Weiterentwicklung und Einsatz eines automatisierten Verifizierungsverfahrens für Analysesimulatoren

GRS - 679

Gesellschaft für Anlagen-
und Reaktorsicherheit
(GRS) gGmbH

Weiterentwicklung und Einsatz eines automatisierten Verifizierungsverfahrens für Analysesimulatoren

Abschlussbericht

Stefan Wenzel
Simone Palazzo
Joachim Herb
Josef Scheuer
Zhuoqi Du
Hristo Goumnerov

Mai 2022

Anmerkung:

Das diesem Bericht zugrunde liegende Forschungsvorhaben wurde mit Mitteln des Bundesministeriums für Umwelt, Naturschutz, nukleare Sicherheit und Verbraucherschutz (BMUV) unter dem Förderkennzeichen 4719R01375 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 - 679
ISBN 978-3-949088-70-4

Deskriptoren

ATHLET, Automatisierung, Continuous Integration, Controller, Gitlab, kontinuierliche Integration, Steuerung, Verifizierung

Inhaltsverzeichnis

1 Einleitung 1

1.1 Wissenschaftliche und technische Arbeitsziele 1

1.2 Strukturierung des Arbeitsprogramms 3

2 Standes von Wissenschaft und Technik 7

3 Erstellung eines generischen ATHLET-Controllers zur Simulationsdurchführung mit automatisierten Maßnahmen 11

3.1 Konzept eines generischen ATHLET-Steuerungsprogramms (ATHLET-Controller) 11

3.2 Beschreibung der Anwendung des ATHLET-Controllers 14

3.2.1 Systemvoraussetzungen 15

3.2.2 Spezifikationsdateien (Ablaufprotokolle) 15

3.2.3 ATHLET-Controller-Befehle in der Kommandozeile 23

4 Erstellung von CI-Konfigurationen zur Durchführung von Analysen unter Verwendung von Analysesimulatoren 25

4.1 Einführung in das Konzept der kontinuierlichen Integration 25

4.2 Umsetzung des CI-Konzeptes zur Anwendung im Kontext der Verifizierung von Analysesimulatoren 25

4.2.1 Konzeptbeschreibung und Vorstellung des Ablaufs 25

4.2.2 Auslösung der Verifikation 35

4.2.3 Erfahrung bei der Implementierung dynamisch generierter Pipelines auf Gitlab-CI 37

5 Protokollierung und Umsetzung der simulationsspezifischen Maßnahmen zur Durchführung der Rechnungen 39

5.1 Einführung in die Protokollierung von Steuerungsbefehlen bei Anwendung des ATHLET-Controllers 39

5.2 Beschreibung erstellter Protokolle für die Ereignisse aus der Simulationsmatrix 44

5.2.1 Ereignis #1 - Volllast 44

5.2.2 Ereignis #2 - Abfahren_UH_UK ... 45
5.2.3 Ereignis #3 - D2-12_DH_SpVentil ... 45
5.2.4 Ereignis #4 - D2-14_HD_Reduzier_AUF ... 46
5.2.5 Ereignis #5 - D2-16_HD_Reduzier_ZU ... 47
5.2.6 Ereignis #6 - D2-21_KMT ... 48
5.2.7 Ereignis #7 - D2-24_Deionat ... 49
5.2.8 Ereignis #8 - D2-01_FD_SiVentil ... 50
5.2.9 Ereignis #9 - D2-07_LAW-MAN ... 50
5.2.10 Ereignis #10 - D2-09_HspW_Pumpen ... 51
5.2.11 Ereignis #11/12 - D2-02-06_HWSenke ... 52
5.2.12 Ereignis #13 - D2-10_PUMA1v4 ... 53
5.2.13 Ereignis #14 - D2-28_Notstrom ... 54
5.2.14 Ereignis #15 - D3-31_DE_Heizrohr ... 55
5.2.15 Ereignis #16 - D3-23_KMV_01F ... 56
5.2.16 Ereignis #17 - D3-30_DH_SiVentil ... 57
5.2.17 Ereignis #18 - D3-05_FD_Leck ... 58
5.2.18 Ereignis #19 - D4a-04_ATWS ... 59

