.
- Jede Handlung braucht Ressourcen (im alten CrewModul Auftragssender und Auftragsempfänger). Dies können entweder handelnde Personen oder aktive Systemkomponenten sein.
- Jede Ressource kann nur eine Handlung auf einmal durchführen.
- Es gibt nur eine Simulationszeit, diese schreitet nie zurück. Sie ist im Simulationszustand gespeichert.

Zentraler Kern des CrewModuls ist der Simulationszustand (state). Dieser Zustand beinhaltet sowohl den Zustand aller am CrewModul beteiligten Klassen wie auch den für das CrewModul relevanten Systemzustand und die Simulationszeit.

3.1.5.3 Klassenstruktur des neuen CrewModuls

Die Klassenstruktur des neuen CrewModul Codes ist in Abb. 3.1 dargestellt.

Einer der ersten Schritte eines Simulationslaufs ist das Auslesen der MindMap Eingabedatei, diese Aufgabe übernimmt der MindMapReader. Aus der MindMap Datei extrahiert der Reader die zugehörigen Handlungen und Handlungslisten sowie die zu einer Handlungsliste gehörigen Bedingungen. Aus den angegebenen Minima und Maxima für die Zeitverteilung bestimmt er die Dauer für jede Handlung als Mittelwert zwischen Minima und Maxima. Dies ist der Startwert für die Handlungsdauer. Des Weiteren werden die Startparameter für jeden Systemzustand, der zu einer Verzweigung führt, festgelegt, und zwar als niedrigster möglicher Wert für diesen Systemzustand.

Die Hauptklasse des neuen CrewModuls ist die CrewModul Klasse. Die CrewModul Klasse hat die Funktion, die Simulation zu starten, wenn nötig einen Restart durchzuführen und mit einer möglichen nächsten Handlung umzugehen. In der Startfunktion werden

über den MindMapReader die Handlungslisten und ihre Startparameter abgefragt. Es wird ein Objekt der ActionGetter Klasse angelegt, dem die Handlungslisten übergeben werden. Die abgefragten Startparameter werden genutzt, um den Simulationszustand zu aktualisieren. Dem ActionGetter wird dann die erste Handlungsliste übergeben. Basierend auf dem Simulationszustand wird vom ActionGetter die nächste mögliche auszuführende Handlung erfragt. Diese Handlung wird bewertet, und wenn sie als ausführbar befunden wird, wird sie ausgeführt und der Simulationszustand und damit auch die Simulationszeit entsprechend aktualisiert. Im eigenständigen Modus läuft der Rechencode, bis keine nächste Handlung mehr gefunden werden kann. Im interaktiven Modus läuft der Rechencode bis MCDET ein Stoppsignal sendet.

Die nächste zentrale Klasse ist die ActionGetter Klasse. Aufgabe der ActionGetter Klasse ist es, basierend auf dem Simulationszustand, eine mögliche nächste zu beendende Handlung zur Verfügung zu stellen. Zu diesem Zweck interagiert der ActionGetter mit den Handlungslisten und Handlungen. Er kann so z. B. Handlungen starten und beenden. Das Starten einer Handlung blockiert alle Ressourcen, die diese Handlung benötigt. Das Beenden befreit diese Ressourcen wieder. Der ActionGetter stellt zu beendende Handlungen in richtiger Reihenfolge zur Verfügung, d. h. mit fortschreitenden Stopzeitpunkten. Der ActionGetter entscheidet, basierend auf dem Simulationszustand und den Bedingungen einer Handlungsliste, welche Handlungsliste nach der letzten Handlung einer aktiven Handlungsliste folgen soll. Dies sind die Grundaufgaben des ActionGetters. Es sind mehrere mögliche Ausformungen eines ActionGetters denkbar, abhängig von Vorschriften, wie verschiedene Handlungen gewichtet und möglicherweise parallelisiert werden sollen. Der ActionGetter kann mit dem in Abschnitt 5.2.2 beschriebenen Eingabeformat umgehen und schlägt die gleichen nächsten Handlungen vor, wie das klassische CrewModul.

Die Klassen Action, ActionList, Condition und Ressource leiten sich aus der MindMap Eingabestruktur ab. So entspricht ein Objekt der Klasse ActionList einer Handlungsliste. Sie besteht aus zugeordneten Handlungen, einer ID, notwendigen Bedingungen und möglichen nächsten Handlungslisten. Ein Objekt der Klasse Action entspricht wiederum einer Handlung. Es hat eine ID, benötigt Ressourcen, führt zu einer Änderung des Simulationszustand und hat eine Dauer. Jede Bedingung für eine Handlungsliste besteht aus dem entsprechendem Variablennamen, einem Schwellwert und ein Abfrage Indikator, der Stimulus, der anzeigt, ob es sich um eine größer, kleiner, größer-gleich- oder kleiner-gleich-Abfrage handelt.

3.1.5.4 Restart Kapazität

Wie in Abschnitt 3.1.4 dargelegt, muss ein Rechencode fähig sein, einen Neustart durchzuführen und eine bestehende Simulation unter Verwendung eines Speicherpunkts fortzuführen, um mit MCDET optimal koppelbar zu sein. Im neuen CrewModul wird nach jeder Handlung eine Restart Datei im json-Format geschrieben. In dieser Datei sind alle nötigen Informationen gespeichert, um den Simulationszustand wiederherzustellen.

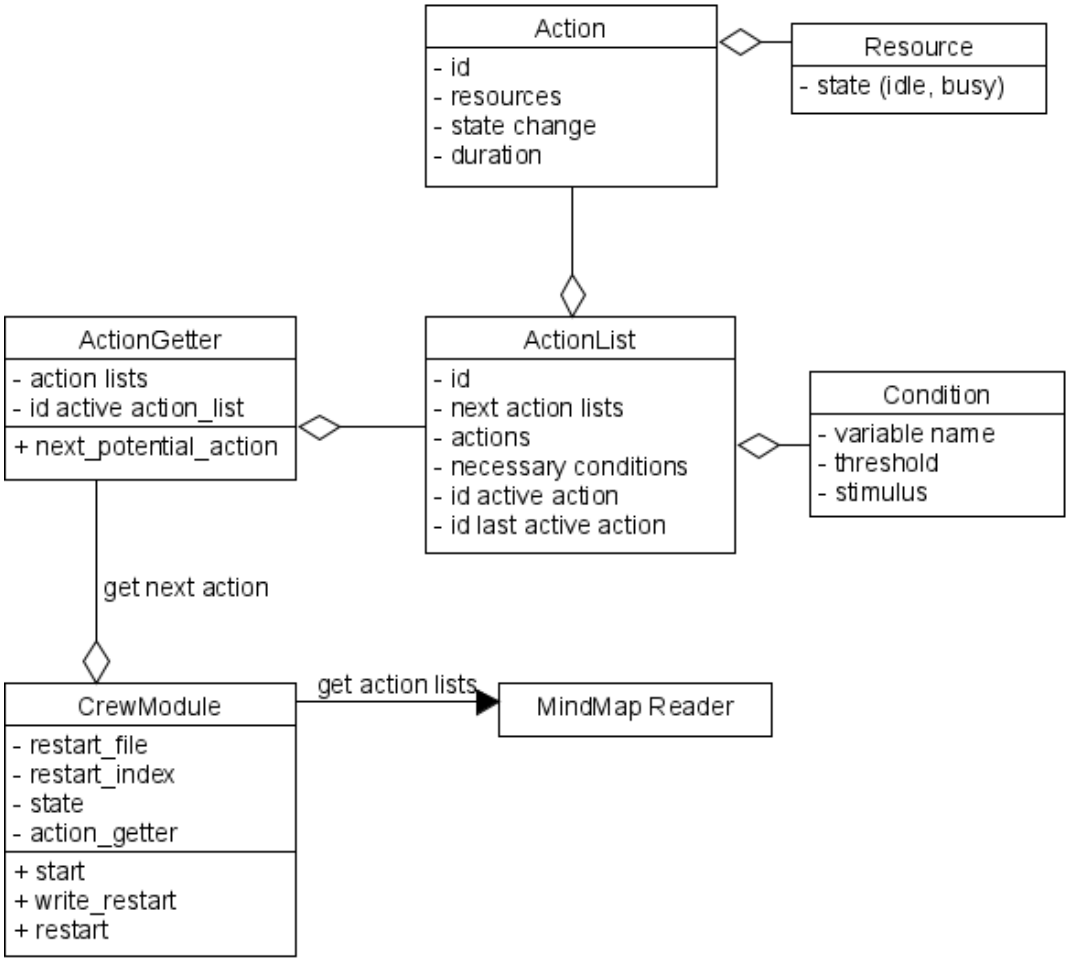


Abb. 3.1 UML-Klassendiagramm des neuen CrewModuls

3.1.5.5 Steuerung über MCDET

Die Ansteuerung des neuen CrewModuls folgt der im Bericht /PES 18/ Abschnitt 2.1.5 dargelegten Architektur des MCDET Scheduling Systems. Das MCDET Scheduling System erlaubt die performante Parallelisierung der verschiedenen Simulationspfade, die durch die MCDET-Struktur mit mehreren dynamischen Entscheidungsbäumen entstehen. Das Scheduling System wurde in diesem Vorhaben erweitert, so dass es nun

möglich ist, neben kompiliertem Code, der als shared library (DLL) eingebunden ist, auch den Python-Code anzusteuern. Passend für den CrewModul-Rechencode wurde ein Simulations-Controller und ein Simulations-Treiber geschrieben. Der Simulations-Controller stellt die Befehlsschnittstelle zum vorliegenden Rechencode zur Verfügung, kontrolliert den Ablauf einer Simulation entsprechend der empfangenen Befehle und erlaubt, den aktuellen Systemzustand der Simulation von außen abzufragen.

Der Simulations-Treiber organisiert den bidirektionalen Datentransfer zwischen dem Probabilistik-Controller und dem Simulations-Controller. Die Daten und Steuerbefehle werden über einen geeigneten Kommunikationskanal (IPC) ausgetauscht.

3.1.5.6 CrewModul/MCDET Output

Das neue CrewModul lässt sich wie ATHLET oder der einfache Tank-Rechencode über MCDET steuern. Das heißt MCDET kann in seinem Probabilistik-Modul verschiedene Werte für kontinuierliche aleatorische Größen, wie z. B. die Dauer einer Handlung ausspielen, und der Scheduler sorgt für die Generierung verschiedener Ereignisbäume in Abhängigkeit der Sets an ausgespielten Größen. Die ausgespielten Größen werden aus einer vom Nutzer angegebenen Wahrscheinlichkeitsverteilung gezogen, so dass zu jedem Set auch eine Wahrscheinlichkeit gehört. Bei diskreten aleatorischen Größen, wie z. B. dem Erfolg oder Nichterfolg einer Kontrollhandlung erzeugt MCDET Verzweigungen in einem Ereignisbaum. In jedem Zweig wird dabei die Wahrscheinlichkeit für diese Verzweigung gespeichert. Das Ausgabeformat sind hierarchisch aufeinander aufbauende HDF5-Tabellen. Zur Auswertung dieser Tabellenstruktur wurden bereits im letzten Vorhaben Post-Processing-Werkzeuge entwickelt, die in diesem Vorhaben weiter verfeinert wurden.

Die Abb. 3.2 zeigt die zu einem Zweig eines dynamischen Ereignisbaumes gehörende HDF5-Tabelle. In diesem Fall wurden mehrere Entscheidungsbäume mit unterschiedlich ausgespielten Dauern für verschiedene Handlungen angelegt. Jede Zeile enthält die jeweilige Simulationszeit, die ID der durchgeführten Handlung und der zugehörigen Handlungsliste, die für diesen Baum ausgespielten Dauern von drei verschiedenen Handlungen und die Systemzustände bezüglich Erfolgs von Kontrollhandlung 1 (b_kh1), erfolgter Kontrolle von Pumpe jnb00 (b_jnb00), Erfolg von Kontrollhandlung 2 (b_kh2) und Erfolg von Kontrollhandlung 3.

	time	action	action_list	t_hl_1d01_act_0	t_hl_1d1_act_0	t_hl_1d2_act_0	t_hl_1d3_act_0	b_kh1	b_jnb00	b_kh2	b_kh3
0	30.6622...	1.0	1.01	19.6622006921...	206.93368730...	188.08541665...	188.08541665...	2.0	2.0	1.0	1.0
1	218.747...	0.0	1.2	19.6622006921...	206.93368730...	188.08541665...	188.08541665...	2.0	2.0	1.0	1.0
2	218.747...	1.0	1.2	19.6622006921...	206.93368730...	188.08541665...	188.08541665...	2.0	2.0	1.0	1.0
3	2468.74...	0.0	2.1	19.6622006921...	206.93368730...	188.08541665...	188.08541665...	2.0	2.0	1.0	1.0
4	2483.74...	1.0	2.1	19.6622006921...	206.93368730...	188.08541665...	188.08541665...	2.0	2.0	1.0	1.0
5	2483.74...	2.0	2.1	19.6622006921...	206.93368730...	188.08541665...	188.08541665...	2.0	2.0	1.0	1.0
6	2551.24...	0.0	3.1	19.6622006921...	206.93368730...	188.08541665...	188.08541665...	2.0	2.0	1.0	1.0
7	2552.24...	1.0	3.1	19.6622006921...	206.93368730...	188.08541665...	188.08541665...	2.0	2.0	1.0	1.0

Abb. 3.2 Ausgabe einer der HDF5-Zustandstabellen eines MCDET gesteuerten Laufs des neuen CrewModuls

3.1.5.7 CrewModul zukünftige Weiterentwicklungen

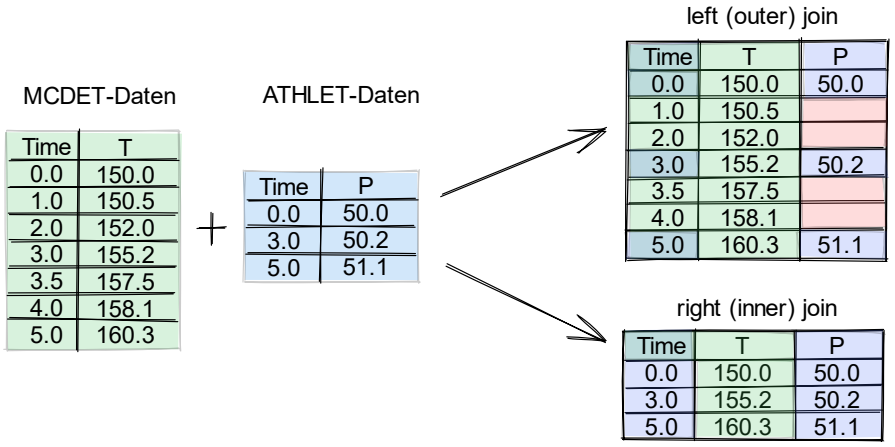
Künftige Weiterentwicklungen betreffen u. a. die Gestaltung des FreeMind MindMap Inputs. Da das FreeMind Software Programm nicht weiter gewartet wird, wird ein gleichwertiger Ersatz gesucht. Zudem soll der Input benutzerfreundlicher und damit weniger fehleranfällig gestaltet werden. Auch soll damit umgegangen werden, dass ein MindMap Input nicht optimal dafür geeignet ist, wenn eine Handlungsliste zu einer anderen Handlungsliste in einer anderen Verzweigung führen soll. Auch soll das CrewModul so gestaltet werden, dass es über MCDET in Interaktion mit einem Rechencode wie z. B. ATHLET arbeiten kann.

3.2 Erweiterung der Zugriffsmöglichkeiten von MCDET auf ATHLET-CD-Größen

Abhängig von der Ausbaustufe des eingesetzten Rechencodes sind viele Rechencode-Größen im direkten Zugriff während der Simulation verfügbar und können von MCDET überwacht und zur zeitschrittgenauen Evaluierung herangezogen werden. Obwohl die Anbindung von ATHLET/-CD im Rahmen dieses Arbeitspunkts weiter ausgebaut werden konnte und mittlerweile einen sehr robusten Stand erreicht hat, gibt es auch hier immer wieder Werte, welche vom Rechencode zwar berechnet, aber während der Simulation für MCDET nicht zur Verfügung stehen. Dadurch fehlen diese Größen in der Ausgabedatei (HDF5) von MCDET und können somit auch nicht mit den üblichen Algorithmen des Post-Processings ausgewertet werden. Um auch diese Werte in die Auswertung miteinbeziehen zu können, müssen die jeweiligen Zeitreihen aus den Ausgabedateien des Rechencodes gelesen und unter Zuordnung zur entsprechenden Sequenz in das HDF5-Format konvertiert werden.

In diesem Arbeitspunkt sollte neben Erweiterungen zum direkten Datenzugriff ein Konverter implementiert werden, der die benötigten Daten aus den ATHLET/-CD-Dateien (KEY-/PD-File) lesen und in die HDF5-Ausgabedatei von MCDET integrieren kann.

Der Direktzugriff auf den ATHLET/-CD-Simulationszustand konnte um TFO-, HCO-Daten und die Stützpunkte von GCSM-Tabellen erweitert werden. Für alle weiteren Größen kann dann der erarbeitete Datenkonverter verwendet werden. Zur Implementierung der darin enthaltenen Funktionalität wurde ein Python-Package entwickelt, welches die benötigte Konvertierung in mehrere Schritte unterteilt und sowohl von der Kommandozeile (Skriptbarkeit) als auch durch direkte Einbindung in andere Python-Module genutzt werden kann. Durch das separate Einlesen der Plotdaten-Struktur (KEY-File) können die gewünschten Zeitreihen vor der Konvertierung selektiert und flexibel gruppiert und hierarchisch zusammengestellt werden. Die dafür angelegten Datentabellen können dann in einer beliebigen HDF5-Zieldatei, wie einer bestehenden MCDET-Ausgabedatei, hinzugefügt werden, bevor diese Tabellen im darauffolgenden Schritt mit den Daten der Zeitreihen (PD-File) befüllt werden. Die Abb. 3.3 skizziert diese Situation sehr vereinfacht durch die Tabellen MCDET-Daten und ATHLET-Daten.



Time	T
0.0	150.0
1.0	150.5
2.0	152.0
3.0	155.2
3.5	157.5
4.0	158.1
5.0	160.3

Time	P
0.0	50.0
3.0	50.2
5.0	51.1

Time	T	P
0.0	150.0	50.0
1.0	150.5	
2.0	152.0	
3.0	155.2	50.2
3.5	157.5	
4.0	158.1	
5.0	160.3	51.1

Time	T	P
0.0	150.0	50.0
3.0	155.2	50.2
5.0	160.3	51.1

Abb. 3.3 MCDET- und ATHLET-Datentabellen

Die von MCDET erstellten Datentabellen unterscheiden sich durch die zeitliche Auflösung von importierten ATHLET-Daten. Die im Post-Processing notwendige Zusammenführung (join) kann in diesem Fall auf zwei verschiedene Modi (left/right) erfolgen, welche hier den Modi outer/inner entsprechen. Dabei muss für die im left (outer) join entstehenden Datenlücken (rot) festgelegt werden, wie diese gefüllt werden sollen (z. B. Stufung, Interpolation).

Um im Post-Processing die Auswertung zu erleichtern, wurde in weiteren Entwicklungsarbeiten untersucht, wie der Datenzugriff auf MCDET-Ausgabedaten und importierte Zeitreihen möglichst einheitlich erfolgen kann. Durch eine entsprechende hierarchische Ablage und der Verwendung des Zeitstempels als Pivot-Spalte (Time) können auch nachträglich hinzugefügte Tabellen zu vollständigen Sequenzen zusammengestellt werden (vgl. Abb. 3.3). Allerdings enthalten die importierten Tabellen nur die Daten der Zeitpunkte, zu denen der Rechencode seine Plotdaten geschrieben hatte. Da der Rechencode seine Daten in aller Regel mit einer weitaus geringeren Frequenz schreibt als das zeitschrittgenaue Monitoring von MCDET, müssen die Zeitreihen noch auf eine gemeinsame Zeitskala gebracht (join) werden, bevor diese einheitlich weiterverarbeitet werden können. Hierzu wurden die gängigsten Möglichkeiten (inner/outer-, left/right- join) untersucht und in Form einer exemplarischen Implementierung getestet. Für einen stabilen Einsatz dieser Datenzusammenführung sind weitere Entwicklungsarbeiten notwendig.

3.3 Anpassungen für die Nutzung von Linux-Clustern und Gitlab Runnern

Das MCDET Scheduling-System wurde als plattformunabhängige Anwendung konzipiert und implementiert. Darüber hinaus wurden viele Inkompatibilitäten zwischen Python-Versionen oder durch zu spezifische Systemprogrammierung (Windows/Linux) ausgelöste Probleme behoben. Damit ist MCDET, eine passend eingerichtete Python-Umgebung vorausgesetzt (vgl. Abschnitt 4.3.1), grundsätzlich sowohl auf Windows als auch auf Linux lauffähig, und es stellt sich schnell die Frage nach der Nutzung von Cluster-Systemen.

Im Vergleich zur Ausführung auf Arbeitsplatzrechnern herrschen in Cluster-Umgebungen (z. B. Linux- oder Gitlab-Cluster) oft sehr unterschiedliche Systembedingungen. Zweifellos bieten diese Systeme durch ihre Flexibilität, Skalierbarkeit und bündelbare Ressourcen (Rechenkerne, Arbeitsspeicher, Volume-Speicher) große Vorteile bei der Durchführung von MCDET-Analysen. Jedoch erfordert ihre Nutzung auch die Parametrisierung des jeweiligen Ressourcen-Management-Systems, wie z. B. die Erstellung von PBS (Portable Batch System)-Skripten auf Linux-Clustern oder auch YAML[1]-Skripten unter Gitlab. Diese Konfigurationsschritte dienen dazu, dem Cluster vorab die zur Job-Durchführung benötigten Ressourcen mitzuteilen und können somit nicht vom MCDET

[1] Yet Another Markup Language oder Ain’t Markup Language (rekursives Akronym)

Scheduling-System selbst durchgeführt werden. Da hierbei, abhängig von Cluster-Umgebung und Ressourcen-Bedarf der Simulationsprozesse, vielerlei Randbedingungen zu beachten sind, kann diese Einrichtung bislang nur manuell und speziell auf die Bedürfnisse eines Analyselaufs erfolgen.

Verschiedene Testanwendungen unter Verwendung des in Gitlab integrierten Jobsystems haben gezeigt, dass sich MCDET-Rechenläufe sehr gut dafür eignen, die in Cluster-Systemen vorhandenen Ressourcen und Möglichkeiten zur Parallelisierung zu nutzen. Neben der parallelen Abarbeitung von DETs, welche besonders durch die Nutzung mehrerer Kerne beschleunigt werden kann, können MCDET-Läufe mit unterschiedlichen Wertekombinationen für die epistemischen Unsicherheiten auch auf verschiedene Rechner (Knoten) aufgeteilt werden. Im Gitlab-Jobsystem kann dies über eine Job-spezifische Parametrisierung der Scheduler Startargumente erzielt werden. Diese sorgt dafür, dass der Scheduler auf einem Knoten einzelne oder mehrere in bestimmten Intervallen enthaltene Wertekombinationen bzgl. der Epistemik auswählt und für diese die MCDET-Rechenläufe startet. Darüber hinaus wurde die Flexibilität beim Ansteuern von MCDET-Rechenläufen dahingehend erweitert, dass in Bezug auf eine epistemische Wertekombination nur einzelne DETs oder ein Intervall von DETs berechnet werden können. Dies erlaubt ein einfaches Verteilen der Rechenjobs auf heterogene Rechensysteme, die beispielsweise aus unterschiedlich leistungsstarken Rechnern innerhalb eines Rechnerclusters bestehen. Das Gitlab Jobsystem erlaubt auch die Abarbeitung eines MCDET-Rechenlaufs über ihr Webinterface live zu verfolgen und den Fortschritt der Jobs zu beurteilen. Die dabei entstehenden MCDET-Ergebnisse (HDF5-Ausgabedateien) können nach Abschluss der Berechnungen als sogenannte artifacts heruntergeladen und durch ein HDF5-Master-File für das Post-Processing zusammengeführt werden. Somit wurde die Möglichkeit geschaffen, das rechenintensive Erstellen von DETs parallel auf verteilten Systemen zu rechnen und die fertigen Rechnungen automatisiert zusammen zu führen.

4 Entwicklungen zur interaktiven Anwendung von MCDET und zur Neustrukturierung des MCDET-Inputs

4.1 Überarbeitung des MCDET-Eingabeformats

Das MCDET-Eingabeformat wurde so überarbeitet, dass es den Anwender auch bei schwierigeren probabilistischen Spezifikationen durch eine standardisierte Syntax und eine klare Eingabestruktur unterstützt. Ein Ziel bei der Neustrukturierung war u. a., die bisherigen Limitierungen bei der Definition von mathematischen Ausdrücken oder Funktionen für das Ansprechen eines Triggers zu überwinden und auch komplexe Ausdrücke als Triggerfunktion zu ermöglichen.

Im Zuge der schrittweisen Umstellung von MCDET auf Python war zunächst das Ziel, die Eingabe auf Python aufzubauen. Dabei stellte sich allerdings heraus, dass dies für die Kommunikation und den Austausch von komplexen Objekten zwischen dem Fortran-Kern und Python keine robuste Lösung ist. Stattdessen sollte auch die Auswertung der Trigger, welche die komplexen Funktionen beinhalten, auf die Python-Seite ausgelagert werden. Auf Basis dieser Entscheidung ergab sich als logische Konsequenz, dass sowohl Terminierungsfenster als auch Funktionsfenster ebenfalls in Python ausgelagert wurden. So bestehen Terminierungsfenster (‚Termination Windows', früher: ‚Absorbing Windows') im Wesentlichen aus einem Trigger (siehe Abschnitt 4.1.6.1) und Funktionsfenster (‚Function Windows') neben einem Trigger aus Funktionen (siehe Abschnitt 4.1.6.5), welche, analog zu den Triggerfunktionen, durch die Umstellung auf Python auch aus komplexeren Ausdrücken bestehen können.

Da sich die Fenster für Verzweigungen (‚Transition Windows') und für das zufällige Ausspielen der aleatorischen Unsicherheitsgrößen (‚Sampling Windows') auf die DET-Struktur auswirken, wurden die Aktionen dieser Fenster noch nicht in Python implementiert. Für diese Fenster werden nur die Trigger in Python ausgewertet. Eine vollständige Umstellung des MCDET-Kerns von dem Fortran auf Python ist künftig geplant.

Für Anwender wird sich durch die Umstrukturierung lediglich das Eingabeformat ändern. Dabei bleibt die Grundidee der Eingabestruktur äquivalent. Aber im Gegensatz zu dem vorherigen Eingabeformat, das aus einem einfachen Textfile bestand, besteht die neue Eingabestruktur aus einer Python-Datei.

Genauso wie bei der alten Eingabe ist die Angabe eines deterministischen Codes, der den Rechencode aufruft, sowie die minimale Cut-off-Wahrscheinlichkeit, bis zu der eine Sequenz noch gerechnet wird, in der neuen Eingabe obligatorisch. Zudem müssen Simulationsvariablen und mindestens ein Verzweigungsfenster definiert werden. Im Falle der Berücksichtigung unsicherer Größen, egal ob epistemisch oder aleatorisch, müssen die Variablennamen spezifiziert werden. Alle weiteren Eingabeparameter, Time/State Variablen, Terminierungsfenster, Samplingfenster oder Funktionsfenster sind optional.

Eine Neuerung ist, dass die Verzweigungswahrscheinlichkeiten, die nicht zu Beginn bekannt waren, sondern bei denen z. B. die zugrundeliegenden Verteilungsfunktionen noch von epistemischen Größen abhängen, auch als Time/State Variablen betrachtet werden. Daher fand eine Umbenennung der Time/State-Variablen zu Time/State/Probability-Variablen statt, oder kurz TSP-Variablen. Diese Zusammenführung konnte erreicht werden, da nun in Funktionsfenstern auch komplexere Funktionen definiert werden können, wie z. B. Verteilungsfunktionen. Eine weitere Neuerung ist die Umbenennung des Begriffs ‚Fenster'. Bisher bestand ein Fenster aus einem Trigger (bestimmtes Zeit-/Zustandsfenster einer Simulation) und einer Aktion. Dies wurde nun umgestellt und insgesamt als ‚Aktion' bezeichnet, die einen Trigger besitzt, der bestimmt, ob die Aktion ausgeführt wird. Im Folgenden wird daher ausschließlich von Aktionen gesprochen und, um Verwirrungen zu minimieren, auch die englischen Begriffe verwendet, da auch die Eingabe mit den englischen Begriffen erfolgt.

Die Zusammenhänge der neuen Python-Struktur, die in

Abb. 4.1 dargestellt ist, sowie der Aufbau des neuen Eingabeformats werden in den folgenden Abschnitten erläutert.

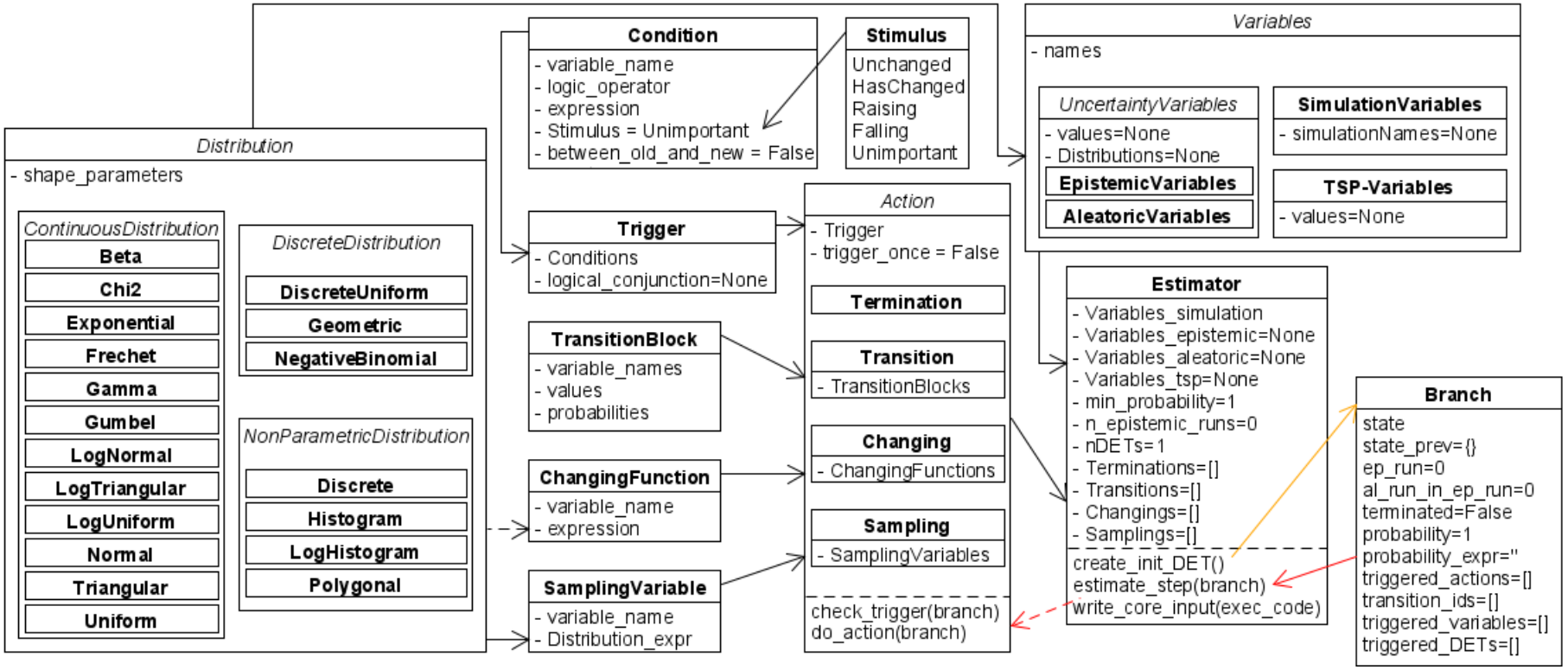


Abb. 4.1 Klassenstruktur des neuen auf Python basierten Estimators für das überarbeitete Eingabeformat

4.1.1 Branch

Ein Branch ist das zentrale Objekt, dass den aktuellen Zustand eines Rechenschrittes einer Sequenz enthält, sowie jegliche sonstigen Parameter, die für eine Bewertung durch MCDET notwendig sind. Ein Branch wird zwar von einem Anwender in der Eingabe nicht erstellt, sondern nur intern in MCDET verwendet, hilft aber, die Funktionalitäten der anderen Eingabeobjekte zu verstehen. Nach jedem Rechenschritt wird ein Branch aktualisiert und von dem Kern von MCDET bewertet, weshalb dieser auch in dem sogenannten Estimator integriert ist. Für jeden neu zu rechnenden DET und bei jeder Abzweigung wird ein neuer Branch erstellt, der dann in einer neuen Rechensequenz aktualisiert und bewertet wird.

Die wichtigste Information eines Branches ist der Zeitpunkt und der Zustand (‚state') aller Variablen, d. h. ein Python-Dictionary mit den Werten, welche die Variablen (einschließlich Zeitpunkt) aktuell haben. Eine Besonderheit ist hierbei der Zustand des initialen DETs, auch DET0 genannt, bei dem jeder aleatorischen Größe nicht ein einziger Wert, sondern ein Vektor an Werten zugeordnet wird. Dieser Vektor bezieht sich auf die Stichprobe an DETs, die im weiteren Verlauf generiert werden soll. Jeder seiner Werte wird also einem bestimmten DET zugeordnet.

Neben dem aktuellen Zustand (einschließlich Zeitpunkt) wird der vorherige Zustand im Branch gespeichert, um auch die Änderungen untersuchen zu können. Des Weiteren besteht ein Branch aus den folgenden weiteren Parametern, die vor allem organisatorisch relevant sind:

- ep_run_idx, der Index des epistemischen Laufes
- al_run_idx, der Index des aleatorischen Laufes innerhalb des epistemischen Laufes. Dies kann auch als DET-Nummer bezeichnet werden, wie es im alten Format genannt wurde.
- probability, die Wahrscheinlichkeit des Branches
- terminated, ein Parameter, der anzeigt, ob ein Branch bereits terminiert wurde.

Zusätzlich zu diesen Parametern, die auch alle in der alten Struktur vorhanden waren, wurde die Möglichkeit geschaffen, weitere Informationen zu hinterlegen, die zwar nicht

zum Erstellen eines DETs notwendig sind, die sich aber als nützliche Erweiterung für die weitere Auswertung ergeben haben.

In der alten Struktur konnte anhand des „window“-Parameters im EventTable der MCDET-Ausgabedatei ermittelt werden, welches spezielle Zeit-/Zustands-Fenster zuletzt eingetreten ist bzw. welche spezielle Aktion zuletzt getriggert wurde. Die Information, für welche DETs die Aktion ausgeführt wurde, konnte aus der Ausgabedatei nur durch Zurückiterieren der Elternpfade bestimmt werden. Allerdings betraf dies nur die Transition Actions. Zwar konnten auch Termination Actions ermittelt werden, indem über den „reason“ Parameter ermittelt wurde, warum der Branch beendet wurde, doch ist dies weniger für einen Benutzer offensichtlich. Auch wurde dabei nicht gespeichert, welche Termination Action verantwortlich war. Für Function Actions oder für Sampling Actions gab es hingegen keine Möglichkeit, deren mögliches Einwirken herauszufinden. In der neuen Struktur wurde über das triggered_actions Attribut des Branches die Möglichkeit geschaffen, alle Aktionen in der Reihenfolge, in der sie bisher getriggert und angewendet wurden, abzuspeichern.

Da in einer Abzweigung mehrere Simulationspfade (Kinder) mit jeweils unterschiedlichen Zustandsänderungen erzeugt werden können, wurde in der neuen Eingabestruktur neben der Information, dass eine Transition Action auf den Elternpfad angewendet wurde, hinzugefügt, welche der Änderungen auf welchen Kinderpfad angewendet wurde. Zum eindeutigen Identifizieren der Änderung werden drei Zahlen benötigt; die Aktionsnummer, der Index des Transition Blocks (siehe Abschnitt 4.1.6.3 zu Transition Actions) innerhalb der Transition und der Index der Änderung innerhalb des Transition Blocks. Durch jede Transition wird dem Branch des Kindes eine solche Wertekombination zu einer Liste hinzugefügt, die im Branch unter „transition_ids“ abgespeichert ist.

Zudem wurde die Möglichkeit eingebaut, die Wahrscheinlichkeit nicht nur als ein Wert, sondern als mathematische Formel abzuspeichern und bei jeder weiteren Abzweigung zu erweitern. Dies ist im Branch unter „probability_expr“ abgespeichert. Indem nicht nur der Wert, sondern auch die Formel zur Berechnung abgespeichert wird, kann im Nachhinein die Pfadwahrscheinlichkeit angepasst werden. Dies wurde mit Hinblick auf die in Abschnitt 2.3 beschriebenen alternativen Pfadwahrscheinlichkeiten implementiert.

4.1.2 Distribution

Verteilungen müssen angegeben werden, um einerseits die Werte von epistemischen und aleatorischen Unsicherheiten auszuspielen, oder andererseits, um integrierte Verzweigungswahrscheinlichkeiten berechnen zu können. Dabei stehen dieselben Verteilungsfunktionen wie in der alten Eingabestruktur zur Verfügung. Allerdings haben sich bei manchen Verteilungen die Eingabeparameter geändert, da auf die weit verbreitete und vielfach genutzte Python-Bibliothek scipy /VIR 20/ aufgebaut wurde und die Eingabeparameter dementsprechend angepasst wurden. Das bietet den Vorteil, dass die Python-Anwender, von denen die meisten mit scipy vertraut sind, da scipy ein Standardwerkzeug für wissenschaftliches Arbeiten in Python ist, sich nicht für MCDET umstellen müssen.

Die zur Verfügung stehenden Verteilungen sind in

Abb. 4.1 aufgelistet und unterteilen sich in kontinuierliche (stetige), diskrete und nicht-parametrisierte Verteilungen. Von den angegebenen Verteilungen sind die LogTriangular-, die LogHistogram- und die Polygonal-Verteilungen nicht in der scipy-Bibliothek enthalten und wurden in MCDET neu in Python implementiert.

4.1.3 Variables

Es gibt vier Arten von Variablen: Simulations-, epistemische, aleatorische, und Time/State/Probability-Variablen, die zusammen den Gesamtzustand (state) eines Branches bilden, der von MCDET bewertet wird. Wie bereits erwähnt, wurden die in der alten Eingabestruktur noch vorhandenen sog. Branching Probabilities, mit denen die Wahrscheinlichkeiten von Verzweigungen innerhalb von MCDET berechnet werden konnten, zu den Time/State-Variablen hinzugefügt, weshalb diese zu Time/State/Probabilitiy-Variablen umbenannt wurden. Innerhalb der alten Eingabestruktur war diese Zusammenlegung aller sonstigen Variablen, die keine Simulations- oder Unsicherheitsvariablen sind, nicht möglich, da keine komplexen Ausdrücke in den Funktionen der Function Windows beschrieben werden konnten.

Alle ‚Variables'-Objekte werden mit den Namen der Variablen initialisiert, sowie weiteren spezifischen Parametern, die in den einzelnen Klassen beschrieben werden. Die Namen sollten einzigartig sein, da sie als Identifikation der Variablen innerhalb des states dienen.

4.1.3.1 SimulationVariables

Alle Simulationsvariablen, die entweder für die Bewertung und Abzweigung oder auch für das Post-Processing und die Auswertung notwendig sind, sollten hier aufgeführt werden. Meist besitzen die Simulationsprozesse weitaus mehr Variablen, allerdings sollte man sich auf die notwendigen Variablen beschränken, um einerseits die Datenmenge und andererseits die Laufzeit zu reduzieren. Die Namen der Simulationsvariablen innerhalb von MCDET sollten durch einen einfachen zusammenhängenden String (ohne Leerzeichen) angegeben werden und können sich von den jeweiligen Namen, die im Rechencode verwendet werden, unterscheiden. Zum einen kann z. B. bei dem künftigen parallelen Ansteuern mehrerer Rechencodes die Zeitvariable „time“ mehrfach vorkommen und muss deshalb umbenannt werden. Daher kann optional mit dem Parameter „simulation_names“ der verwendete Name der Variablen im Rechencode angegeben werden. Aufgrund der großen Anzahl an Variablen innerhalb eines Rechencodes, sind Variablen meist in Modulen oder Submodulen strukturiert. Wie genau die Simulationsvariable mit ihren Modulen geschrieben werden muss, hängt, wie auch in der alten Eingabestruktur, von dem verwendeten Rechencode bzw. dessen Schnittstelle, dem Simulation Controller, ab.

4.1.3.2 EpistemicVariables

Epistemische Variablen gehören neben den aleatorischen Variablen zu den Unsicherheitsvariablen. Ihre Variation ist Ausdruck für ungenauen Kenntnisstand. Ihre Werte müssen entweder vom Anwender übergeben oder bei der Initialisierung ausgespielt werden. Das Ausspielen von Werten durch Sampling Actions ist für epistemische Variablen nicht möglich, weil Sampling Actions nur auf aleatorische Variablen anwendbar sind. Bei der Eingabe müssen entweder die ausgespielten Werte der epistemischen Variablen direkt mitübergeben werden oder deren Verteilungsfunktion, so dass MCDET sie während der Initialisierung ausspielen kann. Die zweidimensionale Liste von Werten wird dabei so eingelesen, dass jede Zeile ein Set von Parametern für einen epistemischen Lauf ist. Die Anzahl der Zeilen sollte daher mindestens die Anzahl der epistemischen Läufe entsprechen und die Anzahl der Spalten genau der Anzahl an von außen eingelesenen epistemischen Variablen.

Epistemische Variablen, die von MCDET ausgespielt werden, müssen getrennt von den extern ausgespielten Variablen definiert werden. Neben den internen Variablen muss eine Liste mit den jeweiligen Verteilungen übergeben werden, wobei die in

Abschnitt 4.1.2 beschriebenen Verteilungen zur Verfügung stehen. In der alten Eingabestruktur war kein Ausspielen während der Initialisierung möglich, sondern nur ein Übergeben von extern ausgespielten Werten. Grund dafür war, dass zwischen epistemischen Größen Abhängigkeiten (z. B. Korrelationen) vorhanden sein können und diese innerhalb des MCDET-Kerns noch nicht behandelt werden können. Deshalb wurde früher das Ausspielen von Werten mithilfe von SUSA /KLO 21/ durchgeführt. Da aber geplant ist, den MCDET-Kern um Abhängigkeitsmodelle zu erweitern, wurde das Ausspielen von Werten für die epistemischen Variablen bereits ermöglicht. Allerdings dürfen zwischen den betreffenden epistemischen Variablen noch keine Abhängigkeiten vorhanden sein. In diesem Fall muss – genau wie früher – auf SUSA zurückgegriffen werden.

Wie epistemische Variablen in der neuen Eingabestruktur definiert werden können, ist anhand folgenden Beispiels dargestellt.

```
EpistemicVariables(['prob1', 'prob2'], [[0.1, 0.2], [0.15, 0.3], [0.05, 0.25]],
                   ['prob3', 'prob4'], [Uniform(0.05, 0.1), Normal(0.1, 0.1)])
```

Es werden vier epistemische Variablen definiert. Die ersten beiden Variablen mit den Namen prob1 und prob2 werden mit Werten definiert, die bereits extern ausgespielt wurden, z. B. mit SUSA. Die beiden anschließend definierten Variablen mit den Namen prob3 und prob4 werden jeweils mit einer Verteilung definiert, aus denen während der Initialisierung die entsprechenden Werte von MCDET intern ausgespielt werden.

4.1.3.3 AleatoricVariables

Aleatorische Variablen bilden, neben den epistemischen, die andere der beiden Arten von Unsicherheitsvariablen und beschreiben Unsicherheiten aufgrund zufälliger Ereignisse. Die Werte der im Allgemeinen stetigen aleatorischen Variablen müssen nicht unbedingt zu Beginn bereits feststehen und können auch während der Simulation erst durch Sampling Actions ausgespielt werden (siehe Abschnitt 4.1.6.7). Neben der Möglichkeit, wie bei den epistemischen Variablen, entweder die Werte oder die Verteilungen zu übergeben, gibt es bei aleatorischen Variablen auch die Möglichkeit sie in der Initialisierung nicht mit Werten zu definieren, sondern diese erst während der Simulation durch Sampling Actions auszuspielen.

Das Erstellen von (stetigen) aleatorischen Variablen ist äquivalent zu den epistemischen Variablen. Der Unterschied besteht bei den aleatorischen Variablen, für die intern Werte ausgespielt werden sollen. Egal, ob das Ausspielen bei der Initialisierung oder durch

Sampling Actions erfolgen soll, werden diese aleatorischen Variablen immer zusammen definiert werden. Allerdings muss für Variablen, die durch Sampling Actions ausgespielt werden, in der Liste von Verteilungen der in der Python-Sprache leere Wert „None“ eingefügt werden, da die betreffenden Verteilungen nur an einer Stelle, bei den Sampling Actions, definiert werden sollen.

Zudem ist zu beachten, dass die Verteilung nicht von anderen Variablen, z. B. von epistemischen Variablen abhängen kann. Falls diese Abhängigkeit besteht, müssen diese aleatorischen Variablen in einer Sampling Action ausgespielt werden, selbst wenn das Ausspielen während der Initialisierung stattfinden soll.

Das Erstellen von aleatorischen Variablen ist im folgenden Beispiel dargestellt und soll die Analogie und den Unterschied zur Definition epistemischer Variablen verdeutlichen.

```
AleatoricVariables(['t_fail1', 't_fail2'], [[100, 200], [103, 188], [99, 210]],
                   ['t_fail3', 't_fail4'], [Normal(500, 10), None])
```

Es werden vier aleatorische Variablen definiert. Die ersten beiden Variablen mit den Namen t_fail1 und t_fail2 werden, wie bei der obigen Definition epistemischer Variablen, mit extern ausgespielten Werten definiert. Bei den beiden anschließend definierten Variablen mit den Namen t_fail3 und t_fail4, die intern ausgespielt werden sollen, wird allerdings nur die erste mit einer Verteilung definiert. Die zweite Variable muss bei der weiteren Definition von entweder Sampling Actions oder Changing Actions Werte zugewiesen bekommen.

4.1.3.4 Time/State/Probability (TSP) Variables

TSP-Variablen beinhalten alle sonstigen Variablen, die nicht Simulations- oder Unsicherheitsvariablen sind. Während Time/State-Variablen und die Verzweigungswahrscheinlichkeiten (Probability) in der alten Eingabestruktur getrennt voneinander definiert werden mussten, können sie in der neuen Struktur gemeinsam behandelt werden. TSP-Variablen können bei der Definition einen initialen Wert oder den leeren Wert „None“ zugeteilt bekommen. Geändert werden können die Werte nur in Changing Actions.

4.1.4 Condition

Aktionen benötigen jeweils einen Trigger, der überprüft, wann sie ausgeführt werden sollen. Diese Trigger bestehen i.d.R. aus einer Liste von Bedingungen, sog. „Conditions“.

Jede Condition prüft, ob eine Bedingung bezüglich einer Variablen erfüllt ist. Die Condition besteht neben der Variablen, für die eine Bedingung überprüft werden soll, aus einem logischen Operator und einem mathematischen Ausdruck, einer sogenannten Expression. Diese Expression kann entweder eine einfache Zahl, oder ein String sein, der mathematische Funktionen enthalten und auch von anderen Variablen im „state“ abhängen kann. Dabei ist zu beachten, dass beim Potenzieren die Python-Schreibweise „**“ benutzt werden sollte, nicht das häufig übliche „^“. Die möglichen, bisher implementierten mathematischen Funktionen sind abs, sqrt, exp, log, sin, cos, tan, arcsin, arccos, arctan. Auch die Konstante pi kann verwendet werden.

Es gibt zwei weitere Argumente, die bei der Definition einer Condition übergeben werden können.

Mit dem „Stimulus“, der bereits im alten Eingabeformat vorhanden war, konnte die Veränderung zum letzten Wert der Condition-Variable berücksichtigt werden. War dieser Wert auf -1 gesetzt, wurde der Stimulus nicht berücksichtigt. Bei 0 musste der Wert unverändert sein, damit die Condition erfüllt wurde, und bei 1 musste sich der Wert verändert haben, damit die Condition eintrat. Anstatt der bisherigen Zahlen verwendet das neue Eingabeformat Enums, was die Fehleranfälligkeit verringert. Zudem wurden die Möglichkeiten hinzugefügt, zu prüfen, ob die Variable im Vergleich zum vorherigen Wert gestiegen oder gefallen ist. Die möglichen Enums sind somit

- HasChanged,
- Unchanged,
- Raising,
- Falling und
- Unimportant.

Falls der Stimulus nicht gesetzt wird, wird er in der Condition nicht betrachtet, bzw. der Standardwert ist Unimportant. Um die Entwicklung des Variablenwertes zu überprüfen, sollte beim Aufrufen der Condition der vorherige Wert der zu überprüfenden Variable mit übergeben werden. Falls ein Stimulus gesetzt wird, der nicht Unimportant ist, muss dieser vorherige Wert mit übergeben werden, um die Condition zu erfüllen.

Die boolesche Option „between_old_and_new“ ist ein weiterer optionaler Übergabeparameter, der nur hinzugenommen werden kann, falls ein Gleichheitsoperator verwendet

wird. Falls, wie beim Vergleich von float-Zahlen üblich, keine exakte Gleichheit erreicht werden kann, oder eine ausgespielte Triggerzeit zwischen zwei Simulationsschritten liegt, kann mithilfe dieser Option geprüft werden, ob der berechnete Ausdruck der Condition zwischen dem vorherigen und dem jetzigen Wert liegt. Dies wurde im vorherigen Input implizit angenommen, wenn zusätzlich der Stimulus entsprechend gesetzt wurde. Mithilfe dieser zusätzlichen Option ist dies nun klarer für den Anwender ersichtlich und steuerbar.

Die Überprüfung einer Condition wird mit dem Aufrufen des Condition-Objektes mit dem state durchgeführt. Dabei kann optional der vorherige Wert der zu überprüfenden Variable mitübergeben werden. Nachfolgend sind ein paar Beispiele für das Erzeugen und das Aufrufen von Conditions gezeigt.

```
cond1 = Condition('time', '=', '2 * pres + sqrt(valve)', between_old_and_new=True)
cond2 = Condition('temp', '>=', 1200, stimulus=Stimulus.Raising)
state = {'time': 202, 'temp': 1200, 'pres': 100, 'valve': 4}
cond1(state)
> True
cond2(state)
> False
cond2(state, 1190)
> True
```

In dem Beispielcode werden zunächst in den ersten beiden Zeilen jeweils eine Condition definiert. Die erste Condition soll verdeutlichen, wie die Variable time mit einem mathematischen Ausdruck, welcher von weiteren Variablen abhängen kann, in Beziehung gesetzt werden kann. Zusätzlich wurde der optionale Parameter „between_old_and_new“ gesetzt. In der zweiten Condition muss der Parameter temp einen Wert größer als 1200 besitzen, und zusätzlich im Vergleich zum vorherigen Wert ansteigen, um die Condition zu erfüllen. Anschließend wird ein Zustand, „state“ erzeugt, mit dem die Conditions überprüft werden. Während die erste Condition für diesen Zustand erfüllt ist, ist die zweite Condition aufgrund des Stimulus nur dann erfüllt, wenn zusätzlich der vorherige Wert von temp übergeben wird, der kleiner ist als der aktuelle.

4.1.5 Trigger

Trigger-Objekte bestimmen, wann eine Aktion (siehe Abschnitt 4.1.6) ausgeführt wird. Soll eine Aktion bereits während der Initialisierung ausgeführt werden, braucht keine Condition oder kein Trigger-Objekt erstellt werden und ein einfacher String „init“ ersetzt

den Trigger. Ein Beispiel, wie dies bei der Erstellung einer Aktion aussieht, findet sich im Abschnitt 4.1.6 zu Aktionen.

Falls ein Trigger von einer oder mehreren Bedingungen bzw. Conditions abhängt, müssen zunächst diese Bedingungen oder Conditions, wie im vorherigen Abschnitt beschrieben, definiert werden. Bei der Definition eines Triggers über eine Liste an Conditions gibt es im Vergleich zur alten Eingabestruktur zwei Erweiterungen. Im Standardfall wird angenommen, dass alle Conditions der übergebenen Liste erfüllt sein müssen, damit ein Trigger eine Aktion auslöst. Programmtechnisch werden die Conditions mit einem „and“-Operator verknüpft. Im neuen Eingabeformat ist es nun möglich, dass nur manche der angegebenen Bedingungen erfüllt sein müssen. D. h., nun lassen sich die Conditions auch mit „or“-Operatoren miteinander verknüpfen. Falls ein „or“-Operator verwendet wird, müssen alle logischen Verknüpfungen mithilfe des Übergabeparameters „logical_conjuctions“ angegeben werden. Dieser benötigt in dem Fall eine Liste aus Strings, die entweder aus „and“ oder „or“ bestehen. Auch wenn eine durch „or“ abgetrennte Subliste von Conditions bereits erfüllt ist, werden beim Aufrufen des Triggers stets alle Conditions überprüft.

Wie bei der Condition werden Triggerobjekte direkt mit dem state als notwendiger Übergabeparameter aufgerufen. Optional kann der state des letzten Simulations- oder Bewertungsschrittes übergeben werden, was für die Betrachtung des Stimulus oder bei der „between_old_and_new“-Option benötigt wird.

Zwei weitere optionale Parameter können beim Aufrufen des Triggers übergeben werden und erweitern den Rückgabewert um ein Python-Dictionary. Mit der Option „get_triggered_variables“ wird eine Liste der angesprochenen Variablen zusätzlich zurückgegeben. Falls eine Variable in mehreren erfüllten Conditions vorkommt, wird diese nur einmal zurückgegeben. Die Option „get_triggered_dets“ wird im DET0 (siehe Abschnitt 4.1.7) verwendet, wenn den aleatorischen Variablen ein Zahlenvektor zugewiesen ist, und gibt an, bei welchen der Werte im Vektor, die alle in einem eigenen DET verwendet werden, der Trigger erfüllt ist.

Im folgenden Beispiel wird gezeigt, wie mehrere Conditions miteinander verknüpft werden können.

```
cond1 = Condition('time', '=', '2 * pres + sqrt(valve)', between_old_and_new=True)
cond2 = Condition('temp', '<', 1300)
cond3 = Condition('temp', '>', 1100)
```

```
cond4 = Condition('pres', '>', 100)
cond5 = Condition('valve', '=', 4)
trig = Trigger([cond1, cond2, cond3, cond4, cond5], ['and', 'and', 'or', 'and'])
trig({'time': 200, 'temp': 1200, 'pres': 101, 'valve': 4}, get_triggered_variables=True)
> True, {'triggered_variables': ['pres', 'valve']}
trig({'time': 204, 'temp': 1200, 'pres': 100, 'valve': 4},
     {'time': 200, 'temp': 1200, 'pres': 101, 'valve': 4})
> True
trig({'time': 202, 'temp': [1100, 1200, 1300], 'pres': 100, 'valve': 4},
     get_triggered_variables=True, get_triggered_dets=True)
> True, {'triggered_variables': ['time', 'temp'], 'triggered_dets':[False, True, False]}
```

Es werden zunächst fünf Conditions definiert, die anschließend in der Definition des Triggers verknüpft werden. Für die Erfüllung des Triggers müssen entweder die ersten drei oder die letzten beiden Conditions erfüllt sein. Zudem ist im letzten Aufruf gezeigt, wie die getriggerten Variablen und DETs zurückgegeben werden.

4.1.6 Action

Wie bereits erwähnt, wurde der Begriff Fenster bzw. Windows in der alten Eingabestruktur durch den Begriff Actions ersetzt, da dieser Begriff mehr die Funktionalität beschreibt.

Jede Action benötigt für ihre Initialisierung einen Trigger, der bestimmt, wann die Action ausgeführt werden soll. Zusätzlich kann jede Action mit einer Identifikationsnummer, „action_number“ ausgestattet werden. Allerdings verwendet MCDET intern eine eigene Nummerierung. Trotzdem kann dies für Testzwecke vom Anwender gesetzt werden. Des Weiteren kann mit der Option „trigger_once“ festgelegt werden, dass eine Action nur einmal ausgeführt werden soll.

Die beiden wesentlichen Funktionen, die beide mit einem Branch-Objekt (siehe Abschnitt 4.1.1) aufgerufen werden, die jedes Action-Objekt besitzt, sind:

- **check_trigger(branch):** Diese Funktion überprüft, ob die Action ausgeführt werden soll, weil ein Trigger eingetreten ist, und ob die Action nur einmal triggern soll. Neben einem Branch-Objekt kann optional angegeben werden, ob sich der Estimator im Initialisierungsschritt befindet. In diesem Fall triggern nur die Actions, die statt einem Trigger-Objekt den String „init“ besitzen.
- **do_action(branch):** Diese Funktion führt die Action aus und ist in den Kinderklassen beschrieben.

Diese beiden Funktionen sind zwar nicht für das Eingabeformat relevant, beschreiben aber, wie die Zusammenhänge und Funktionalitäten der Python-Struktur aussehen. Die vier für die Eingabe möglichen Aktionen, die von dieser Action-Basisklasse erben, sowie weitere Hilfsklassen, die für die Erstellung der jeweiligen Actionklassen notwendig sind, werden in den folgenden Unterabschnitten beschrieben.

4.1.6.1 Termination

Termination Actions sind die einfachste Action-Klasse. Sie besitzen keine weiteren Initialisierungsparameter und ändern lediglich das „terminated“-Attribut eines Branches auf True. Dies bewirkt, dass innerhalb des Bewertungsschritts kein weiterer Bewertungsprozess für diesen Branch stattfindet, also nicht weiter überprüft wird, ob eine Aktion durchgeführt werden soll. Auch global wird dieser Branch nicht weiter gerechnet und gibt den Weg frei, so dass ein weiterer Branch in der Warteliste simuliert und bewertet werden kann.

4.1.6.2 TransitionBlock

Bevor eine Transition Action definiert werden kann, müssen zunächst alle in der Transition vorkommenden Verzweigungsblöcke oder TransitionsBlocks definiert werden. Ein TransitionBlock bestimmt, wie eine oder mehrere Variablen gleichzeitig auf eine oder mehrere neue Werte mit der jeweils selben Wahrscheinlichkeit verändert werden soll. Dies ist äquivalent zu den in Abschnitt 3.1.2 beschriebenen Änderungen am alten Fortran-Kern zur Zustandsänderung für eine Gruppe von Rechencode-Größen. Dies kann konkreter anhand der Initialisierungsparameter beschrieben werden.

Zum Erstellen eines TransitionBlocks werden folgende Eingaben benötigt:

- **„variable_names“:** Hierbei kann entweder ein einfacher String übergeben werden, wenn nur eine Variable verändert werden soll, oder eine Liste mit den Variablennamen, die verändert werden sollen.
- **„probabilities“:** Hier werden die Wahrscheinlichkeiten für die jeweiligen neuen Branches (child branches), die in diesem TransitionBlock erstellt werden sollen, angegeben. Dabei können entweder konkrete Zahlen oder auch Variablennamen übergeben werden. Falls nur ein neuer Branch abzweigen soll, braucht nur ein einfacher Wert oder Variablenname übergeben werden. Bei mehreren Abzweigungen muss

eine Liste übergeben werden, die in ihrer Länge der Anzahl der Abzweigungen entspricht.

- **„values“:** Hier muss eine zweidimensionale Liste übergeben werden, in der die Werte festgelegt werden, auf die die Variablen gesetzt werden. Diese 2D-Liste besteht aus n Zeilen und m Spalten, wobei n die Anzahl der Variablennamen beschreibt und m die Anzahl der Verzweigungen. Jede Zeile besteht somit aus m Werten, auf die jeweils eine Variable in den verschiedenen Abzweigungen gesetzt wird. Analog zu den Wahrscheinlichkeiten können die Werte entweder aus konkreten Zahlen oder Variablennamen bestehen.

4.1.6.3 Transition

Eine Transition Action wird neben dem Trigger und weiteren optionalen Parametern der Action-Basisklasse mit einer Liste aus TransitionBlocks (siehe Abschnitt 4.1.6.2) initialisiert. Jede der Verzweigungsblöcke generiert ein Set an Abzweigungen bzw. neuen Branches, die zwischen den Blöcken noch jeweils kombiniert werden. Die Anzahl der erzeugten Branches ergibt sich somit zu:

$$\prod_{i=1}^{N}(1 + m_i) \; - \; 1$$

wobei *N* die Anzahl der TransitionBlocks und *m* die Anzahl der Verzweigungen pro TransitionBlock *i* ist. Anhand des folgenden Beispielcodes wird verdeutlicht, welche Möglichkeiten es gibt, TransitionBlocks und Transition Actions zu definieren und wie viele Verzweigungen dadurch entstehen.

```
branch = Branch(dict(time=10, level=2, press=100, t_trig=5, p_level=0.4, xsec=0.4,
mass=300, flux=-20))
trig = Trigger(Condition('time', '>=', 't_trig'))
tblock1 = TransitionBlock('time', [[12, 't_trig']], [0.1, 'p_level'])
tblock2 = TransitionBlock(['press', 'level'], [[120],[3]], 0.1)
tblock3 = TransitionBlock('xsec', 44, 0.02)
tblock4 = TransitionBlock(['mass', 'flux'], [[320, 310],[-10, -15]], [0.01, 0.02])
transit = Transition(trig, [tblock1, tblock2, tblock3, tblock4], 42)
new_branches = transit.do_action(branch)
len(new_branches)
> 35
branch.probability_expression
```

```
> 1 * (1 - sum([0.1, p_level])) * (1 - 0.1) * (1 - 0.02) * (1 - sum([0.01, 0.02]))
new_branch[0].probability_expr
> 1 * 0.1 * (1 - 0.1) * (1 - 0.02) * (1 - sum([0.01, 0.02]))
new_branch[0].transition_ids
> ['42-1-1']
new_branch[-1].probability_expr
> 1 * p_level * 0.1 * 0.02 * 0.02
new_branch[-1].transition_ids
> ['42-1-2', '42-2-1', '42-3-1', '42-4-2']
```

Im obigen Beispiel sind alle Möglichkeiten, einen TransitionBlock zu erstellen, aufgezeigt: eine Variable mit einer Verzweigung (tblock3), mehrere Variablen mit einer Verzweigung (tblock2), eine Variable mit mehreren Verzweigungen (tblock1) und mehrere Variablen mit mehreren Verzweigungen (tblock4). Daraus ergeben sich 2*2*3*3-1=35 Verzweigungen.

In diesem Vorhaben wurde damit begonnen, die Kombinatorik zwischen den TransitionBlocks zu implementieren und die Ausdrücke der Pfadwahrscheinlichkeiten zu bestimmen. Wie im Beispiel zu sehen, ergibt sich für die Wahrscheinlichkeit des Elternpfades nach der Transition

$$p_{\mathrm{parent}} \times \prod_{i=1}^{N} \left(1 - \sum_{j=1}^{M_i} p_{i,j} \right)$$

wobei N wieder die Anzahl der TransitionBlocks, M_i die Anzahl der Abzweigungen in TransitionBlock i und p_{ij} die Verzweigungswahrscheinlichkeit für die Verzweigung j im iten Transitionblock ist. p_{parent} ist die Wahrscheinlichkeit des Elternpfades bis zur aktuellen Transition.

Neben dem Elternpfad sind im obigen Beispiel auch die Wahrscheinlichkeiten von zwei innerhalb der Transition erzeugten Branches dargestellt. Dabei kann anhand der „transition_ids“ festgestellt werden, welche Abzweigungen in den Branches stattgefunden haben. Wie bereits in Abschnitt 4.1.1 zu den Branches beschrieben, bestehen diese Identifikationsnummern aus drei Zahlen: der Nummer der Transition Action, der Nummer des TransitionBlocks (wobei die Nummerierung der Reihenfolge in der Liste bei der Initialisierung entspricht) und der Nummer der Abzweigung innerhalb eines TransitionBlocks, welche wiederum durch die Reihenfolge der Werte und Wahrscheinlichkeiten bei der Initialisierung festgelegt ist. Dabei ist bei den letzten beiden Zahlen zu beachten, dass

Transitionblock- und Abzweigungsnummern bei 1 anfangen zu zählen. Durch die Kombination der TransitionBlocks können innerhalb einer Transition mehrere Änderungen stattgefunden haben. Dies ist im obigen Beispielcode in der letzten Zeile zu sehen, wo in einer Liste die verschiedenen Identifikationsnummern der durchgeführten Verzweigungen eines Branches gezeigt werden.

4.1.6.4 ChangingFunction

Eine ChangingFunction ist wie ein TransitionBlock eine Helferklasse für Changing Actions. Zum Verständnis, wie die Verknüpfung zwischen einer ChangingFunction und einer Changing Action funktioniert, hilft hier die Analogie zwischen einer Condition und einem Trigger (siehe Abschnitt 4.1.4 und 4.1.5). Ein Trigger verwendet eine Liste von Conditions, die jeweils eine Variable mit einer Expression verknüpfen, wobei die Verknüpfung ein logischer Operator ist. Genauso besteht eine Changing Action aus einer Liste an ChangingFunctions, die jeweils eine Variable mit einer Expression verknüpfen. Der Unterschied besteht hierbei allerdings in der Verknüpfung, welche aus einem Gleichheitsoperator besteht, da der Wert der Variable auf das Resultat der Expression gesetzt wird.

In einer Expression können dieselben mathematischen Funktionen verwendet werden wie in einer Condition. Zusätzlich können die Verteilungen des distribution Moduls (siehe Abschnitt 4.1.2) verwendet werden, wodurch die separate Betrachtung der Branching Probabilities in der alten Eingabestruktur ersetzt werden konnte.

Im Gegensatz zu Transition Actions, wo die Werte von Simulationsvariablen geändert werden, was zu Abzweigungen führt, werden in Changing Actions keine Simulationsvariablen geändert, sondern nur Time/State/Probability Variablen oder interne (stetige) aleatorische Variablen. Epistemische Variablen ändern sich grundsätzlich nicht, während sich ein DET aufbaut.

4.1.6.5 Changing

Changing Actions ändern direkt den Zustand von Variablen im aktuell zu bewertenden Branch ohne eine Abzweigung zu verursachen. Nur Time/State/Probability-Variablen und interne aleatorische Variablen, also solche, die nicht extern oder auch nicht während der Initialisierung ausgespielt werden, können durch Changing Actions geändert werden.

Eine ChangingAction wird mit einer Liste aus den zuvor beschriebenen ChangingFunctions initialisiert. Falls eine solche Aktion getriggert wird, wird der Zustand oder state des zu bewertenden Branches entsprechend den Changing Functions geändert.

Wie Changing Functions und Changing Actions konkret definiert werden können, wird anhand des folgenden Beispiels verdeutlicht.

```
func1 = ChangingFunction('level', 5)
func2 = ChangingFunction('p_level', 'level/100')
change = Changing(trig, [func1, func2])
```

Es werden Changing Functions definiert, wobei in der ersten der Wert der Variable level auf 5 gesetzt wird und in der zweiten der Wert der Variable p_level auf einen mathematischen Ausdruck, der von weiteren Variablen abhängt. Diese werden anschließend, zusammen mit einem zuvor definierten Trigger, für die Definition einer Changing Action verwendet.

4.1.6.6 SamplingVariable

SamplingVariable ist äquivalent zu ChangingFunction und TransitionBlock eine weitere Hilfsklasse, die für Sampling Actions benötigt werden. Hierin wird der Name und die Verteilung einer innerhalb von MCDET während der Simulation auszuspielenden (stetigen) aleatorischen Variable definiert. Andere Variablen können hierbei nicht angegeben und ausgespielt werden. Bei den Verteilungen kann aus den in Abschnitt 4.1.2 beschriebenen Verteilungen ausgewählt werden. Die Verteilung kann entweder direkt als Verteilung übergeben werden oder als Expression, analog den Expressions bei Conditions oder ChangingFunctions, falls die Verteilungsparameter von weiteren Parametern abhängen, die sich im state befinden, z. B. epistemischen Parametern.

4.1.6.7 Sampling

Einer Sampling Action werden die verschiedenen SamplingVariable-Objekte, die zu einem Trigger ausgeführt werden sollen, in einer Liste übergeben und zusammengeführt. Analog zu den Transition Actions werden hierbei nur die Trigger auf Python-Seite ausgewertet. Die eigentliche Aktion, das Ausspielen der aleatorischen Werte, wird weiterhin im Fortran-Kern durchgeführt.

Das Erstellen einer Sampling Action ist im folgenden Beispiel dargestellt:

```
param1 = SamplingVariable('t_trig', Normal(100, 10))
param2 = SamplingVariable('t_fail', Uniform(500, 20))
sample = Sampling(trig, [param1, param2])
```

Es werden zwei SamplingVariablen definiert. Einmal soll aus einer Normalverteilung und einmal aus einer uniformen Verteilung gezogen werden und anschließend, mit einem zuvor definierten Trigger, eine Sampling Action definiert.

4.1.7 Estimator

Der Estimator ist das zentrale Objekt, in dem alle zuvor beschriebenen Actions und Variables zusammengeführt werden, und der sowohl die Initialisierung der epistemischen und aleatorischen Läufe sowie die Bewertung jedes einzelnen Simulationsschrittes durchführt.

Zum Initialisieren benötigt ein Estimator die Cut-off-Wahrscheinlichkeit (minimale Wahrscheinlichkeit, unterhalb der ein Rechenlauf nicht weitergerechnet wird), die Anzahl der epistemischen und der aleatorischen Läufe, die gerechnet werden sollen, und die Variablen und Aktionen, die ausgeführt werden sollen. Wenn lediglich ein Simulationslauf gestartet werden soll, ohne Verzweigungen oder Terminierungen, kann ein Estimator auch nur mit Simulationsvariablen definiert werden, die während des getakteten Rechenlaufs rausgeschrieben werden sollen. Wenn auch dieses selektierte Rausschreiben nicht nötig ist, braucht kein Estimator definiert werden, und diese Anforderung kann über die direct_run Option durchgeführt werden (siehe Abschnitt 2.2 zur Behandlung von klassischen Monte-Carlo-Simulationen).

Zu den relevanten Methoden, die ein Estimator zur Verfügung stellt, gehört das Schreiben einer Eingabedatei für den Fortran-Kern. Diese benötigt als zusätzlichen Übergabeparameter den Kommandozeilenaufruf, in der alten Eingabestruktur auch Deterministic Code genannt, mit der eine Simulation oder der Simulation-Controller von der Konsole aus gestartet wird. Der Simulation-Controller muss dabei im Simulation-Modul von MCDET implementiert sein (siehe z. B. Abschnitt 3.1.5.5 zur Implementierung des CrewModuls).

Des Weiteren generiert ein Estimator die initialen DETs, auch DET0 genannt, die pro epistemischem Lauf erzeugt werden. Dabei werden, wie in Abschnitt 4.1.6.7 zu Sampling

Actions und den beiden Abschnitten 4.1.3.2 und 4.1.3.3 zu epistemischen und aleatorischen Variablen beschrieben, die Werte der Unsicherheitsvariablen übergeben, sofern sie mit einem init-Trigger definiert wurden und nicht von Prozessgrößen abhängen. Falls es sich um intern noch auszuspielende aleatorische Variablen handelt, werden diese aus den übergebenen Verteilungen ausgespielt.

Die Hauptaufgabe eines Estimators ist das Auswerten jedes Simulationsschrittes sowie das Ausführen von getriggerten Aktionen. Dabei werden, wie bereits beschrieben, für Transition und Sampling Actions nur die Trigger im Python-Estimator ausgewertet. Die eigentliche Aktion des Ausspielens oder der Verzweigung mit neuen Changesets findet weiterhin im Fortran-Kern statt. Dies findet in der Methode „estimate_step(branch)“ statt, die als Übergabeparameter ein Branch-Objekt bekommt, dessen Zustand bewertet wird.

Die Bewertung eines Branches findet in derselben Reihenfolge wie im alten Fortran-Kern statt. Zunächst werden in einer Schleife die Changing Actions auf den Branch angewendet, falls sie getriggert wurden. Im Anschluss folgt analog eine Schleife über die Termination Actions. Dabei bestimmt die Reihenfolge einer jeden Action-Liste, mit der der Estimator initialisiert wurde, die Reihenfolge der Ausführung. Falls der Branch noch nicht terminiert wurde, wird in einer nächsten Schleife überprüft, welche Sampling Action getriggert wurde. Im Anschluss erfolgt eine analoge Schleife über die Transition Actions. Die beiden zuletzt genannten Schleifen unterschieden sich von den beiden ersten Schleifen, da nur die Trigger ausgewertet, aber nicht die Aktionen ausgeführt werden, welche weiterhin im Fortran-Kern stattfinden. Trotzdem wird bereits die Identifikationsnummer der getriggerten Action der „triggered_actions“-Liste des Branches hinzugefügt.

Um die Informationen weiterzugeben, welche Sampling und Transition Action getriggert wurde, wird eine zweidimensionale boolesche Matrix angelegt, deren Elemente zunächst nur aus False bestehen. Neben der Information, welcher Action getriggert hat, benötigt der Fortran-Kern die Information, welche Variablen in den Triggern der jeweiligen Action überprüft wurden. Diese können, wie in Abschnitt 4.1.5 zu Triggern beschrieben, mit dem Übergabeparameter „get_triggered_variables“ erhalten werden. Diese Matrix besitzt für jede Sampling und Transition Action eine Zeile, die jeweils für jede Variable eine Spalte besitzen. Falls nun der Trigger einer Action erfolgreich ist, werden für die jeweiligen Variablen die entsprechenden Spaltenelemente der Action-Zeile auf True gesetzt. Diese Matrix wird dem Python-Dictionary, in dem die Information für die Kommunikation zwischen Fortran-Kern und Python-Scheduler ausgetauscht wird, hinzugefügt.

Falls der zu bewertende Branch zum DET0 gehört und damit allen DETs, die im Verlauf von MCDET Rechenläufen generiert werden, zuzuordnen ist, benötigt der Fortran-Kern zusätzlich die Information, welche DETs in einer Action getriggert wurden. Da nur in Transition Actions Verzweigungen bzw. Abspaltungen stattfinden können, werden auch nur die getriggerten DETs von Transition Actions betrachtet. Ein DET0 ist ein Konstrukt zur Effizienzsteigerung von MCDET-Rechenläufen, wenn mehrere DETs innerhalb eines epistemischen Laufes gerechnet werden sollen. In dem Fall werden die DETs nicht unabhängig voneinander gestartet, sondern es wird so lange in einem initialen DET0 gerechnet, bis sich die DETs durch die Werte der (stetigen) aleatorischen Unsicherheitsvariablen unterscheiden. Daher wird aleatorischen Variablen in einem DET0 jeweils ein Vektor von Werten zugewiesen. Jeweils ein Wert pro Vektor wird dann für die Berechnung eines DETs berücksichtigt, so dass schließlich eine Stichprobe von DETs innerhalb eines epistemischen Laufes generiert wird. Es wird in einem DET0, äquivalent zu den getriggerten Variablen, eine zweidimensionale boolesche Matrix erzeugt, deren Elemente zunächst nur aus False-Werten bestehen. Die Matrix besitzt für jede Action eine Zeile, die Anzahl ihrer Spalten ist durch die Anzahl der DETs (oder der aleatorischen Läufe) pro epistemischem Lauf definiert. Falls eine Action in einem DET0 getriggert wird, bei der ein Wert eines aleatorischen Vektors beteiligt ist, wird der entsprechende Wert in der Matrix auf True gesetzt, wie bereits in Abschnitt 4.1.5 zu Triggern beschrieben. Auch diese Information wird für alle DETs gesammelt und in der datamap für den Fortran-Kern hinterlegt.

4.2 Entwicklung zur einfacheren Spezifikation von Rechencode-Größen in MCDET

Neben der in Abschnitt 4.1 beschriebenen Überarbeitung der Eingabeformats wurden auch durch die Erweiterung des direkten Zugriffs von MCDET auf ATHLET/-CD-Größen (vgl. Abschnitt 3.2) deutliche Vereinfachungen in der Variablen-Spezifikation der MCDET-Eingabe möglich. TFOs und HCOs können mittlerweile über ihren symbolischen Namen adressiert werden, was den Zugriff auf deren Node-Variablen ungemein erleichtert. Darüber hinaus ist jetzt der koordinatenweise Zugriff auf die Stützpunkte von GCSM-Tabellen möglich, wodurch diese nicht mehr wie bisher über manuell bestimmte Indizes spezifiziert werden müssen.

4.3 Entwicklung einer grafischen Benutzeroberfläche

Neben den oben genannten inhaltlich-methodischen Arbeiten waren sowohl für die Bereitstellung der für die Ausführung notwendigen Systemumgebung (Python-Environment) als auch für die Implementierung des Post-Processings, dessen graphischen Darstellungen, bis hin zur Bedienbarkeit unter einer graphischen Benutzeroberfläche zusätzliche Arbeiten durchzuführen.

4.3.1 Automatische Einrichtung der Ausführungsumgebung

Sowohl MCDET-Läufe als auch die Ergebnisauswertungen im Post-Processing erfordern das Zusammenspiel einer Vielzahl von Bibliotheken und Python-Packages. Um reproduzierbares Verhalten und somit nachvollziehbare Ergebnisse zu erzielen, sind einheitliche Zusammenstellungen der einzelnen Komponenten und ihrer Versionen unerlässlich. Diese werden in Form von Python-Environments zusammengefasst, welche durch die entwickelten Startskripte automatisiert eingerichtet werden können. Da die Anforderungen für den Ablauf des Schedulers und den interaktiven Arbeitsschritten des Post-Processings sehr unterschiedlich sind, musste die Einrichtung der dafür erstellten Python-Environments separat konfigurierbar gemacht werden. Das reduziert die Gefahr von Versionskonflikten deutlich und erlaubt darüber hinaus eine schnelle und zielgerichtete Installation der notwendigen Pakete. Die automatisierte Einrichtung der Python-Environments hat mittlerweile einen sehr stabilen Stand erreicht, so dass diese sowohl für die Erstellung von Entwicklungs- und Anwendungsumgebungen als auch zur Einrichtung von CI-Testumgebungen verwendet werden kann. Darüber hinaus sorgt sie auch hier für einheitliche Ablaufbedingungen. Bislang wurden die Startskripte nur für Windows-Umgebungen erstellt. Die deutlich unterschiedlichen Systembedingungen unter Linux-Systemen machen eine Portierung der Skripte erforderlich, bevor die automatisierte Einrichtung auch unter Umgebungen wie z. B. einem Linux-Cluster (vgl. Abschnitt 3.3) genutzt werden kann. Diese Portierung könnte künftig angegangen werden, um die Verwendbarkeit von MCDET auch unter diesen Systemen zu erleichtern.

MCDET-Analysen werden in Form vieler automatisch gestarteter und meist parallel ablaufender Simulationsprozesse berechnet. Damit ein fehlerfreier Lauf möglich ist, ist es entscheidend für alle Prozesse sowohl eigene Arbeitsverzeichnisse als auch die passende Systemumgebung (Umgebungsvariablen) sicherzustellen. Eine dafür entwickelte Erweiterung erlaubt die benötigten Einstellungen in Form von JSON-Dateien zentral oder auch verzeichnisspezifisch zu konfigurieren. Da diese Einstellungen oft

entscheidend für das Importieren von Python-Packages und das Laden dynamischer Bibliotheken (DLLs) sind, wird ein einheitlicher Ablauf von MCDET unter den sehr unterschiedlichen Systembedingungen (PC, CI-Docker, Cluster) überhaupt erst möglich.

4.3.2 Überwachung und Kontrolle von Simulationsprozessen

Um auf Desktoprechnern den Start des MCDET-Schedulers zu vereinfachen und die Abarbeitung der Simulationsprozesse kontrollieren zu können, ist die Bereitstellung einer entsprechenden Benutzeroberfläche geplant. Das erarbeitete Konzept zur Aufteilung und interaktiven Steuerung wird in Abb. 4.2 dargestellt. Der bisher implementierte Prototyp erlaubt sowohl die freie als auch die dialog-basierte Auswahl der notwendigen Eingabedateien (die zu verwendende DLL des MCDET-Kerns, die MCDET-Eingabedatei) und des Wurzelverzeichnisses für die Arbeitsverzeichnisse der sukzessive gestarteten Simulationsprozesse. Nach dem Start können neben Statusmeldungen und Konsolenausgaben auch die momentan aktiven Simulationen (Tasks) in einem separaten Fenster verfolgt werden.

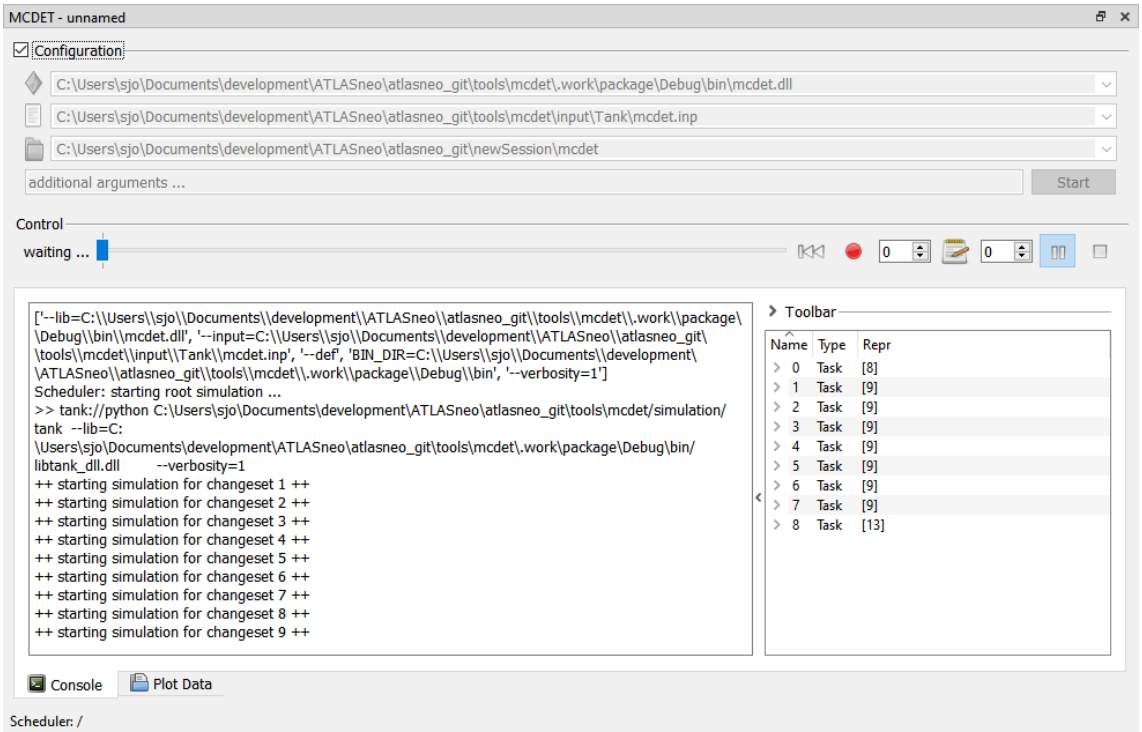


Abb. 4.2 Prototyp einer Benutzeroberfläche für Start und Überwachung von MCDET-Rechenläufen sowie die Kontrolle von Simulationsprozessen (Tasks)

Für den weiteren Ausbau der Benutzeroberfläche ist die Hierarchiedarstellung und die Statusabfrage zu den einzelnen Tasks geplant. Dabei ist allerdings zu beachten, dass sich durch die interaktive Bedienung für den Scheduler eine neue Betriebsart ergibt, für

die dieser erst entsprechend erweitert werden muss. Im Gegensatz zur Konsolen-basierten Ausführung muss der Scheduler hier als separater Thread abgearbeitet werden und jederzeit unterbrechbar sein, um auf die Daten der aktuellen Taskliste und der Simulationen zugreifen zu können. Hierfür ist die Nachrüstung einer Ereignis-basierten Synchronisation sowie die Überarbeitung der HDF5-Datenablage erforderlich, um die Thread-sichere Arbeitsweise sicherzustellen. Diese Entwicklungsschritte müssten künftig durchgeführt werden.

4.3.3 Visualisierung der DET-Topologie

Einer der Entwicklungsschritte hin zu einer grafischen Benutzeroberfläche ist die Visualisierung der DET-Topologie während der Berechnung, wobei Stand und aktuell bearbeitete Ereignispfade während der DET-Berechnungen fortlaufend dargestellt werden sollen. Im Gegensatz zur interaktiven Überwachung (siehe oben), welche die Topologie der Simulationsprozesse zeigt, zielt die Visualisierung auf eine möglichst deskriptive Darstellung der DET-Struktur und der aufgetretenen Ereignisse ab. Es wurden Jupyter-Notebooks verwandt, um den essenziellen Workflow für eine Darstellung der sich aufbauenden Ereignisbäume während der Laufzeit zu erarbeiten. Basierend auf der sich aufbauenden Verzeichnisstruktur und den in einem HDF5-File gespeicherten Informationen zu den durch den Scheduler ausgeführten Ereignissen (Start eines Prozesses, Abspaltung eines Prozesses und Terminierung eines Prozesses), kann die sich entwickelnde Baumstruktur rekonstruiert werden. Dabei wurden zwei mögliche Darstellungen der Baumstruktur erprobt. Im einen Fall entsprechen die Knoten den sich aufbauenden Verzeichnissen (siehe Abb. 4.3), im anderen Fall entspricht jeder Knoten dem Start oder der Abspaltung eines Prozesses (siehe Abb. 4.4) und die x-Achse zeigt den Zeitpunkt des Ereignisses an. In beiden Fällen können die sich ergebenden Bäume interaktiv unter Nutzung der Python-Bibliothek Plotly dargestellt werden. Durch die interaktive Darstellung erhält der Nutzer die Möglichkeit z. B. bestimmte Ausschnitte des Bildes zu vergrößern, um auch feinere Strukturen im Baum zu untersuchen, oder er kann mit der Maus über einen Knoten fahren, um mehr Informationen zu erhalten. All dies verringert die optische Last der Darstellung der MCDET-Baumstruktur. Die Schwierigkeit bei der Erarbeitung eines Algorithmus zur Darstellung der Baumstruktur lag darin, dafür zu sorgen, dass sich selbst bei großen Baumstrukturen keine überschneidenden Linien ergeben, so dass die verschiedenen Verbindungen optisch zumindest mit der Zoom-Funktion weiterhin gut aufgelöst werden können. Bei der Darstellung der Ereignisbäume wird der y-Achsenwert für einen Prozess konstant gehalten. Den abgespaltenen Prozessen werden kleinere y-Achsenwerte als den Mutterprozessen zugewiesen.

Der hier dargestellte Prototyp kann nur mit einer fertigen HDF5-MCDET-Ausgabedatei, also nach Beendigung einer MCDET-Simulation, erstellt werden. Die Darstellung der Baumstruktur während eines MCDET-Laufs ist dann möglich, wenn keine Probleme durch gleichzeitigen Schreib-/Lesezugriff auf die MCDET-Ausgabedatei entstehen. Hier kann das im vorigen Unterpunkt beschriebene Konzept zur Überwachung und Kontrolle von Simulationsprozessen erweitert werden. In diesem Konzept kann, wie oben dargelegt, während der MCDET-Laufzeit auf die für die Ereignisbäume benötigten Informationen zugegriffen werden. Über die QtWebEngine können plotly plots (siehe Abb. 4.2 und Abb. 4.3) dann in eine Qt-basierte GUI eingebunden werden.

Jeder Knoten entspricht einem neu angelegten Verzeichnis, in dem die jeweilige deterministische Simulation läuft. Der Nutzer kann Informationen über den Anfangszustand der neuen Ereignisverzweigung erhalten, indem er mit dem Mauszeiger auf den jeweiligen Knoten zeigt. Hier dargestellt sind Informationen, um welchen DET es sich handelt, der Zeitpunkt des Ereignisses, die Wahrscheinlichkeit für die Zeitsequenz zum Zeitpunkt des Ereignisses und der Pfad, unter dem die jeweilige Simulation läuft.

Die Möglichkeit, sich die Baumstruktur im Post-Processing interaktiv ansehen zu können, bringt einen signifikanten Mehrwert. So können außergewöhnlich früh beendete Pfade schneller gefunden werden und die interaktiv angezeigten Daten helfen zu verstehen, warum der Prozess beendet wurde sowie auch die entsprechende Information in der HDF5-Ausgabedatei von MCDET zu finden ist.

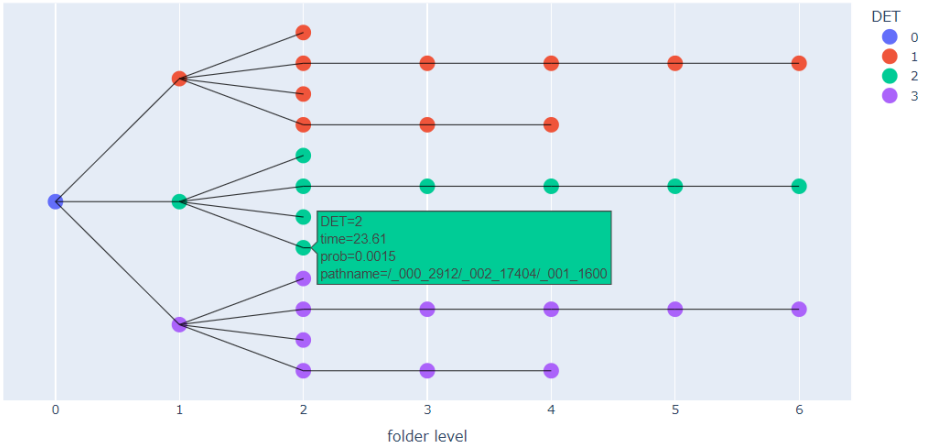


Abb. 4.3 Verzeichnisstruktur der dynamischen Ereignisbäume, dargestellt über die plotly Python-Bibliothek (Die unterschiedlichen Farben markieren die verschiedenen Ereignisbäume.)

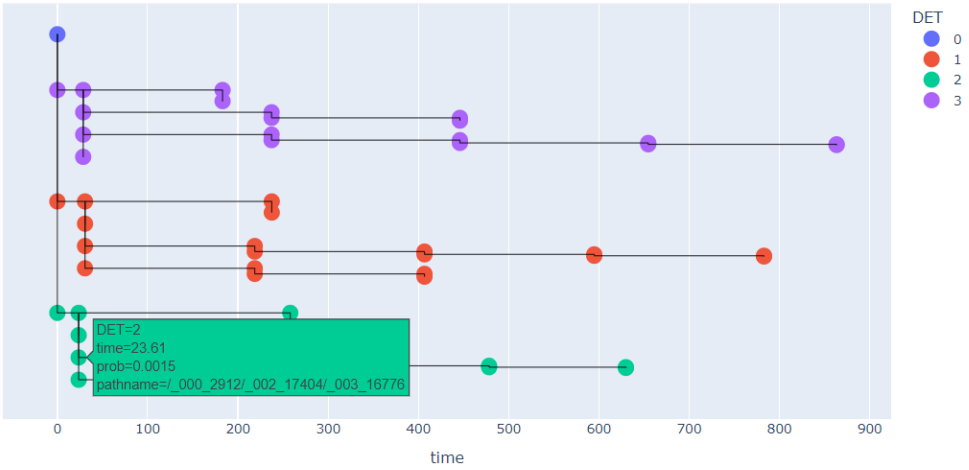


Abb. 4.4 Ereignisbaumstruktur dargestellt über die plotly Python-Bibliothek

In der Abbildung entspricht jeder Knoten einem Ereignis. Dargestellt sind nur Erzeugung und Abspaltung eines Prozesses und nicht die Terminierung, um die optische Last zu verringern.

5 Benutzerführung zum CrewModul

Im Rahmen der Erstellung eines Benutzerhandbuches für MCDET/CrewModul wird in diesem Vorhaben zuerst mit der Benutzerführung für das klassische CrewModul begonnen.

Für das CrewModul wurden in diesem Vorhaben Weiterentwicklungsarbeiten durchgeführt (siehe Abschnitt 3.1.5). Die nachfolgenden Beschreibungen beziehen sich auf das klassische Crew-Modul, auf dem die Weiterentwicklungsarbeiten aufbauen. Die allgemeinen Überlegungen zur Modellierung menschlicher Handlungen in einer dynamischen PSA behalten jedoch weiter ihre Gültigkeit. Dort wo Unterschiede zum neuen CrewModul bestehen, werden diese explizit aufgelistet.

5.1 Vorgehensweise und Konzept des CrewModuls

Die Vorgehensweise zur Modellierung und Simulation eines Handlungsablaufs mit dem CrewModul erfolgt in verschiedenen Schritten, wobei unterschiedliche Programme eingesetzt werden:

1) **Beschreibung des Handlungsablaufs in Abhängigkeit verschiedener Randbedingungen, die sich durch Unsicherheiten stochastischer Einflüsse oder bestimmter Prozesszustände ergeben**

   Im Gegensatz zur Modellierung physikalischer Prozesse, die durch Gesetzmäßigkeiten und mathematische Gleichungen definiert werden, ist eine mathematische Formulierung bei menschlichen Handlungsabläufen und Entscheidungsprozessen, in denen die beteiligten Personen ein bestimmtes Ziel verfolgen, im Allgemeinen nicht möglich. Aus diesem Grund ist man gezwungen, dass Handlungsabläufe, die sich in Abhängigkeit stochastischer Einflussgrößen oder verschiedener System- und Prozesszustände ergeben können, antizipiert und explizit beschrieben werden müssen.

   Die Beschreibung des Handlungsablaufs sollte vorzugsweise unter Einbeziehung der Expertise auf dem Gebiert Human-Factor (HF) erfolgen. Da stochastische Ereignisse (z. B. menschliche Fehler, Stressentwicklung, Aufenthaltsorte von Personen, Wegezeiten etc.) sowie System- oder Prozesszustände Einfluss auf den Handlungsablauf haben können, ist es notwendig, die Beschreibung des Ablaufs für alle unterschiedlichen Ereignisse oder Prozesszustände durchzuführen. In diesem Zusammenhang müssen auch diejenigen Handlungen definiert werden, bei denen mensch-

liche Fehler auftreten können und in der Analyse berücksichtigt werden sollen. Die Wahrscheinlichkeiten der menschlichen Fehlhandlungen können über die gängigen Methoden THERP /SWA 83/ oder ASEP SWA 87 bestimmt werden. Ein beispielhafter Auszug einer solchen Beschreibung ist in Abschnitt 4.2.5 aufgeführt.

2) Modellierung des Handlungsablaufs für das CrewModul

Nach Vorlage der Beschreibung des Handlungsablaufs, erfolgt die Modellierung des Ablaufs für das CrewModul über das Mind-Mapping Tool FreeMind®. FreeMind® ist eine frei erhältliche Software zum Visualisieren und Strukturieren von Inhalten. Zur benutzerfreundlichen und übersichtlichen Modellierung eines Handlungsablaufs unter Berücksichtigung von Abhängigkeiten und Unsicherheiten hat sich die Verwendung einer MindMap-Oberfläche bisher am vorteilhaftesten erwiesen. Die Modellierung des Handlungsablaufs erfolgt nach einer bestimmten Struktur und vorgegebenen Regeln (siehe Abschnitt 4.2).

3) Erstellung des Eingabedatensatzes für das CrewModul

Ist die Modellierung des Handlungsablaufs über die graphische Oberfläche abgeschlossen, wird der Inhalt der erstellten Mind-Map verwendet, um daraus den Eingabedatensatz für die Simulationen mit dem CrewModul zu erstellen. Dazu wird das eigens dafür entwickelte Python-Programm ‚readMindMap.py' verwendet. Mit diesem Programm wird der Inhalt der Mind-Map gelesen und daraus automatisch der Eingabedatensatz für das CrewModul erstellt. Die Vorgehensweise dazu und eine kurze Beschreibung des erstellten Eingabedatensatzes wird in Abschnitt 4.3 behandelt.

4) Erzeugung der Zufallszahlen für die Zeitverteilungen der Basishandlungen

Ein Handlungsablauf besteht aus vielen zusammengesetzten Einzelhandlungen (Basishandlungen). Ein charakteristisches Merkmal der dynamischen Modellierung von Handlungsabläufen besteht darin, dass für die einzelnen Basishandlungen die jeweiligen Ausführungszeiten angegeben werden müssen. Die Ausführungszeiten können sowohl als konstante Werte oder aber auch als Zufallsvariable definiert werden, die einer bestimmten Verteilung folgen. Für die Verteilungen der Ausführungszeiten wird in der Regel eine Gleichverteilung angenommen. Je nach Informationsstand können aber auch geeignete andere Verteilungen (z. B. Dreiecks- oder Histogrammverteilungen) für die Ausführungszeiten spezifiziert werden.

Für das klassische CrewModul wird die Zufallsauswahl der Ausführungszeiten, die als Zufallsvariable spezifiziert werden, über SUSA /KLO 21/ durchgeführt. Im neuen Vorhaben ist geplant, die Zufallsauswahl direkt innerhalb von MCDET durchzuführen. Damit könnte künftig dieser Zwischenschritt der SUSA-Anwendung entfallen.

5) Erstellung des Input-Datensatzes für MCDET

Um den Handlungsablauf in Abhängigkeit von stochastischen Einflussgrößen simulieren zu können, wird das Crew-Modul mit MCDET gekoppelt. Durch die Kopplung mit MCDET wird der Handlungsablauf unter Berücksichtigung aller möglichen Kombinationen der im Modell spezifizierten stochastischen Einflüsse simuliert.

Für das klassische CrewModul erfolgt die Erstellung des Input-Datensatzes für die alte Version von MCDET. Für die neue CrewModul Version entspricht die Erstellung des Input Datensatzes der Erstellung eines Input Datensatzes für eine standardmäßige MCDET Analyse mit einem deterministischen Rechencode.

6) Simulation des Handlungsablaufs

Im klassischen CrewModul erfolgt die Simulation des Handlungsmodells mit dem ursprünglich erstellten Fortran-Programm für das CrewModul, das in Verbindung mit MCDET (ebenfalls die ältere Version) für die Simulationsläufe gestartet wird. In diesem Vorhaben wurde das Programm des CrewModuls von Fortran auf Python umgestellt und diese Python-basierte Version mit der neuen Schedulerversion von MCDET gekoppelt.

7) Auswertung der Simulationsergebnisse

Die Auswertung der Simulationsergebnisse des CrewModuls hat zum Ziel, Zeitverteilungen für relevante Handlungen unter interessierenden Bedingungen zu schätzen. Unter Verwendung des CrewModuls wird die Schätzung von Zeitverteilungen relevanter Handlungen auf eine fundiertere Basis gestellt, indem Unsicherheiten und Ausführungszeiten explizit im Modell berücksichtigt werden. Die Auswertung erfolgt über die Verwendung eines IPython-Notebooks, das speziell für das Post-Processing der Simulationsergebnisse des klassischen CrewModuls erstellt wurde.

Konzept des CrewModuls

Das CrewModul wurde entwickelt, um folgende Gegebenheiten und Eigenschaften menschlicher Handlungsabläufe berücksichtigen zu können:

- Ein Handlungsablauf setzt sich aus einer Vielzahl einfacherer Einzelhandlungen (Basishandlungen) zusammen. Die Ausführung einer Basishandlung kann in der Regel einer bestimmten Person zugeordnet werden. Zur Ausführung einer Basishandlung wird eine gewisse Zeit benötigt. Die Erfahrung zeigt, dass der Mensch für die Ausführung der gleichen Handlung auch unter konstanten Ausführungsbedingungen normalerweise unterschiedlich viel Zeit benötigt. Ebenso wird auch die Reaktionszeit, mit der eine Person auf den Eintritt eines Ereignisses reagiert, nicht konstant sein, sondern mehr oder weniger stark variieren. Diese zeitlichen Schwankungen ergeben sich aus vielfältigen Faktoren, die von der Art und dem Ausmaß der aktuellen Beanspruchung, Stress durch äußere Einflüsse, Ablenkung durch andere Personen, bis hin zu individuellen Leistungsunterschieden (Tagesform) reichen. Aus diesem Grund können die Ausführungszeiten von Basishandlungen im CrewModul als Zufallsgrößen modelliert werden, deren Variation durch eine bestimmte Wahrscheinlichkeitsverteilung beschrieben wird. Wenn es sich bei den Basishandlungen überwiegend um einfache elementare Tätigkeiten handelt, kann davon ausgegangen werden, dass sich die Verteilungen der Ausführungszeiten in der Regel relativ einfach durch Expertenurteil abschätzen lassen, ohne auf Daten aus der Betriebserfahrung oder experimentelle Daten zurückgreifen zu müssen.

- Bei der Ausführung menschlicher Maßnahmen sind oftmals mehrere Individuen beteiligt, die miteinander kommunizieren und deren Handlungen von den Handlungen der anderen Operateure sowie von ergonomischen und kognitiven Faktoren sowie von der Wahrnehmung und Interpretation des Systemzustandes abhängen können. Informationen über den Zustand des Handlungsablaufs (und ggf. auch über den System- und Prozesszustand) erhalten die beteiligten Personen neben den Anzeigen in der Warte u. a. auch über die Kommunikation, die zwischen ihnen stattfindet. Kommunikationen können unterlassen oder auch falsch verstanden werden. Dies kann zu Zeitverzögerungen, Unterlassungsfehlern oder Ausführungsfehlern führen, was wiederum Einfluss auf den Handlungsablauf und schließlich auf den Unfallablauf haben kann. Sind mehrere Personen am Handlungsablauf beteiligt, sollten die Situationen berücksichtigt werden können, dass Handlungen verschiedener Personen zeitlich parallel ablaufen können, aber auch Abhängigkeiten zwischen den Handlungen verschiedener Personen und vom Prozesszustand bestehen können; Beispielsweise, wenn bestimmte Handlungen erst dann begonnen werden können, wenn andere Handlungen erfolgreich durchgeführt wurden oder wenn bestimmte Prozessbedingungen vorliegen.

- Zufällige (stochastische) Einflüsse können wesentliche Auswirkungen auf den Handlungsablauf haben und unterschiedliche Handlungsabläufe zur Folge haben. Außerdem können sie den Stress des Personals beeinflussen, wobei sich der Stresszustand wiederum auf den weiteren Handlungsablauf auswirken kann. Die Stressentwicklung ist dabei als eine dynamische Größe zu betrachten, die sich im zeitlichen Verlauf einer Handlungsmaßnahme ändern kann und auch bei verschiedenen Personen unterschiedlich verlaufen kann. Die Einschätzung des Stresslevels für bestimmte Personen ist grundsätzlich mit großen Unsicherheiten behaftet. Diese Unsicherheiten müssen im Modell berücksichtigt werden können.

- Entsprechend stochastischer Einflüsse können auch bestimmte Prozesszustände Auswirkungen auf den Handlungsablauf haben. Wenn z. B. im Rahmen einer Brandbekämpfungsmaßnahme der Brandläufer den Brandraum erreicht und die Rauchentwicklung im Brandraum noch nicht so weit fortgeschritten ist, so dass der Brandraum noch begehbar ist, wird die weitere Aktion des Brandläufers ganz anders sein, als wenn der Brandraum nicht mehr begehbar ist. Im ersten Fall wird der Brandläufer versuchen, den Entstehungsbrand mit dem Feuerlöscher zu löschen, während er im zweiten Fall vermutlich auf die Löschgruppe der Feuerwehr warten wird.

- Im Rahmen der Beschreibung einer Handlungsmaßnahme können in bestimmten Situationen Unsicherheiten darüber bestehen, für welche weiteren Maßnahmen sich die beteiligten Personen entscheiden. Diese Unsicherheiten treten insbesondere dann auf, wenn dem Personal in bestimmten Situationen alternative Handlungsmöglichkeiten zur Verfügung stehen. Das CrewModul in Verbindung mit MCDET ist in der Lage, Unsicherheiten über alternative Handlungsmöglichkeiten, Unsicherheiten über die Stressentwicklung verschiedener Personen, Unsicherheiten bzgl. stochastischer Einflüsse und Unsicherheiten bzgl. anderer Unsicherheitsquellen zu berücksichtigen.

Das grundlegende Konzept des CrewModuls besteht darin, einen komplexen Handlungsablauf, bei dem eine oder mehrere Personen beteiligt sind und interagieren können, durch eine Vielzahl von einfachen Einzelhandlungen (Basishandlungen) zu beschreiben.

Basishandlungen: Eine Basishandlung wird als eine abgeschlossene einfache Einzelhandlung verstanden, die von einem bestimmten Operateur durchgeführt wird. Die Basishandlungen können dabei einfache Tätigkeiten sein (z. B. Bedienung eines Schaltknopfes) oder auch eine Kommunikation zwischen Personen beschreiben (z. B. Schichtleiter (SL) weist Reaktorfahrer (RF) an, Hauptkühlmittelpumpen abzustellen). Je

feiner die Zerlegung eines komplexen Handlungsablaufs in Basishandlungen erfolgt, desto detaillierter kann der Handlungsablauf der zu bewertenden Maßnahme und die darin stattfindenden Wechselwirkungen modelliert und analysiert werden. Der Grad, wie fein die Maßnahme in Basishandlungen zerlegt wird, steht im Ermessen des Benutzers.

Jeder Basishandlung sind Attribute zugeordnet, die eine nähere Beschreibung der Basishandlung erlauben. Eine Beschreibung der Attribute einer Basishandlung erfolgt in Abschnitt 5.2.2.

Die Komplexität einer Basishandlung kann unterschiedlich hoch sein. Beispielsweise ist die Basishandlung ‚Elektriker führt Arbeiten am Reaktorschutz aus' komplexer zusammengesetzt als die Basishandlung, SL liest Anzeige zum DE-Füllstand ab'. Je elementarer (d. h. weniger komplex) eine Basishandlung definiert ist, desto leichter kann auch ein Zeitrahmen für die Ausführung der Handlung abgeschätzt werden. Bei der Spezifikation von Basishandlungen sollte darauf geachtet werden, dass die Basishandlung nicht zu komplex ist und keine wichtigen Interaktionen beinhaltet, die dann in der Analyse nicht explizit berücksichtigt werden können. Es sollte versucht werden, Basishandlungen, die zu komplex erscheinen, weiter zu zerlegen.

Verzweigungsvariable zur Berücksichtigung von Unsicherheiten und Abhängigkeiten: Neben den Basishandlungen, die Aktionen von Personen beschreiben, werden im CrewModul zusätzlich Verzweigungsvariablen verwendet, mit denen angegeben wird, an welcher Stelle des Handlungsablaufs es zu Verzweigungen, z. B. aufgrund einer bestimmten Unsicherheit, kommt. Über die Verzweigungsvariablen können Abhängigkeiten des Handlungsablaufs von aleatorischen Unsicherheiten (stochastischen Ereignissen) sowie Abhängigkeiten von Prozesszuständen modelliert werden. Die Abhängigkeit des Handlungsablaufs vom Prozesszustand tritt z. B. in der Situation auf, wenn im Rahmen einer Brandbekämpfung der Brandraum aufgrund der Verrauchung betreten werden kann oder nicht. Verzweigungsvariablen können durch zwei oder auch mehrere Alternativen definiert werden, die dann jeweils unterschiedliche Handlungsabläufe zu Folge haben. Die Definition der Alternativen einer Verzweigungsvariablen und die jeweiligen Wahrscheinlichkeiten, mit denen die entsprechenden Alternativen auftreten, werden in der Input-Datei des Werkzeugs MCDET spezifiziert, mit dem das CrewModul zu koppeln ist.

Handlungslisten: Die Basishandlungen, in die ein Handlungsablauf zerlegt worden ist, werden sequenziell zusammengefügt, um bestimmte Teilhandlungen zu beschreiben.

Diese sequenzielle Abfolge von Basishandlungen wird als eine Handlungsliste (HL) bezeichnet. Eine Handlungsliste (Sequenz von Basishandlungen) endet dann, wenn der weitere Handlungsablauf von Unsicherheiten oder Prozesszuständen abhängt. Eine Handlungsliste ist definiert durch

- eine eindeutige Identifikationsnummer,
- eine Bedingung, wann die jeweilige Handlungsliste aktiviert wird, und
- der Sequenz von Basishandlungen, durch die die Handlungsliste beschrieben wird.

Eine Handlungsliste wird dann aktiviert und ausgeführt, wenn die ihr zugeordnete Bedingung erfüllt ist. Es ist darauf zu achten, dass die einer Handlungsliste zugeordnete Bedingung eindeutig ist. Durch die Zuordnung einer speziellen Bedingung zu jeder Handlungsliste ist es möglich, Handlungsabläufe z. B. in Abhängigkeit

- von den dynamischen Entwicklungen des Prozess- und Systemzustands,
- vom Zustand des bisher erfolgten Handlungsablaufs,
- von kognitiven und ergonomischen Faktoren und
- von stochastischen Einflussfaktoren (aleatorischen Unsicherheiten)

zu berücksichtigen.

Die Flexibilität des Konzepts erlaubt dem Benutzer, die Handlungsabläufe in Wechselwirkung mit Prozesszuständen und stochastischen Einflussgrößen in einem beliebigen Detaillierungsgrad darzustellen.

In Abschnitt 5.2 wird anhand von Beispielen beschrieben, wie ein Handlungsablauf für die Anwendung des CrewModuls modelliert werden kann.

5.2 Modellierung eines Handlungsablaufs über die grafische MindMap Oberfläche des CrewModuls

Die Beschreibung menschlicher Handlungsabläufe kann sehr schnell komplex und unübersichtlich werden, wenn zeitabhängige Wechselwirkungen des Handlungsablaufs mit Prozess- und Systemzuständen sowie stochastischen Einflussgrößen berücksichtigt werden. Um eine systematische Modellierung eines komplexen Handlungsablaufs zu gewährleisten, erfolgt die Modellerstellung über eine grafische Oberfläche, die durch ein

Mind-Mapping-Tool zur Verfügung gestellt wird. Eine Mind-Map besteht aus einer Baumstruktur, in deren Knoten beliebige Informationen eingetragen werden können. Diese Art der Darstellung wurde als geeignet erachtet, Zusammenhänge und Abfolgen von Handlungen zu erstellen und zu visualisieren. Als Oberfläche für das CrewModul wird das plattformunabhängige Mind-Mapping-Tool ‚FreeMind' verwendet, das als freie Software (z. B. unter *http://freemind.sourceforge.net*) heruntergeladen werden kann. Für die Arbeitsweise mit ‚FreeMind' wird auf die Dokumentation von ‚FreeMind' verwiesen, die unter dem Menüpunkt *Hilfe → Dokumentation* aufgerufen werden kann. Auf die Arbeitsweise mit ‚FreeMind' wird deshalb in dieser Beschreibung nicht weiter eingegangen.

Die Vorteile der Modellierung über ein grafisches Mind-Mapping-Werkzeug besteht einerseits in der Möglichkeit, die Handlungsabläufe übersichtlicher darzustellen, wobei die Verzweigungen von Handlungsabläufen in Abhängigkeit stochastischer Ereignisse sowie System- und Prozesszuständen über die grafische Darstellung sofort sichtbar sind. Andererseits können Modifikationen im Handlungsablauf relativ unkompliziert vorgenommen werden. Wenn z. B. der Wunsch besteht, einzelne Handlungen differenzierter zu modellieren, oder wenn verschiedene Handlungsstrategien in das Handlungsmodell eingebaut und analysiert werden sollen.

An Beispielen soll veranschaulicht werden, wie die Beschreibung eines Handlungsablaufs aussehen kann, und wie die entsprechende Modellierung über die grafische Oberfläche in der entsprechenden Syntax durchzuführen ist. Die Beispiele sind so gegliedert, dass zunächst beispielhaft dargestellt wird, wie eine entsprechende Beschreibung lauten könnte und danach beschrieben wird, wie die Beschreibung in der grafischen Oberfläche zu modellieren ist.

5.2.1 Anfangsknoten

Nach dem Aufruf von ‚FreeMind.exe' erscheint zunächst eine Arbeitsfläche mit einem leeren Anfangsknoten mit der Bezeichnung ‚Neue Mindmap'. In diesen Anfangsknoten können bestimmte Informationen bzgl. des Handlungsablaufs eingegeben werden. Durch Anklicken des Knotens und der Menüauswahl ***Bearbeiten → Knoten in einem separaten Editor bearbeiten*** erscheint ein Editor für den Knoten, in den die entsprechenden Informationen eingegeben werden können.

Titel: *myTitle* (optional)

Anzahl der Personen: *n* (Anzahl der an dem Handlungsablauf beteiligten Personen)
1 - *Kurzbezeichnung der 1. Person* (optional: Beschreibung der Person)
2 - *Kurzbezeichnung der 2. Person*
:
n - *Kurzbezeichnung der n. Person* (optional: Beschreibung der Person)
#
W*eitere Informationen bzgl. der Maßnahme* (optional)

Die fett gedruckten Eingaben sind obligatorisch und dienen als Schlüsselwörter für die Interpretationen des Inhalts und zur automatischen Erstellung der Input-Datei für das CrewModul. Der kursive Text ist vom Benutzer einzugeben.

Es ist darauf zu achten, dass vor und nach dem Trennungszeichen ‚–' bei der Bezeichnung der Personen ein Blank gesetzt wird.

Das Ende eines Knotens wird generell mit einem # gekennzeichnet. Nach dem Abschluss eines Knotens durch # können beliebige weitere Informationen für den Knoten eingegeben werden.

Ein Beispiel eines Anfangsknotens ist in Abb. 5.1 gegeben. Nach der Eingabe enthält der Eingangsknoten einen Titel für die modellierte Maßnahme, die Anzahl der beteiligten Personen und die Bezeichnungen der einzelnen Personen. Eine Beschränkung der Anzahl der an der Maßnahme beteiligten Personen ist nicht gegeben.

Beispiel 1

In dem Modell zur Maßnahme „Sekundärseitiges Druckentlasten" (SDE) sind die folgenden sieben Personen beteiligt: ein SL, SL-Vertreter, RF, Leitstandfahrer, zwei Mechaniker und ein Elektriker.

Mit dieser Beschreibung kann der Anfangsknoten folgendermaßen spezifiziert werden:

Abb. 5.1 Eingaben für den Anfangsknoten eines Handlungsmodells

Der Titel, die Kurzbezeichnungen und die optionalen Beschreibungen der Personen sind vom Benutzer frei wählbar.

Mit der Definition der beteiligten Personen im Anfangsknoten wird intern für jede einzelne Person eine Variable erzeugt, bei der die Zeit festgehalten wird, wann die betreffende Person seine letzte Aktion beendet hat und für neue Aktionen zur Verfügung steht. Die Variablennamen haben die Kurzbezeichnung der Person, denen jeweils ein t vorangestellt ist, d. h., tSL, tSLV, tRF, etc.

5.2.2 Eingabestruktur von Handlungslisten und Basishandlungen

Die Spezifikation einer Handlungsliste (HL) erfolgt nach einer bestimmten Struktur:

HL: *IdNr* // Kommentar (optional)

Bed: *Bedingung 1*
Bed: *Bedingung 2*
Basishandlung 1
:
Basishandlung n
#

Nach dem Schlüsselwort **HL** ist eine eindeutige Identifikationsnummer für die Handlungsliste zu vergeben. Für die Identifikationsnummer (IdNr) kann eine beliebige Dezimalzahl angegeben werden. Eine IdNr darf nur genau einer HL zugeordnet werden. Zur besseren Überschaubarkeit ist es jedoch empfehlenswert, eine bestimmte Struktur bei der Vergabe der IdNr einzuhalten. Für die erste HL, die den Beginn der Maßnahme beschreibt, kann z. B. die IdNr 1.0 angegeben werden.

Durch die Angabe des //-Zeichens kann grundsätzlich in jeder Zeile optional ein Kommentar eingegeben werden.

Nach der Eingabe der IdNr erfolgen eine oder auch mehrere Bedingungen, bei denen die HL aktiviert wird. Die Bedingung wird nach dem Schlüsselwort **Bed:** eingegeben. Die Syntax einer Bedingungseingabe lautet:

Bed: & Bedingungsb*lock 1* **&** …. **&** Bedingungsblock *n*

Ein Bedingungsblock besteht aus der Angabe

Variable logischer Operator Wert (oder Variable)

Als Variable für Bedingungen können nur die für den Handlungsablauf definierten Variablen verwendet werden.

Für den logischen Operator kann ein Element der Menge {=, <, ≤, >, ≥, <>} eingesetzt werden, wobei <> den ≠ Operator bezeichnet. Für Variablen sind entsprechende Variablennamen einzugeben, die für die Spezifikation der Bedingung relevant sind.

Jedem Bedingungsblock muss ein **&**-Zeichen vorangestellt werden. Am **&**-Zeichen erkennt das Programm, dass ein Bedingungsblock eingelesen werden muss.

Es ist darauf zu achten, dass die zur Aktivierung einer HL spezifizierten Bedingungen eindeutig sind. D. h. eine Bedingung darf nur die Auslösung einer bestimmten HL zur Folge haben. Um die Einhaltung der Eindeutigkeit einer Bedingung zu vereinfachen, wurde die Variable **HLcont** eingeführt. Die Variable **HLcont** muss in jeder Bedingung verwendet werden, um anzugeben, an welcher vorherigen HL die aktuelle HL anschließt.

Nach der Bezeichnung der HL und der Spezifikation der Bedingung folgt die Spezifikation einer Reihe von Basishandlungen, die den ersten Teilablauf der Handlungsmaßnahme beschreibt. Die Spezifikation einer Basishandlung folgt nach der Syntax:

/ *AusführendePerson* / *AuswirkungAufWenOderWas* / t_{min} t_{max} // *Kurzbeschreibung*

AusführendePerson - beschreibt die Person, die die Handlung ausführt. Für *AusführendePerson* werden die Kurzbezeichnungen der Personen verwendet, z. B. SL, RF etc.

AuswirkungAufWenOderWas - bezeichnet die Person oder die Komponente, die von der Handlung beeinflusst wird. Falls sich die Basishandlung auf eine Aktion bezieht, die Einfluss auf den Ablauf des physikalischen Prozesses hat (z. B. Anschalten von Pumpen, Schließen von Ventilen, Zurücksetzen von Reaktorschutzsignalen etc.) und für diese Maßnahme eine Zeitverteilung ermittelt werden soll, wird an dieser Stelle eine negative Zahl eingesetzt. Es ist darauf zu achten, dass die verwendete negative Zahl genau dieser einen Aktion zugeordnet wird.

t_{min} t_{max} - Angabe der Zeit, die für die Ausführung der Basishandlung benötigt wird. Die Ausführungszeit einer Basishandlung wird grundsätzlich als eine zufällige Größe betrachtet. Ausführungszeiten, die keinen oder nur sehr kleinen zufälligen Variationen unterworfen sind, können jedoch auch als konstante Größen spezifiziert werden. Wenn die Ausführungszeit als Zufallsgröße spezifiziert wird, werden Minimalwert t_{min} und Maximalwert t_{max} des möglichen Zeitbereichs angegeben. Soll die Ausführungszeit als konstante Größe spezifiziert werden, wird der entsprechende konstante Wert zweimal eingegeben. Dadurch erkennt das Programm, welche der Ausführungszeiten als Zufallsvariable und welche Zeiten als konstante Werte in die Analyse eingehen.

optionale Beschreibung - Kurzbeschreibung der Basishandlung. Dieses Attribut einer Basishandlung ist optional, da es für die eigentliche Simulation des Handlungsablaufs nicht benötigt wird. Es wird jedoch empfohlen, eine sinnvolle Kurzbeschreibung des Inhalts der jeweiligen Basishandlung zu geben. Damit wird die Dokumentation und die Nachvollziehbarkeit des Handlungsmodells erheblich vereinfacht. Der optionalen Beschreibung muss das // - Zeichen vorausgehen.

5.2.3 Modellierung von Teilhandlungen durch Handlungslisten und Basishandlungen

Im nachfolgenden Beispiel 2 wird anhand der Beschreibung einer Teilhandlung veranschaulicht, in welcher Form eine Teilsequenz eines Handlungsablaufs durch HL und Basishandlungen modelliert werden kann.

Beispiel 2

Es liegt folgende Beschreibung für den Beginn der Notfallmaßnahme SDE vor:

- Das Kriterium zur Aktivierung der einleitenden Arbeiten für die Notfallmaßnahme SDE ist gegeben, wenn die Füllhöhestände aller vier Dampferzeuger < 4 m sind. Wann das Kriterium ansteht, ist durch den Prozessablauf bestimmt. Die Dampferzeuger-Füllhöhestände (DE-FH) sind in der Warte über die entsprechenden Anzeigen abzulesen. Die Modellierung der Maßnahme beginnt zu dem Zeitpunkt, an dem das Kriterium ‚Füllhöhestände aller 4 Dampferzeuger < 4 m' ansteht.

- Der Leitstandsfahrer (LF), der für das Ablesen der Anzeigen zuständig ist, kann die entsprechenden Anzeigen im Blick haben und sofort erkennen, dass alle vier DE einen Füllhöhestand von < 4 m aufweisen. Er könnte aber auch zufällig abgelenkt und mit anderen Sachen beschäftigt sein, so dass er seine Aufmerksamkeit erst nach einer gewissen Verzögerungszeit wieder auf die Anzeigen der Füllhöhestände richtet. Um diese Zufälligkeiten (aleatorische Unsicherheiten) zu berücksichtigen, wird die Verzögerungszeit, mit der der LF die Anzeigen registriert, als Zufallsvariable betrachtet. Dabei soll der Einfachheit halber angenommen werden, dass die zufällige Verzögerungszeit zwischen 3 s und 20 s liegen kann (In die Abschätzung der Zeitintervalle sollten, soweit vorhanden, Informationen aus der Betriebserfahrung eingehen, um möglichst realistische Verhältnisse in dem Modell abbilden zu können.). Diese Situation des LF soll als erste Basishandlung des Handlungsablaufs spezifiziert werden.

- Nach einer zufälligen Verzögerungszeit zwischen 3 und 20 s steht der LF vor den Pulten des Hauptleitstandes mit den Informations- und Bedieneinrichtungen für die Sekundärseite. Es sind insgesamt acht Füllstandsanzeigen (zwei pro DE) abzulesen. Für das Ablesen der acht Füllstandsanzeigen wird eine Zufallszeit zwischen 16 und 32 s abgeschätzt. Dabei wird angenommen, dass für das Ablesen einer Anzeige eine Ausführungszeit zwischen 2 und 4 s benötigt werden. Diese Aktion soll als zweite Basishandlung in der HL 1 spezifiziert werden.

- Die nächste vom LF durchzuführende Handlung besteht darin, dass der LF den SL darüber informiert, dass Füllhöhestände der DE unter 4 m gesunken sind. Dies ist eine Handlung, die eine Kommunikation zwischen LF und SL beschreibt und über die der SL eine neue Information enthält. Die Handlung hat somit einen Effekt auf den SL, während die beiden vorherigen Aktionen nur den LF betrafen. Es wird angenommen, dass sich der SL in der Nähe des LF in der Warte befindet und sofort verfügbar ist, ohne dass es zu einer Verzögerung kommt, bis die Kommunikation stattfinden kann. Für die Kommunikation wird eine Zeit zwischen 6 und 15 s abgeschätzt. Diese Kommunikationshandlung soll als 3. Basishandlung in die HL eingegeben werden.

Im Folgenden wird dargestellt, wie eine Modellierung der obigen Beschreibung des Ablaufs für das CrewModul durchgeführt wird:

Um eine erste HL zu erzeugen, wird ausgehend vom Eingangsknoten durch die Auswahl der Menüpunkte *Einfügen → Neuer Unterknoten* ein neuer Knoten in der Oberfläche erzeugt. Durch die weitere Auswahl *Bearbeiten → Knoten in einem separaten Editor bearbeiten* öffnet sich in der Oberfläche ein Fenster, in der die entsprechenden Informationen eingegeben werden können. Nach der Eingabe enthält das Modell die in Abb. 5.2 dargestellte Information.

Der obere Teil der Abb. 5.2 zeigt den Editor, in den die Informationen des Knotens eingegeben werden. Der untere Teil zeigt den Knoten nach der Eingabe der Informationen.

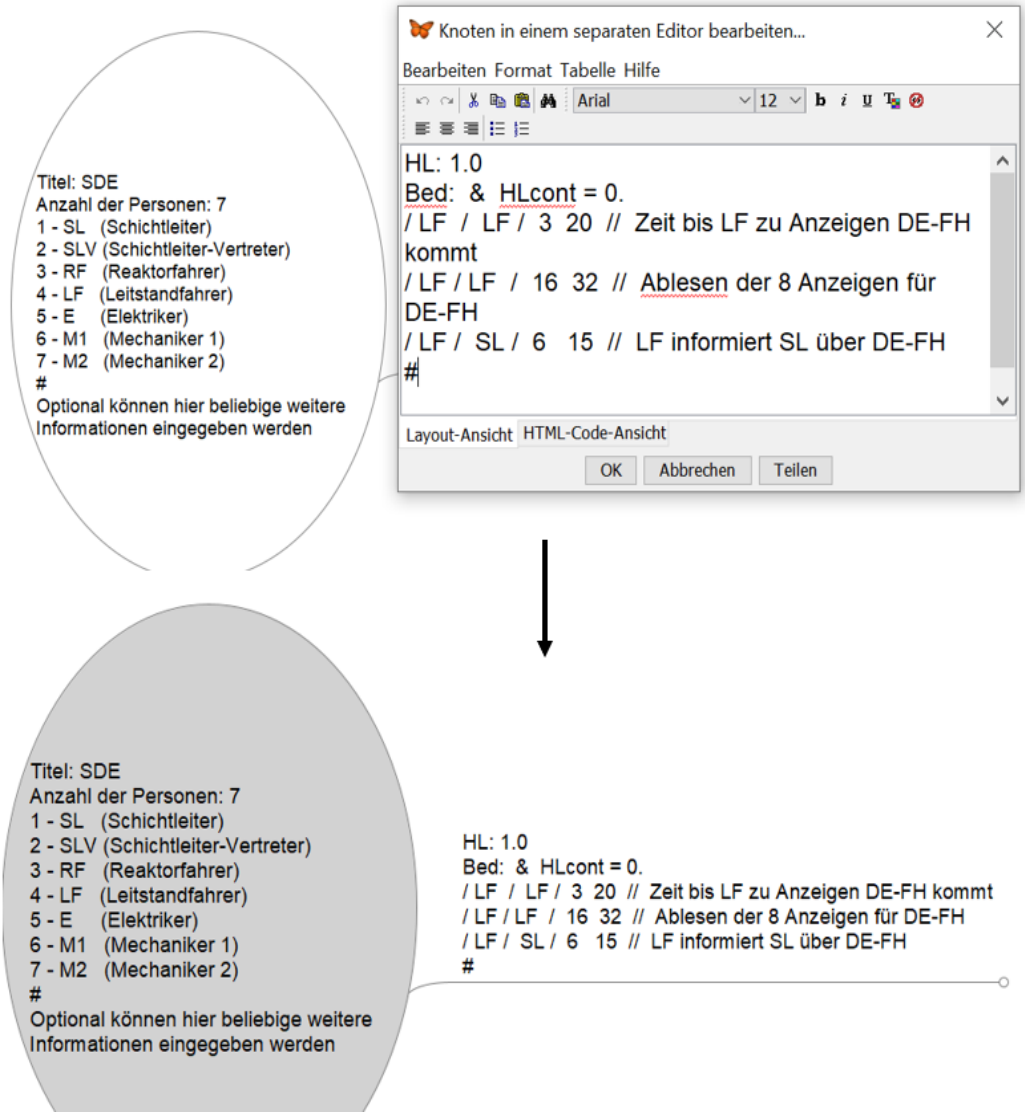


Abb. 5.2 Erzeugung des Knotens für Handlungsliste 1

Da es sich um die erste erzeugte HL handelt, mit der der Beginn der Maßnahme modelliert werden soll, wird der HL die IdNr 1.0 zugeordnet. Um festzulegen, wann die HL aktiviert werden soll und die darin enthaltenen Basishandlungen durch das CrewModul simuliert werden, muss eine Bedingung definiert werden. Zur Aktivierung der HL 1.0 lautet die Bedingung: *Bed: & HLcont = 0.*

Die Bedingung *HLcont = 0* wird so bewertet, dass es keine HL gibt, die vor HL 1.0 ausgeführt wird und auf die HL 1.0 aufsetzen kann. Diese einfache Bedingung reicht aus, um HL 1.0 aktivieren zu können.

Nach der Definition der Bedingung werden die Basishandlungen eingegeben. Der Abb. 5.2 ist zu entnehmen, dass eine Basishandlung nicht unbedingt eine konkrete Aktion

beschreiben muss, sondern auch nur aus einer bestimmten Verzögerungszeit oder Wartezeit bestehen kann.

5.2.3.1 Kommentare

Kommentare können optional durch das Zeichen // eingegeben werden. Die Kommentare dürfen dabei nur im Anschluss der jeweils für das CrewModul-Programm relevanten Eingaben angegeben werden, z. B.

HL: 2.32 // Kommentar
Bed: & HLcont = 1.0 & … // Kommentar
/ SL / RF / 20 45 // Kommentar

5.2.3.2 Ende einer Handlungsliste

Durch das #-Zeichen am Ende der Eingaben einer HL wird die jeweilige HL beendet. Danach können optional noch beliebige Kommentare eingegeben werden. Nach Abschluss einer HL müssen diese nicht mehr zwangsläufig durch ein // gekennzeichnet werden.

5.2.3.3 Ende einer Handlungssequenz

Das Ende einer Handlungssequenz (Sequenz von HL) muss mit einer separaten Handlungsliste HL: -99 gekennzeichnet werden. Für HL: -99 muss in der Bedingung nur noch angegeben werden, nach welcher HL die Sequenz beendet wird.

Angenommen die Handlungssequenz besteht aus den Handlungslisten

HL: 1.0 -> HL: 2.1 -> … -> HL: 12.22

Nach den Basishandlungen der HL: 12.22 soll die Handlungssequenz beendet werden. Dann muss nach HL: 12.22 noch die HL: -99 mit der entsprechenden Bedingungsanweisung angehängt werden, nach welcher HL die Sequenz beendet ist (siehe Abb. 5.3).

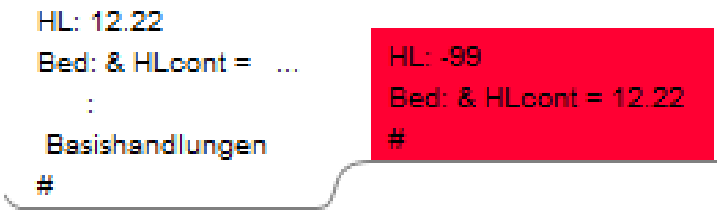


Abb. 5.3 Darstellung wie das Ende einer Handlungssequenz in der MindMap eingefügt werden kann

Optional kann der Abschluss einer Sequenz auch farblich gekennzeichnet. Dies trägt insbesondere bei komplexen Handlungsmodellen zur besseren Übersicht bei.

5.2.4 Verarbeitung der Ausführungszeiten von Basishandlungen

An dieser Stelle soll eine kurze Bemerkung über die Handlungszeiten erfolgen. Jede Basishandlung enthält die Angabe bzgl. ihrer Ausführungszeit. Ist die Ausführungszeit als Zufallsvariable definiert, muss eine Verteilung spezifiziert werden, aus der für jeden Simulationslauf eine zufällig ausgespielte Zeit verwendet wird. Die Spezifikation der Verteilung und die Erzeugung der Zufallswerte für die einzelnen Basishandlungen erfolgt im klassischen CrewModul mit dem Software-Progamm SUSA /KLO 21/.

Angenommen für die erste Basishandlung der HL: 1.0 wurde der Zufallswert t_1 = 10 s, für die zweite Basishandlung der Wert t_2 = 21 s und für die dritte Basishandlung der Wert t_3 = 8.5 s ausgespielt. Die Zeiten der ausgespeilten Basishandlungen werden bei der Simulation aufsummiert. D. h., die Zeiten, wann die einzelnen Basishandlung ausgeführt worden sind, betragen

t_1 = 10 s für Basishandlung 1

$t_1 + t_2$ = 31 s für Basishandlung 2 und

$t_1 + t_2 + t_3$ = 39,5 s für Basishandlung 3.

Die Basishandlung 3, in der der LF den SL informiert, beginnt bei 31 s und ist bei 39.5 s abgeschlossen. Die Variable tLF (intern angelegte Zeitvariable für den LF) hat zu Beginn der Basishandlung 3 den Wert 31, da LF in Basishandlung 1 und 2 beteiligt war. Da der SL in den ersten beiden Basishandlungen nicht involviert war, hat tSL (intern angelegte Zeitvariable für den SL) zu Beginn der Basishandlung 3 den Wert 0. In Basishandlung 3 wird durch die Kommunikation von LF und SL die Zeit des SL mit der des LF synchronisiert. D. h., zu Beginn der Basishandlung 3, in der der LF beginnt, mit dem SL zu

kommunizieren, wird tSL = tLF = 31 s gesetzt. Am Ende der Basishandlung 3, die 8.5 s in Anspruch nimmt, werden dem LF und SL die gleichen Zeiten tLF = tSL = 39.5 s zugeordnet.

Angenommen, der SL führt vor Beginn der Basishandlung 3 eine Aktion durch, die 50 s in Anspruch nimmt. Dann hätte tSL zu Beginn der Basishandlung 3 einen Wert von 50 s. Da tLF = 31 s < tSL = 50 s bedeutet das, dass der SL für die Kommunikation noch nicht zur Verfügung steht, weil er noch mit der vorhergehenden Aktion beschäftigt ist, die erst nach 50 s abgeschlossen ist. In diesem Fall wartet der LF, bis SL seine Aktion beendet hat. In diesem Fall würde die Basishandlung 3 erst bei 50 s beginnen und bei 58.5 s enden.

In analoger Form werden alle Zeiten der jeweiligen Basishandlungen verarbeitet.

5.2.5 Modellierung von Unsicherheiten über Verzweigungsvariable

In diesem Abschnitt wird anhand von Beispielen erläutert, wie Unsicherheiten für das CrewModul modelliert werden. Bei der Beschreibung des Ablaufs in Beispiel 2 (siehe Abschnitt 2.3) wurde angenommen, dass sich der SL in der Nähe des LF in der Warte befindet und sofort verfügbar ist, ohne dass es zu einer Verzögerung kommt, bis die Kommunikation stattfinden kann. Im nachfolgenden Beispiel 3 wird angenommen, dass Unsicherheit darüber besteht, wann der SL in der Warte verfügbar ist, um die Information vom LF entgegenzunehmen.

Beispiel 3

Der Beginn des Handlungsablaufs ist für die ersten beiden Basishandlungen identisch zu dem in Beispiel 2. Zu dem Zeitpunkt, an dem der LF den SL informieren möchte, besteht Unsicherheit darüber, ob und wann der SL für die Kommunikation verfügbar ist. Die Verfügbarkeit hängt davon ab, mit welchen anderen Aufgaben der SL zu diesem Zeitpunkt beschäftigt ist. Es wird hier beispielhaft angenommen, dass der SL mit einer Wahrscheinlichkeit von 0.4 innerhalb von 5 - 30 s, mit einer Wahrscheinlichkeit von 0.5 innerhalb von 30 - 120 s und mit einer Wahrscheinlichkeit von 0.1 innerhalb von 120 - 420 s verfügbar ist (Für eine belastbare Abschätzung könnten Erfahrungswerte aus der Praxis herangezogen werden.).

Unsicherheiten werden über sogenannte Verzweigungsvariablen erfasst. Der Name für die Verzweigungsvariable, über die die Unsicherheit erfasst werden soll, kann vom

Nutzer frei gewählt werden, z. B. verfSL. Obligatorisch ist, dass dem gewählten Namen ein **b**_ vorangestellt sein muss. D. h. die Verzweigungsvariable würde damit b_verfSL heißen.

Nach dem Ablesen der DE-FH durch den LF besteht Unsicherheit darüber, wann der SL verfügbar ist. Dies wird im Modell durch folgende Anweisung angegeben:

/ LF / b_verfSL / 0 0 // *optionale Beschreibung*

Diese Anweisung hat zwar die gleiche Form wie eine Basishandlung, muss aber etwas anders interpretiert werden. Durch das Prefix **b**_ erkennt das Programm, dass es sich um eine Verzweigungsvariable handelt und fügt diese automatisch der Variablenliste (Key-Liste) des CrewModuls hinzu. Die Variablen, die in der Key-Liste enthalten sind, müssen später auch in MCDET spezifiziert werden, um eine Kommunikation zwischen dem CrewModul und MCDET zu ermöglichen. Durch die Kopplung des CrewModuls mit MCDET ist es möglich, den Handlungsablauf in Abhängigkeit der definierten unsicheren Größen zu berücksichtigen.

Um die Unsicherheit zu berücksichtigen, nach welcher Zeit der SL verfügbar ist, kann die Modellierung gemäß Abb. 5.4 erfolgen:

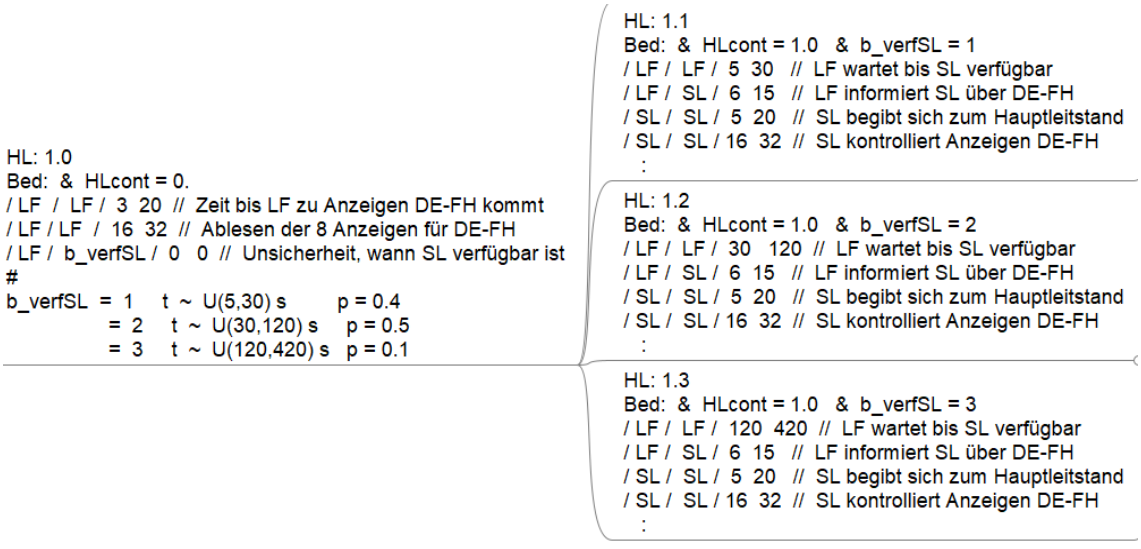


Abb. 5.4 Modellierung des Ablaufs mit Berücksichtigung der Unsicherheit, nach welcher Zeit der SL verfügbar ist

Obwohl die Anweisung */ LF / b_verfSL / 0 0 // Unsicherheit, wann SL verfügbar ist* in HL: 1.0 keine Aktion im eigentlichen Sinne beschreibt, muss dennoch eine Person genannt werden, um den Zeitpunkt der Verzweigung richtig zuordnen zu können. Da die

Unsicherheit auftritt, nachdem der LF die Anzeigen abgelesen hat und den SL darüber informieren möchte, wird der LF als Bezugsperson der Verzweigungsvariablen angegeben. Da es sich bei dieser Anweisung um keine Aktion handelt, wird für die Zeit der konstante Wert 0 (d. h. / 0 0 //) angegeben. Zum besseren Verständnis wurde noch eine Beschreibung gegeben, um welche Unsicherheit es sich handelt. Eine solche Beschreibung hinzuzufügen ist zwar empfehlenswert, jedoch für die Simulationsrechnungen nicht notwendig. Sie dient vor allem der besseren Nachvollziehbarkeit des Modells.

Wichtig ist, dass nach der Angabe einer Verzweigungsvariablen die betreffende HL mit einem # beendet wird. Dies ist deshalb notwendig, da der weitere Ablauf von den Ausprägungen der Verzweigungsvariablen abhängt. Generell muss jede HL mit einem # abgeschlossen werden. Nach dem Abschluss einer HL durch #, können optional noch zusätzliche Bemerkungen hinzugefügt werden. In der HL: 1.0 (siehe Abb. 5.4) wurden z. B. die Unsicherheiten bzgl. der Verfügbarkeitszeiten des SL mit ihren zugeordneten Wahrscheinlichkeiten notiert.

Beispiel 3 (Fortsetzung)

- Die Verzweigungsvariable b_verfSL hat drei Ausprägungen. In diesem Fall sind somit drei unterschiedliche HL zu definieren. Wenn b_verfSL den Wert 1 annimmt, dann beträgt die Wartezeit des LF zwischen 5 und 30 s, bis der SL verfügbar ist.

- Wenn der SL in der Warte verfügbar ist, informiert der LF den SL über die Füllhöhestände der Dampferzeuger, wozu der LF zwischen 6 und 15 s benötigt.

- Nachdem der SL über den Zustand der DE-FH vom LF informiert wurde, begibt er sich zum Pult des Hauptleitstandes, um selbst die Anzeigen der DE-FH zu kontrollieren und bestätigen zu können. Dazu muss sich der SL, der sich irgendwo in der Warte aufhält, zum Pult begeben. Der SL kann sich in unmittelbarer Nähe zum Hauptleitstand befinden, bei der die benötigte Zeit für den Weg zu den Anzeigen relativ klein sein wird. Der SL kann aber auch etwas weiter entfernt sein, so dass er etwas mehr Zeit benötigen wird, um den Weg zum Hauptleitstand zurückzulegen. In Abhängigkeit davon, wo sich der SL in der Warte zufällig aufhält, wird er für den Weg mehr oder weniger Zeit benötigen. Die Zeit für den Weg zum Hauptleitstand wird als Zufallsvariable betrachtet, die zwischen 5 s und 20 s abgeschätzt wird.

- Die Zeit, die der SL für das Ablesen der acht Füllstands anzeigen benötigt, entspricht dem Zeitrahmen, der für den LF abgeschätzt wurde, d. h. zwischen 16 und 32 s.

- Für die Ausprägungen b_verfSL = 2 und b_verfSL = 3 sind die Beschreibungen analog. Der einzige Unterschied in Ablauf besteht darin, dass der LF länger warten muss, bis der SL in der Warte verfügbar ist.

Die Modellierung der obigen Beschreibung inklusive der Unsicherheiten bzgl. der Verfügbarkeit des SL ist in den Handlungslisten HL: 1.1, HL: 1.2 und HL: 1.3 in Abb. 5.3 dargestellt. Um eine gewisse Struktur bei der Vergabe der IdNr für die HL einzuhalten, werden die drei HL, die von HL: 1.0 abzweigen, mit HL: 1.1, HL: 1.2 und HL: 1.3 bezeichnet.

Befindet man sich auf dem Knoten der HL: 1.0 kann durch die Menüpunkte *Einfügen* → *Neuer Unterknoten* ein neuer Knoten für eine neue HL erstellt werden. Knoten lassen sich aber auch einfach kopieren, womit eine erhebliche Einsparung der Schreibarbeit verbunden sein kann. Dies ist z. B. bei den Handlungslisten HL: 1.1 bis HL: 1.3 zu erkennen, bei denen die meisten Eingaben gleich sind und Änderungen nur an bestimmten Stellen vorzunehmen sind, z. B. in der Basishandlung *‚LF wartet bis SL verfügbar‘*. Dort werden in den drei HL die unterschiedlichen Zeiten angegeben, die der LF warten muss, bis der SL in der Warte verfügbar ist. Alle anderen Basishandlungen sind in den drei HL gleich.

Da die Handlungslisten HL: 1.1 bis HL: 1.3 die Fortsetzung des Ablaufs der HL: 1.0 bedeuten, muss in den Bedingungen für alle drei HL die Anweisung *& HLcont = 1.0* enthalten sein. Damit wird ausgedrückt, dass alle drei HL eine Fortsetzung des Handlungsablaufs von HL: 1.0 sind. Da die Fortsetzung des Handlungsablaufs von den Unsicherheiten der Verzweigungsvariablen b_verfSL abhängt, muss in den Bedingungen noch zusätzlich angegeben werden, auf welche Ausprägung der Verzweigungsvariablen sich die jeweilige HL bezieht. Die Bedingungen der drei HL sind der Abb. 5.4 zu entnehmen.

Analog können auch weitere Unsicherheiten, die im Handlungsablauf berücksichtigt werden sollen, modelliert werden. In der Simulation des Handlungsablaufs durch das CrewModul erhalten die Verzweigungsvariablen ihre Werte durch Kommunikation mit MCDET, in dem die Wahrscheinlichkeiten der Verzweigungen spezifiziert werden müssen. Für jede Sequenz werden dem CrewModul automatisch die entsprechenden Werte der Verzweigungsvariablen über MCDET geliefert. Durch MCDET werden die Verzweigungen des Handlungsmodells strukturiert abgearbeitet, so dass alle möglichen Kombinationen von Verzweigungen dem CrewModul übergeben und in der Analyse

berücksichtigt werden. Durch die Kopplung des CrewModuls mit MCDET können somit die Einflüsse aleatorischer Unsicherheiten auf den Handlungsablauf berücksichtigt werden.

5.2.6 Unsicherheiten bzgl. der Aktivierungszeiten von Aufgaben

Häufig kommt es vor, dass der SL für die Ausführung von Aufgabenblöcke (Tasks) bestimmte Personen anweisen muss. Wenn sich die Personen außerhalb der Warte befinden, müssen diese in die Warte zurückgerufen werden. Diejenige Person, die zuerst an der Warte ankommt, wird zur Ausführung der ihr zugeordneten Aufgabe angewiesen. D. h., es kommt zuerst derjenige Aufgabenblock zur Ausführung, dessen Operateur zuerst die Warte erreicht und vom SL die Anweisung erhalten kann. Der weitere Handlungsablauf hängt somit davon ab, welche Person als erstes in der Warte ankommt. Ein solche Situation wird in Beispiel 4 beschrieben und modelliert.

Beispiel 4

Angenommen, die letzte Basishandlung der HL: 5.0 beschreibt die Situation, dass der SL Mechaniker 1 und 2, die sich jeweils an unterschiedlichen Orten der Anlage befinden, in die Warte ruft, um diesen jeweils bestimmte Anweisung zur Durchführung von Aktionen zu geben. Wer zuerst in der Warte ankommt, erhält die Anweisung zuerst und kann dementsprechend früher mit seiner Arbeit beginnen. Wer zuerst in der Warte ankommt ist zufällig. Die Modellierung dieser Unsicherheit kann wie in Abb. 5.5 erfolgen.

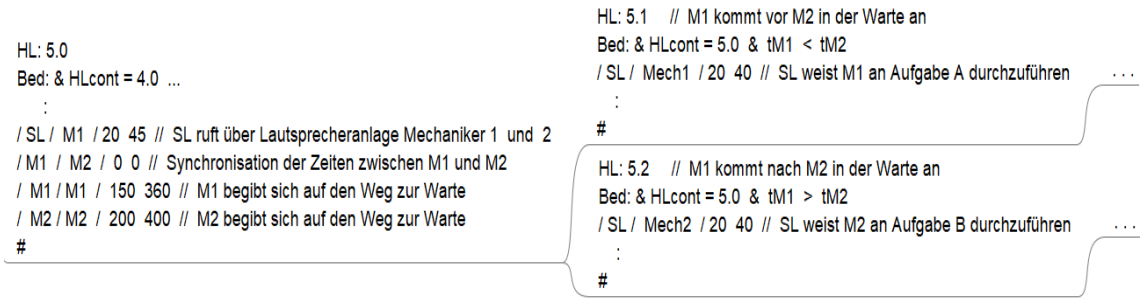


Abb. 5.5 Modellierung der Unsicherheit, ob zuerst Mechaniker M1 oder M2 in der Warte ankommt, um die Anweisungen vom SL entgegenzunehmen

In der Bedingung der HL: 5.1 wird tM1 < tM2 spezifiziert. D. h., die HL wird aktiviert, wenn die Zeit, die M1 für den Weg in die Warte braucht, kleiner ist als die Zeit von M2. Danach erhält M1 die entsprechende Anweisung vom SL und kann beginnen, die dafür notwendigen Aktionen durchzuführen.

Durch die Bedingung der HL: 5.2 wird die HL genau dann aktiviert, wenn die zuletzt ausgeführte Handlungsliste HL: 5.0 ist und M2 früher in der Warte ankommt als M1. Die Zeiten, die die Mechaniker jeweils für den Weg zur Warte benötigen, werden aus den jeweiligen Verteilungen, z. B. Gleichverteilungen $t_1 \sim U(150, 360)$ und $t2 \sim U(200, 400)$, für jeden Simulationslauf zufällig ausgespielt. In diesem Fall ist es nicht so, wie bei den Verzweigungspunkten, bei denen jede Ausprägung der unsicheren Größe gerechnet wird. Sondern je nachdem, wer in einem Simulationslauf die kleinere Zeit aufweist, für den wird die entsprechende HL aktiviert, d. h. für jeden Simulationslauf entweder HL: 5.1 oder HL: 5.2.

In Abb. 5.4 kommt noch eine Besonderheit vor, die näher zu erläutern ist. Aktuell kann in einer Basishandlung nur eine Person (oder eine Kodierung) angegeben werden, die von der Basishandlung beeinflusst wird. In HL: 5.0 kommt jedoch die Basishandlung vor, in der der SL die beiden Mechaniker über die Lautsprecheranlage in die Warte ruft. D. h., sowohl M1 als auch M2 werden gleichzeitig informiert. Behelfsmäßig wird in einem solche Fall für jede Person, die von der Basishandlung betroffen ist, eine separate Basishandlung definiert.

In HL: 5.0 setzt SL seinen Ruf über die Lautsprecheranlage ab und wird zu dem Zeitpunkt von M1 registriert. Da der Ruf aber zur gleichen Zeit auch M2 erreicht, wird eine separate Basishandlung für M2 definiert, in der die Zeiten von M1 und M2 synchronisiert werden. D. h., M1 und M2 erhalten zur gleichen Zeit die Information über die Lautsprecheranlage.

5.2.7 Menschliche Fehler und Recovery Handlungen

In diesem Abschnitt wird die Modellierung von Unsicherheiten bzgl. menschlicher Fehler und von Recovery-Handlungen beschrieben. Dazu ist in Beispiel 5 der zu modellierende Handlungsablauf beschrieben.

Beispiel 5

- Die beteiligten Personen bestehen aus dem SL und dem Reaktorfahrer (RF). Der Handlungsablauf beginnt mit der manuellen Durchführung der Reaktorschnellabschaltung (RESA). Es wird davon ausgegangen, dass die mündliche Genehmigung der Betriebsleitung zur Reaktorabschaltung und die Freigabe der Ausführung durch diensthabenden SL vorliegt. Im Falle einer RESA wird das Notkühlsystem (NKS) nach einer im Reaktorschutzsystem fest eingestellten Verzögerungszeit von 10 s

angefahren. Nach der manuellen RESA und dem automatischen Anfahren des NKS soll überprüft werden, ob die Notkühlpumpen erfolgreich gestartet wurden. Dazu muss der SL dem RF die entsprechende Anweisung geben. Im Handlungsablauf soll die Möglichkeit eines menschlichen Fehlers sowie die einer Recovery-Aktion berücksichtigt werden.

- Der SL wartet die Anfahrzeit des NKS ab. Für die Wartezeit wird eine Zeitspanne zwischen 20 und 40 s abgeschätzt.

- Die nächste Aktion besteht darin, dass der SL die Anweisung zur Überprüfung des Zustands der Notkühlpumpen gibt. Dabei wird angenommen, dass der SL die Anweisung zur Kontrolle der Pumpen vergessen kann. Als Wahrscheinlichkeit für den menschlicher Fehler wird p = 0.003 abschätzt. Der weitere Handlungsablauf erfolgt in Abhängigkeit davon, ob der SL die Anweisung gibt oder nicht.

- Wenn der SL dem RF die Anweisung zur Kontrolle gibt, wozu ein Zeitbedarf zwischen 10 und 20 s abgeschätzt wird, wird das NKS vom RF überprüft. Zur Überprüfung des NKS wird ein Zeitbedarf zwischen 150 und 250 s angenommen.

- Wenn der SL vergisst, die Anweisung zu geben, besteht die Möglichkeit, dass Fehler innerhalb einer Zeitspanne zwischen 10 und 40 s bemerkt wird, und der SL die Anweisung nachholt (Recovery-Aktion). Es wird angenommen, dass der Unterlassungsfehler mit einer Wahrscheinlichkeit von p = 0.8 bemerkt wird. Die Abschätzung der menschlichen Fehlerwahrscheinlichkeiten kann z. B. nach THERP /SWA 83/ oder ASEP SWA 87 erfolgen.

- Wird der Unterlassungsfehler nicht bemerkt, werden die nachfolgenden Aktionen weiter ausgeführt, obwohl eine oder mehrere Notkühlpumpen nicht laufen.

- Bei der Überprüfung des NKS hängt der weitere Handlungsablauf von der aleatorischen Unsicherheit ab, ob das NKS verfügbar ist oder nicht. Diese aleatorische Unsicherheit wird durch die Verzweigungsvariable b_verfNKS (siehe Abb. 5.5) beschrieben. Wenn mindestens eine von drei Pumpen nicht erfolgreich gestartet werden kann, wird das NKS als nicht verfügbar betrachtet. Werden alle drei Pumpen gestartet, ist das NKS verfügbar. Als Wahrscheinlichkeit für die Verfügbarkeit des NKS wird 0.91 abgeschätzt. Für die Abschätzungen von Komponenten- oder Systemverfügbarkeiten können Werte aus der klassischen PSA verwendet werden.

- Wenn das NKS verfügbar ist, (p = 0.91) wird der Handlungsablauf fortgeführt.

- Ist das NKS nicht verfügbar (p = 0.09), wird die Maßnahme als nicht erfolgreich abgebrochen.

Zur besseren Veranschaulichung ist in Abb. 5.6 zunächst die Verzweigungsstruktur obiger Beschreibung unter Berücksichtigung der Unsicherheiten dargestellt. Danach erfolgt die Beschreibung der Inhalte der einzelnen HL.

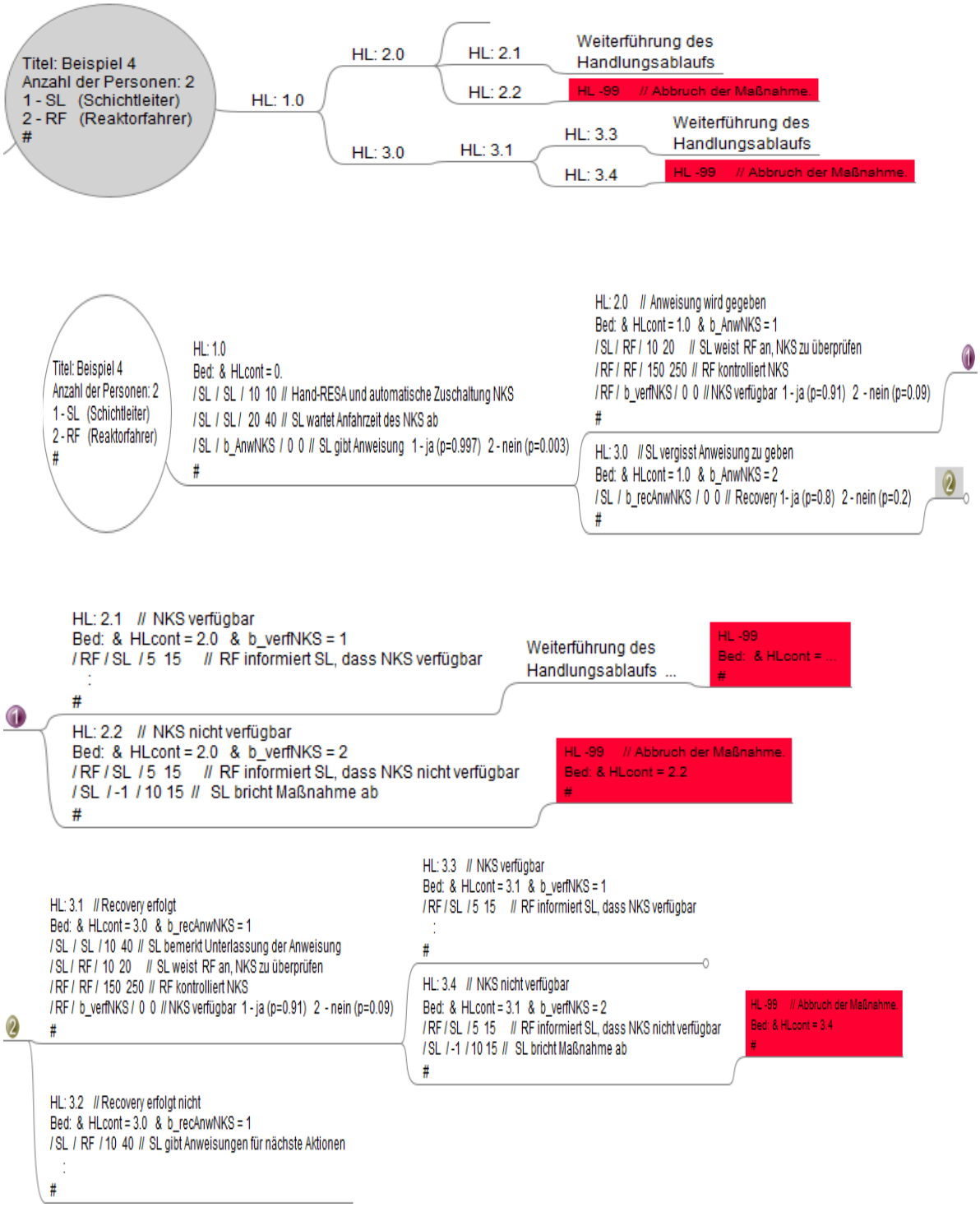


Abb. 5.6 Verzweigungsstruktur und Inhalte der Handlungslisten zu Beispiel 5

In der Beschreibung des Beispiels 5 werden drei unsichere Größen definiert, die im Modell über die Verzweigungsvariable b_AnwNKS, b_recAnwNKS und b_verfNKS berücksichtigt werden.

In HL: 1.0 tritt die Verzweigungsvariable b_AnwNKS auf, die sich auf die Unsicherheit bezieht, ob der SL die Anweisung zur Überprüfung von NKS gibt oder die Anweisung fälschlicherweise unterlässt. Für die Verzweigungsvariable werden zwei Ausprägungen definiert,

1 – SL gibt Anweisung und 2 – SL unterlässt Anweisung.

Um den von HL: 1.0 fortgesetzten Handlungsablauf in Abhängigkeit der Unsicherheiten von b_AnwSL zu modellieren, müssen zwei HL modelliert werden, die von HL: 1.0 abzweigen. Die beiden von HL: 1.0 abzweigenden HL sind HL: 2.0 und HL: 3.0.

Die HL: 2.0 bezieht sich auf den durchzuführenden Handlungsablauf für den Fall, dass der SL die Anweisung gibt (Dies kann optional als Kommentar hinter dem //-Zeichen hinzugefügt werden). Da HL: 2.0 die Fortsetzung von HL: 1.0 beschreibt, wenn der SL die Anweisung gibt – dies wird durch b_AnwSL = 1 gekennzeichnet – lautet die Bedingung zur Aktivierung von HL: 2.0

Bed: & HLcont = 1.0 & bAnwSL = 1

Nach Spezifikation der Bedingung, erfolgt die Eingabe der entsprechenden Basishandlungen. Diese beziehen sich auf die Aktionen, dass

- SL dem RF die Anweisung gibt, das NKS zu überprüfen und
- der RF das NKS kontrolliert.

Der Weitere Handlungsablauf hängt davon ab, ob der RF bei der Überprüfung feststellt, ob das NKS verfügbar ist oder nicht. Diese Unsicherheit wird durch die Verzweigungsvariable b_verfNKS modelliert.

Von der HL: 2.0 zweigen die HL: 2.1 und HL: 2.2 in Abhängigkeit davon ab, ob das NKS verfügbar ist. Diese Unsicherheit wird durch die Verzweigungsvariable b_verfNKS gekennzeichnet. Im Fall b_verfNKS = 1 ist das NKS verfügbar und es folgen in HL: 2.1 die entsprechenden Basishandlungen zur Modellierung des weiteren Handlungsablaufs, bis diese Handlungssequenz beendet ist.

Es ist wichtig, dass das Ende einer Handlungssequenz mit der HL: -99 gekennzeichnet wird. Die HL: -99 enthält die Bedingung, die angibt, bei welcher Handlungsliste die Sequenz beendet wird.

In HL: 2.2 wird der Ablauf beschrieben, wenn das NKS nicht verfügbar ist (b_verfNKS = 2). In diesem Fall informiert der RF, der das NKS überprüft hat, den SL, dass das NKS nicht verfügbar ist. Für diese Information wird ein Zeitbedarf zwischen 5 und 15 s abgeschätzt, wobei angenommen wird, dass sich der SL in der Nähe des RF befindet. Daraufhin gibt der SL die Anweisung zum Abbruch der Maßnahme, wobei in Zeitbedarf zwischen 10 und 15 s angenommen wird. Danach wird die HL: 2.2 durch ein # beendet. An dieser Stelle sieht man, dass eine HL nicht zwangsläufig nur bei einer Verzweigungsvariablen beendet werden kann.

Nach Beendigung der HL: 2.2 ist die Handlungssequenz, die aus den HL: 1.0 -> HL: 2.0 -> HL: 2.2 besteht, beendet. Das Ende einer Handlungssequenz wird durch eine separate HL mit der Kennzeichnung HL: -99 angegeben. Als Bedingung von HL: -99 muss die HL angegeben werden, bei der die Sequenz beendet wird. In diesem Fall wird als Bedingung *Bed: & HLcont = 2.2* angegeben (siehe Abb. 5.6).

Die HL: 3.0 bezieht sich auf den durchzuführenden Handlungsablauf für den Fall, dass der SL die Anweisung unterlässt. Dieser Fall wird durch die Verzweigungsvariable b_AnwSL = 2 beschrieben. Des Weiteren ist die HL: 3.0 die unmittelbare Fortsetzung von HL: 1.0, wenn SL die Anweisung unterlässt. Die Bedingung zur Aktivierung von HL: 3.0 lautet deshalb:

Bed: & HLcont = 1.0 & bAnwSL = 2

Wenn der SL die Anweisung vergisst, besteht die Möglichkeit, dass der Fehler des SL erkannt wird und dieser durch eine Recovery-Aktion behoben wird. Es besteht aber auch die Möglichkeit, dass der Fehler nicht erkannt wird. Diese Unsicherheit wird durch die Verzweigungsvariable b_recAnwNKS beschrieben.

Da der weitere Handlungsablauf in HL: 3.0 davon abhängt, ob der Fehler erkannt wird oder nicht, wird in HL: 3.0 lediglich die entsprechende Verzweigungsvariable durch die Anweisung

/ SL / b_recAnwNKS / 0 0 // …

gesetzt und die HL mit einem # abgeschlossen.

Die HL: 3.1 und HL: 3.2 beziehen sich auf die Abläufe in Abhängigkeit davon, ob der Unterlassungsfehler bemerkt wird und die Recovery-Aktion durchgeführt wird oder nicht. Die Interpretationen der Inhalte der weiteren HL sind analog zu den bisherigen Beschreibungen.

5.2.8 Kennzeichnung auszuwertender relevanter Aktionen

Bei der Erstellung des Handlungsmodells müssen diejenigen Aktionen gekennzeichnet werden, die im Post-Processing der Simulationsergebnisse ausgewertet werden sollen. Die Auswertung der Simulationsergebnisse des CrewModuls besteht darin, Wahrscheinlichkeitsverteilungen bzgl. des Zeitpunkts zu ermitteln, wann die entsprechenden Aktionen durchgeführt werden. Um diejenigen Aktionen zu kennzeichnen, für die im Rahmen der Auswertung die Zeitverteilungen ermittelt werden sollen, muss in dem Attribut *‚AuswirkungAufWenOderWas‘* der entsprechenden Basishandlung eine negative Zahl n, n ∈ {-1, …, -90} eingegeben werden.

Beispiel 6

Bei dem in Abb. 5.6 modellierten Handlungsablauf soll eine Verteilung des Zeitpunktes ermittelt werden, wann die Maßnahme aufgrund der Nichtverfügbarkeit des NKS abgebrochen wird. Die Basishandlung, in der der SL die Maßnahme beendet, tritt einmal in HL: 2.2 auf, in der die Nichtverfügbarkeit des NKS ohne vorhergehenden menschlichen Fehler festgestellt wird und in HL: 3.4 in der die Nichtverfügbarkeit des NKS erst im Rahmen einer Recovery-Aktion nach dem Auslassungsfehler des SL festgestellt wurde. Die Basishandlung für den Abbruch der Maßnahme lautet:

/ SL / -1 / 10 15 // SL bricht Maßnahme ab

Da der Zeitpunkt ausgewertet werden soll, wann die Maßnahme abgebrochen wird, steht in dem entsprechenden Attribut der Basishandlung die negative Zahl -1.

Eine einmal vergebene Kodierung, ist dann in dem Handlungsablauf genau der jeweiligen Aktion eindeutig zugeordnet. So wird die Kodierung -1 im obigen Beispiel für den Abbruch der Maßnahme aufgrund der Nichtverfügbarkeit des NKS verwendet. Der Nutzer hätte der Aktion auch eine andere negative Zahl zuordnen können, solange diese noch nicht für eine andere Aktion verwendet worden ist.

Relevante Handlungen, die in der Auswertung der Simulationsergebnisse berücksichtigt werden sollen, beziehen sich im Wesentlichen auf solche Aktionen, die einen Einfluss

auf den zugrundeliegenden physikalischen Prozessablauf haben, z. B. Abschalten von Pumpen, Öffnen von Ventilen etc.

Die berechneten Zeitverteilungen der gekennzeichneten Aktionen eines Handlungsablaufs können dann später als unsichere stochastische Größen (aleatorische Unsicherheiten) in die MCDET-Analyse des physikalischen Prozesses intergiert werden. Dazu wird MCDET mit dem deterministischen Rechenprogramm gekoppelt, der den physikalischen Prozess simuliert.

5.3 Erstellung des Eingabedatensatzes für das CrewModul

Die Modellierung des Handlungsablaufs in der Oberfläche des CrewModuls folgt einer bestimmten Struktur. Diese Struktur erlaubt es, dass anhand des Inhalts der Oberfläche die notwendigen Eingabedateien für das CrewModul automatisch erstellt werden können. Ist die Modellierung des Handlungsablaufs in der Oberfläche abgeschlossen, wird durch Auswahl der Menüpunkte der Oberfläche

Datei -> Export -> Als HTML

der Inhalt des in der Oberfläche modellierten Ablaufs in Textform dargestellt. Damit alle Knoten (bzw. HL) gelesen werden können, ist zuvor zu gewährleisten, dass alle Knoten vollständig aufgeklappt (sichtbar) sind. Vor dem obigen Aufruf kann dies einfach durch die Auswahl der Menüpunkte

Navigieren -> Alles aufklappen

erfolgen.

Aktuell wird der in der HTML-Datei dargestellte Text in eine Textdatei mit der Extension .txt kopiert, z. B. *FileName.txt*.

Zum Lesen und zur automatischen Erstellung der Eingabedateien für das CrewModul wurde ein separates Python-Programm *‚readMindMap.py‘* geschrieben. Bei der Anwendung des Programms in einem geeigneten Python-Editor (z. B. Spyder) ist zur Erstellung der Eingabedateien lediglich der Name des Textfiles *FileName* (ohne Extension) anzugeben. Die Eingabedateien beinhalten

- die im Handlungsablauf definierten Basishandlungen mit ihren Attributen,

- die definierten Handlungslisten mit ihren Bedingungen und Abfolgen von Basishandlungen,
- die relevanten Variablen, die für die Kommunikation und Steuerung zwischen dem CrewModul und MCDET notwendig sind. Dies sind z. B. die im Handlungsablauf definierten Verzweigungsvariable oder die automatisch vom CrewModul angelegte Variable HLcont und
- Kodierungen, wann interne Zustandsänderungen von Variablen erfolgen.

Alle nachfolgend beschriebenen Eingabedatensätze werden anhand der Informationen der grafischen Oberfläche automatisch erstellt. Obwohl der Nutzer den Inhalt der Inputs nicht kennen muss, soll der Vollständigkeit halber dennoch eine kurze Beschreibung der erzeugten Input-Dateien gegeben werden.

5.3.1 Datei der Basishandlungen

Alle Basishandlungen, in die ein komplexer Handlungsablauf zerlegt worden ist, werden in einer separaten Datei *(‚cBasAct.scl'*) aufgelistet, wobei für jede einzelne Basishandlung die Informationen der oben beschriebenen Attribute aufgeführt werden. Ein Ausschnitt für eine Datei der Basishandlungen ist in Tab. 5.1 dargestellt.

Tab. 5.1 Struktur der Datei der Basishandlungen des Handlungsablaufs

```
1001  1  1  #1  1.0  5.0
1002  1  1  #2  180.0  600.0
1003  1  4  #3  40.0  120.0
1004  1  4  #4  30.0  120.0
1005  1  -111  1.0  1.0  1.0
1006  1  4  #5  60.0  300.0
1007  1  1  #6  10.0  30.0
1008  1  1  #7  10.0  30.0
1009  1  -112  1.0  1.0  1.0
1010  1  2  #8  15.0  60.0
2001  2  2  #9  5.0  15.0
2002  2  2  #10  4.0  16.0
2003  2  2  #11  1.0  4.0
          :
```

Die Datenstruktur in Tab. 5.1 ist gegeben durch:

- Spalte 1: Identifikationsnummer der Basishandlung (wird intern festgelegt)
- Spalte 2: Indexnummer der Person, die Handlung durchführt

- Spalte 3: Indexnummer der Person, die von der Handlung beeinflusst wird. Negative Indexnummern < 100 geben besondere Ereignisse an, z. B., ob der Zustand einer Komponente geändert wird, wann eine bestimmte Handlung durchgeführt wird die ggf. Einfluss auf den Prozessablauf hat oder wann eine Verzweigung des Handlungsablaufs stattfindet.

- Spalte 4: Durch das Zeichen ‚#' wird angedeutet, dass es sich bei der Ausführungszeit der Basishandlung um eine Zufallsgröße handelt. Die darauffolgende Zahl gibt die Nummer der unsicheren Größe an. Die Nummern werden zur richtigen Zuordnung der in SUSA simulieren Werte der Ausführungszeiten zu den entsprechenden Basishandlungen benötigt, die im CrewModul verarbeitet werden. Die nicht mit # gekennzeichneten Werte bezeichnen die konstanten Ausführungszeiten. Die negative Werte < -100 kennzeichnen die im Handlungsablauf vorkommenden Verzweigungsvariablen. Die Nummern werden den Verzweigungsvariablen intern zugewiesen.

- Spalte 5 und 6: Minimum und Maximum der Ausführungszeit der jeweiligen Basishandlung.

5.3.2 Datei der Handlungslisten

Die Basishandlungen, in die ein Handlungsablauf zerlegt worden ist, werden sequentiell zusammengefügt, um bestimmte Teilhandlungen zu beschreiben. Diese sequentielle Abfolge von Basishandlungen wird als eine HL bezeichnet. Eine HL (Sequenz von Basishandlungen) endet dann, wenn der weitere Handlungsablauf von bestimmten Bedingungen abhängt, die z. B. durch Prozesszustände oder zufälligen Ereignissen gegeben sind. Eine HL ist definiert durch eine

- eindeutige IdNr,

- eine Bedingung, wann die jeweilige Handlungsliste aktiviert wird, und

- die Angabe der IdNr von Basishandlungen, durch die die Teilhandlung der HL beschrieben wird.

Eine HL wird dann aktiviert und ausgeführt, wenn die ihr zugeordnete Bedingung erfüllt ist. Dazu muss die einer HL zugeordnete Bedingung eindeutig sein. Die Spezifikation einer Bedingung kann durch beliebige Parameter erfolgen. Durch die Zuordnung einer speziellen Bedingung zu jeder Handlungssequenz ist es möglich, Handlungsabläufe z. B. in Abhängigkeit

- von den dynamischen Entwicklungen des Prozess- und Systemzustands,
- vom Zustand des bisher erfolgten Handlungsablaufs,
- von kognitiven und ergonomischen Faktoren und
- von stochastischen Einflussfaktoren

zu berücksichtigen.

Alle HL, die zur Beschreibung eines Handlungsablaufs definiert werden, werden in einer separaten Datei (*‚cActLst.inp'*) in einer bestimmten Struktur abgelegt, die in Tab. 5.2 beispielhaft dargestellt wird.

Tab. 5.2 Struktur der Datei der Handlungslisten

```
1.0  5  1001  1002  1003  1004  1005  2  3  =  0.0  2  =  0.0
1.1  1  1006  2  3  =  1.0  11  <=  1.0
1.12  3  1007  1008  1009  1  3  =  1.1
1.2  8  1010  2001  2002  2003  2002  2003  2004  2005  2  3  =  1.12  12  <=  1.0
1.2  8  1010  2001  2002  2003  2002  2003  2004  2005  2  3  =  2.21  12  <=  1.0
1.21  6  2006  2007  4001  2008  2009  2010  2  3  =  1.2  13  <=  1.0
1.21  6  2006  2007  4001  2008  2009  2010  2  3  =  2.31  13  <=  1.0
1.21  6  2006  2007  4001  2008  2009  2010  2  3  =  2.33  13  <=  1.0
1.31  6  2011  2012  4002  2013  2014  2015  2  3  =  1.21  14  <=  1.0
1.31  6  2011  2012  4002  2013  2014  2015  2  3  =  1.231  14  <=  1.0
1.41  2  2016  2017  2  3  =  1.31  15  <=  1.0
          :
```

Die Datenstruktur in Tab. 5.2 liegt in kodierter Form vor und ist gegeben durch:

- Spalte 1: IdNr der HL. Die Nummern der HL können als Gleitkommazahlen z. B. 1.1, 1.124 etc. definiert werden, um eine bessere Strukturierung der HL zu ermöglichen.
- Spalte 2: Gibt die Anzahl der Basishandlungen der jeweiligen HL an. Danach folgen die IdNr der Basishandlungen, aus denen sich die HL zusammensetzt.

Nach den IdNr der Basishandlungen folgen die Bedingungen der jeweiligen HL in kodierter Form.

5.3.3 Variablenliste

In der Datei *‚cKeyList.inp'* sind alle wesentlichen Variablen enthalten, die für den Simulationsablauf und die Kommunikation zwischen CrewModul und MCDET benötigt werden. Alle Variablen der Liste werden automatisch erzeugt. Ein Beispiel einer solchen Variablenliste ist in Tab. 5.3 dargestellt.

Tab. 5.3 Variablenliste

```
4
1 Time
2 ALId
3 HLcont
4 Code
5 tSL
6 tM1
7 tM2
8 tEL
10 ALW
11 b_erkFhlAnr
12 b_stresSL
    :
```

Die erste Spalte enthält die Nummer der Variablen, die zweite Spalte die Variablennamen.

Die erste Zeile enthält die Anzahl der am Handlungsablauf beteiligten Personen. Die ersten vier Variablen sind:

- **Time** – Zeitpunkt, an dem sich der Handlungsablauf befindet
- **ALId** – Indikator für bestimmte Alarme
- **HLcont** – enthält die IdNr der HL. Die Variable HLcont wird für die Definition von Bedingungen der HL benötigt und gibt die IdNr der vorhergehenden HL an, auf der die aktuelle HL aufsetzt.
- **Code** – Kodierungsnummer der relevanten Handlung

Danach folgen die Zeitvariablen für die beteiligten Personen, die angeben, wann die jeweiligen Personen für nächste Aktionen zur Verfügung stehen. Für die Namen der Zeitvariablen werden die Kürzel der jeweiligen Personen mit einem vorangestellten ‚t' verwendet.

Anschließend folgen die Verzweigungsvariablen. Durch die Kopplung des CrewModuls mit MCDET erhalten diese Verzweigungsvariablen die von MCDET übermittelten Ausprägungen der unsicheren Größen.

Alle Variablen können je nach Bedarf in den Bedingungen von HL verwendet werden. Dies wurde anhand der Beispiele in Abschnitt 5.2.5 und 5.2.6 demonstriert.

6 Zusammenfassung und Ausblick

Zur Fortführung der Weiterentwicklung des Analysewerkzeugs MCDET wurden im Vorhaben RS1570 Arbeiten

- zur Erweiterung des Methodenspektrums für MCDET-Analysen,
- zur Weiterentwicklung des Scheduling-Systems zum flexiblen und effizienten Einsatz von MCDET sowie
- zur Weitentwicklung der interaktiven Anwendung von MCDET

durchgeführt.

Die Entwicklungen zur Erweiterung des Methodenspektrums beziehen sich zum einen auf rein mathematisch-statistische Methodenentwicklungen für die Auswertung von MCDET-Ergebnissen und zum anderen auf Arbeiten, mit denen die Einsatzmöglichkeiten von MCDET-Analysen möglichst effizient erweitert werden können.

Zur effizienten Erweiterung der Einsatzmöglichkeiten von MCDET wurden Methoden und Konzepte entwickelt,

- um eine bestehende MCDET-Analyse im Nachhinein um zusätzliche Fragestellungen erweitern zu können, ohne die gesamten aufwändigen Rechenläufe nochmal durchführen zu müssen. Die nachträgliche Erweiterung kann darin bestehen, dass einige Sequenzen verlängert werden (d. h. späterer Endzeitpunkt der Rechenläufe), um auch ihren späteren Verlauf bewerten zu können. Auch können nachträglich zusätzliche stochastische Einflussgrößen berücksichtigt werden, was eine Erweiterung von bestehenden DETs durch zusätzliche Sub-DETs bedeutet. Darüber hinaus kann die Erweiterung auch darin bestehen, dass nachträglich der Einfluss von epistemischen Unsicherheiten bestimmt werden kann. Dabei wird angenommen, dass sich die epistemischen Unsicherheiten nur auf die Pfadwahrscheinlichkeiten auswirken, so dass diese durch eine Neuberechnung der Sequenzwahrscheinlichkeiten berücksichtigt werden können,
- um die Reproduzierbarkeit einer MCDET-Analyse zu gewährleisten,
- um auch eine einfache Monte-Carlo Simulation ohne die Erzeugung von DETs durchführen zu können.

Im Rahmen der mathematisch-statistischen Entwicklungsarbeiten wurden folgende zwei Methoden entwickelt:

- eine Methode zur Bewertung von Einflussgrößen auf Ergebniscluster (ASPIC),
- eine Methode zur Identifikation von Primimplikanten aus den Ergebnissen von MCDET-Rechenläufen und zur Berücksichtigung der identifizierten Primimplikanten in einer klassischen PSA.

Die Methode ASPIC wurde mit dem Ziel entwickelt, dass man unter Verwendung einer gegebenen Stichprobe von Beobachtungswerten diejenigen Parameter und Parameterwerte bestimmen kann, durch die sich aufgetretene Sequenzcluster möglichst gut erklären lassen. Die Methode ist an einem ausgewählten Anwendungsbeispiel erfolgreich erprobt worden und hat zu zufriedenstellenden Ergebnissen geführt.

Mit der Methode ASPIC, die in einem Jupyter-Notebook implementiert wurde, kann auf der Basis von Entropie-Maßen derjenige Parameter ermittelt werden, bei dem ein maximaler Informationsgewinn erzielt werden kann. Mit dieser Information wird zunächst eine univariate Diskriminanzregel hergeleitet. Diese wird nach der Ermittlung des optimalen Diskriminanzwertes für jeden Parameter auf eine multivariate Diskriminanzregel erweitert, die eine Zuordnung eines Parametervektors zu einem Cluster ermöglicht. Für die Diskriminanzregel wird schließlich die damit verbundene Fehlerwahrscheinlichkeit inklusive der Schätzunsicherheit ermittelt.

Die mit der Methode ASPIC erzielten Ergebnisse können auch zur Prognose darüber verwendet werden, in welchem Cluster eine Sequenz auf der Basis von vorgegebenen Parameterwerten mit einer gewissen Wahrscheinlichkeit liegen wird. In künftigen Arbeiten ist zu untersuchen, ob die Diskriminanzregeln anhand vorliegender Stichprobenwerte weiter verfeinert werden können, damit die Genauigkeit zur Erklärung der Cluster weiter erhöht werden kann. Des Weiteren ist zu untersuchen, ob die für die Methode erforderliche Intervalleinteilung der Parameterbereiche einen signifikanten Einfluss auf die Ergebnisse hat.

Wichtige Schritte bzgl. der entwickelten Methode zur Identifikation von Primimplikanten bestehen in der Extraktion der relevanten Einflussgrößen und Zeitreiheninformationen, aus denen die Primimplikanten ermittelt werden und in der Diskretisierung der extrahierten stetigen Einflussgrößen. Für die Diskretisierung und Auswahl der relevanten Einflussgrößen wird der maschinelle Lernalgorithmus des Random-Forest-Klassifikators

eingesetzt. Auf die ermittelten Implikanten wird ein erweiterter Primimplikantenalgorithmus eingesetzt. Dieser Algorithmus wurde in einem Jupyter-Notebook implementiert und seine Anwendung exemplarisch durchgeführt. Am Beispiel der MCDET/ATHLET-Analyse für ein thermisch-induziertes Heizrohrleck (DEHEIRO) /PES 18/ wurde gezeigt, wie die neue Methode der Primimplikantenanalyse für MCDET angewendet werden kann. Durch die Implementierung der Methodik in ein Jupyter-Notebook kann der Primimplikantenalgorithmus vom Nutzer schrittweise und interaktiv verfolgt werden.

Des Weiteren wurde ein Konzept erarbeitet, durch das über die Nutzung des erweiterten GRSSoftware-Werkzeuges pyRiskRobot /BER 16/, /BER 17/ die gefundenen Implikanten automatisiert in eine klassische PSA integriert werden können. Dieser Ansatz eignet sich besonders, um den Mehrwert einer dynamischen PSA gegenüber einer klassischen PSA herauszuarbeiten, da die Ergebnisse der dynamischen PSA zurück in eine klassische PSA überführt werden und so vergleichbar werden. Hier könnte man weitergehend exemplarische Studien durchführen.

Die Weiterentwicklungen des Scheduling-Systems beziehen sich auf eine Vielzahl verschiedener Themen. So wurde der MCDET-Kern verbessert, um eine Erzeugung von Zwillingspfaden mit identischen Simulationspfaden zu vermeiden. Der nur einmal gerechnete Simulationspfad wird mit der Summe der Wahrscheinlichkeiten aller Zwillingspfade bewertet. Eine weitere Verbesserung besteht darin, dass nun auch Zustandsänderungen für Gruppen von Rechencodegrößen definiert werden können. Vorher mussten dafür aufwändige und manchmal auch komplizierte Eingaben gemacht werden, um das gewünschte Ergebnis zu erhalten.

Ein wichtiges Ziel der Weiterentwicklungen ist die Nachrüstung der generischen Treiberschnittstelle, um auch Störfallszenarien mit epistemischen Unsicherheiten zu unterstützen, welche veränderliche Eingabedateien erfordern. Der bisher erreichte Entwicklungsstand hierzu erlaubt die Verwendung automatisiert variierter Rechencode-Eingabedateien und ihre Zuordnung (über Indizes) zu parallelen Scheduler-Läufen. Dadurch können die Variationen bzgl. der Epistemik bereits in allen Startpunkten der MCDET-Rechenläufe durchgeführt werden.

Durchgeführte Konzeptentwicklungen beziehen sich auf die Kopplung von MCDET mit ‚closed-source' Rechenprogrammen, die Erweiterung der Zugriffsmöglichkeiten von ATHLET und ATHLET-CD Größen, sowie die Nutzung eines Linux-Clusters oder von GitLab Runner für MCDET Anwendungen. Ein Teil der Konzepte wurde bereits umge-

setzt und erfolgreich erprobt, wie z. B. der Zugriff auf ATHLET-Tabellenwerte oder der Einsatz von GitLab-Runnern.

Ziel der Umstrukturierung des klassischen CrewModuls ist, einen Python-basierten Rechencode für menschliche Handlungen zu schaffen, der mit dem neuen MCDET Scheduler interagieren kann. Dafür wurde der bisher in Fortran gehaltene Code des CrewModuls in einen Python-Code umgeschrieben, überarbeitet und modular in Form einer Klassenstruktur aufgesetzt. Dies erleichtert spätere Erweiterungen und Verbesserungen. Es wurden Restart-Möglichkeiten eingerichtet sowie ein Simulations-Controller und ein Simulations-Treiber für den CrewModul-Rechencode geschrieben. Durch die Weiterentwicklungen kann das CrewModul nun auch mit dem neuen Scheduler von MCDET verbunden und gesteuert werden. Für die Zukunft ist beabsichtigt, Weiterentwicklungen am Input vorzunehmen, um diesen benutzerfreundlicher und weniger fehleranfällig zu gestalten sowie zusätzliche Modellierungsmöglichkeiten zu schaffen. Des Weiteren soll das CrewModul so entwickelt werden, dass es über MCDET in Interaktion mit einem Rechencode wie z. B. ATHLET arbeiten kann.

Eine sehr wichtige Weiterentwicklung zur interaktiven Anwendung von MCDET ist das neue MCDET-Eingabeformat auf Python-Basis. Damit wird der MCDET-Nutzer auch bei schwierigeren probabilistischen Spezifikationen durch eine standardisierte Syntax und eine klare Eingabestruktur unterstützt. Die bisherigen Limitierungen bei der Definition von mathematischen Ausdrücken oder Funktionen für den Trigger einer MCDET-Aktion wurden behoben und auch komplexe Ausdrücke ermöglicht.

Für das Ziel der Bereitstellung einer grafischen Benutzeroberfläche wurde eine stabile Ausführungsumgebung für MCDET entwickelt. Sowohl MCDET-Läufe als auch die Ergebnisauswertungen im Post-Processing erfordern das Zusammenspiel einer Vielzahl von Bibliotheken und Python-Bibliotheken. Um reproduzierbares Verhalten und somit nachvollziehbare Ergebnisse zu erzielen, sind einheitliche Zusammenstellungen der einzelnen Komponenten und ihrer Versionen unerlässlich. Die automatisierte Einrichtung der Python-Environments hat mittlerweile einen sehr stabilen Stand erreicht, so dass diese sowohl für die Erstellung von Entwicklungs- und Anwendungsumgebungen als auch zur Einrichtung von CI (Continuous Integration)-Testumgebungen verwendet werden kann und für einheitliche Ablaufbedingungen sorgt.

Außerdem wurde im Hinblick auf die grafische Benutzeroberfläche die Möglichkeit zur Visualisierung der DET-Topologie während der Berechnung geschaffen, wobei Stand

und aktuell bearbeitete Ereignispfade, während der DET-Berechnungen fortlaufend dargestellt werden. Die Visualisierung zeigt eine deskriptive Darstellung der DET-Struktur und der aufgetretenen Ereignisse. Des Weiteren wurde ein Prototyp zur interaktiven Überwachung und Kontrolle entwickelt. Dieser zeigt auch die Topologie der Simulationsprozesse.

Ein wichtiger Bestandteil von Softwareprodukten sind Benutzerführungen bzw. -handbücher. In diesem Zusammenhang wurden für die Benutzerführung bei MCDET-Analysen gut dokumentierte Jupyter-Notebooks breitgestellt, die den Nutzer mit den Schritten einer MCDET-Analyse vertraut machen. Diese Notebooks beziehen sich insbesondere auf das Post-Processing und die möglichen Auswerteoptionen bei MCDET-Analysen. Außerdem erfolgte eine Beschreibung für die Benutzerführung bei Anwendung des CrewModuls von MCDET. Die Beschreibung bezieht sich auf das klassische CrewModul, das die Grundlage für entsprechende Weiterentwicklungen ist. Bei der Beschreibung wurde auch auf allgemeine Überlegungen zur Modellierung menschlicher Handlungen in einer dynamischen PSA eingegangen. Mit Beispielen wurde erläutert, wie ein menschlicher Handlungsablauf unter Berücksichtigung von Unsicherheiten modelliert wird. Es ist beabsichtigt, die Benutzerführung für MCDET und das CrewModul kontinuierlich entsprechend den durchgeführten Weiterentwicklungen anzupassen.

Literaturverzeichnis

/BER 16/ Berner, N., Herb, J.: Generic framework for the automated integration of impacts from hazards in PSA models, In L. Walls. M. Revie, and T. Bedford (Eds.), Risk, Reliability and Safety: Innovation Theory and Practice: Proceedings of the 26th European Safety and Reliabilty Conference, ESREL 2016, Glasgow, Scotland, pp. 2645-2649.

/BER 17/ Berner, N., Herb, J.: Advancement of the methodology for automated integration of external hazards into level 1 PSA modeling. Technical Report GRS-454, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Garching, March 2017.

/DIM 15a/ Di Maio, F. et al.: A computational framework for prime implicant identification in noncoherent dynamic systems. Risk Analysis: an official publication of the society of risk analysis 35, pp. 142-156.

/DIM 15b/ Di Maio, F. et al.: A visual interactive method for prime implicant identification. IEEE Transactions on Reliability 64(2), pp. 539-549.

/ERA 22/ Eraerds, T. et al.: Prime implicant identification in the dynamic process of a steam generator tube rupture scenario, Proceedings of the 32nd European, Safety and Reliability Conference (ESREL 2022), Dublin, Ireland, pp. 2475-2482.

/FDE 22/ Scheuer, J. et al.: Fortran Development Extensions (libfde), URL: https://gitlab.com/Zorkator/libfde, 2022, Rev. 2.8.1.

/HAR 20/ Harris, C. R.: Array programming with NumPy, Nature 585, 357–362, 2020.

/KLO 21/ Kloos, M., Berner, N.: SUSA, Software for Uncertainty and Sensitivity Analyses, Classical Methods, GRS-631, 2021.

/MCG 13/ McGrattan, K. et al.: Fire Dynamics Simulator, Users's Guide, NIST Special Publication 1019, National Institute of Standards and Technology (NIST), Gaithersburg, MD, November 2013.

/OGU 81/ Ogunbiyi, E., Henley, E.J.: Irredundant forms and prime implicants of a function with multistate variables. IEEE Transactions on Reliability R-30, pp 39-42

/PES 95/ Peschke, J.: Methoden zur Gewinnung von Verteilungen für Zuverlässigkeitskenngrößen aus Vorinformation und anlagenspezifischer Betriebserfahrung, GRS-A-2220, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Garching, Januar 1995.

/PES 14/ Peschke, J. et al.: Methodenentwicklung zur Analyse von Personalhandlungen im Rahmen probabilistischer Dynamikanalysen am Beispiel von Brandereignisabläufen mit Brandbekämpfung, GRS-331, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Garching, Juni 2014.

/PES 18/ Peschke, J. et al.: Methodische Weiterentwicklungen und Anwendungen zur probabilistischer Dynamikanalyse, GRS-520, Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH, Garching, September 2018.

/SWA 83/ Swain, A. D., Guttmaпп, H. E.: Handbook oп Human Reliability Analysis with Emphasis oп Nuclear Power Plant Applications, Final Report, NUREG/CR-1278, United States Nuclear Regulatory Commission (NRC), Washington, DC, USA, August 1983.

/SWA 87/ Swain, A. D.: Accident Sequence Evaluation Program-Human Reliability Analysis Procedure, U.S. Nuclear Regulatory Commission (NRC), NUREG/CR–4772, Washington, DC, 1987.

/VIR 20/ Virtanen, P. et al.: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, *Nature Methods*, 17(3), pp. 261-272, 2020.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ASPIC	Quantitative Assessment of Parameter Influence on Value Cluster
ATHLET	Analyse der Thermohydraulik von Lecks und Transienten
ATHLET-CD	ATHLET-Core Degradation
BL	Brandläufer
CI/CD	Continuous Integration/Continuous Deployment
CFD	Computational Fluid Dynamics
DET	Dynamic Event Tree
DLL	Dynamic Link Library (dynamische Bibliothek)
DSA	Deterministische Sicherheitsanalyse
eBEPU	Best Estimate Plus Uncertainty
FDE	Schnittstellen-Bibliothek, FDE, Fortran Development Extensions, https://gitlab.com/Zorkator/libfde
GCSM	ATHLET-Modul für Steuerung und Regelung
GRAMOVIS	Grafische Modellierung und Visualisierung
GUI	Bedienoberflächenelement
HCO	Heat Conduction Object (Wärmeleitungsobjekt)
HL	Handlungsliste
IdNr	Identifikationsnummer
IDPSA	Integrierte deterministisch-probabilistische Sicherheitsanalyse
IPC	Inter Process Communication
JSON	Dateiformat, JavaScript Object Notation, http://json.org/
KEY	Dateiformat, ATHLET-Schlüssel/Index-Tabelle für PD-Datei
LG	Löschgruppe
MCDET	Monte-Carlo Dynamic Event Tree
MCS	Minimal-Cut-Set
NKS	Notkühlsystem
PBS	Portable Batch System, https://www.openpbs.org/

PD	Dateiformat, ATHLET-Plot Daten (siehe: KEY)
PSA	Probabilistischen Sicherheitsanalyse
RESA	Reaktorschnellabschaltung
RF	Reaktorfahrer
SDE	Sekundärseitiges Druckentlasten
SL	Schichtleiter
TFO	Thermo-Fluid-Objekt
YAML	Dateiformat, Yet Another Markup Language oder Ain’t Markup Language (rekursives Akronym), https://yaml.org/

Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH

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

Boltzmannstraße 14
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 978-3-949088-77-3