6 Definition der Bandbreite für die wesentlichen simulationsspezifischen Anlagenparameter ... 61

7 Aufbau eines Meldungsgenerators für die Post-Processing Phase ... 67

8 Exemplarische Durchführung der automatischen Verifizierungsprozedur und Auswertung der Ergebnisse der Simulationen ... 73

8.1 Allgemeine Anwendung des automatisierten Verifizierungsverfahrens ... 73
8.2 Anwendung an einem generischen Vorkonvoi-Datensatz und damit verbundene Anpassungen am Analysesimulator ... 75
8.2.1 Überarbeitung der implementierten Füllstandsmessung in Dampferzeuger und Druckhalter ... 75
8.2.2 Ausfall aller in Betrieb befindlichen Hauptspeisewasserpumpen mit Zuschaltung der Reservepumpe (D2-09) ... 77

8.2.3 Fehlöffnen der HD-Reduzierstation (D2-16) ... 78
8.2.4 Störung in der KMT-Regelung, die zu einem unkontrollierten Ausfahren von Steuerstäben führt (D2-21) ... 80
8.2.5 Ausfall der Hauptspeisewasserversorgung und mechanisches Verklemmen aller Steuerstäbe (ATWS) ... 81
8.2.6 Notstromfall gleich oder kürzer als 10 Stunden (D2-28) ... 82
8.2.7 Abfahren der Anlage in den Zustand „Unterkritisch Kalt“ ... 82
8.3 Übertragung auf einen generischen Konvoi-Datensatz ... 83
8.3.1 Voraussetzungen zur Übertragung der Verifikationsprozedur ... 83
8.3.2 Beschreibung erstellter Protokolle für die exemplarische Übertragung der Verifikationsprozedur auf einen Konvoi-Datensatz ... 85
8.3.3 Anpassungen und Optimierungen am Konvoi-Analysesimulator nach exemplarischer Übertragung der Verifikationsprozedur ... 88

9 Ausdehnung des automatisierten Verifizierungsverfahrens auf einen gekoppelten ATHLET/QUABOX-CUBBOX Vorkonvoi-Datensatz ... 91
9.1 Notwendige Anpassungen der Verifizierungsprozedur auf Gitlab-CI ... 92
9.2 Ereignisse zur Verifizierung der gekoppelten ATHLET/QUABOX-CUBBOX Datensatzes ... 93
9.2.1 Volllast mit BOC- und EOC-Kernbedingungen ... 93
9.2.2 Lastabwurf auf Eigenbedarf (EOC) (D2-07) ... 94
9.2.3 Fehlöffnen eines FD-Sicherheitsventils (EOC) (D2-01) ... 95
9.2.4 Auswurf des wirksamsten Steuerelements (BOC) (D3-16) ... 97

10 Zusammenfassung und Ausblick ... 103

Literaturverzeichnis ... 107

Abbildungsverzeichnis ... 111

Tabellenverzeichnis ... 113

1 Einleitung

1.1 Wissenschaftliche und technische Arbeitsziele

Die GRS entwickelt und wendet anlagenspezifische Analysesimulatoren an, um das komplexe thermohydraulische und leittechnische Anlagenverhalten für anomale Betriebszustände, Störfälle und auslegungsüberschreitende Störfälle analysieren, aktuelle Fragestellungen kurzfristig beantworten und wissenschaftlich-technische Fragen auf dem Gebiet des Reaktor- und Anlagenverhaltens klären zu können. Die Basis für diese Analysen stellen Eingabedatensätze dar, welche für eine konkrete kerntechnische Anlage und zur Anwendung mit dem jeweiligen Rechenprogramm der GRS (z. B. ATHLET) entwickelt werden. Um eine hohe Aussagesicherheit des simulierten Anlagenverhaltens zu erzielen, kommt dem Arbeitsschritt der Verifizierung dieser Datensätze eine hohe Bedeutung zu. Unter Verifizierung wird in diesem Zusammenhang eine umfassende systematische Vorgehensweise verstanden, welche die Prüfung der Korrektheit des Modells gegen eine definierte Bewertungsbasis beschreibt.

Im Rahmen des AP 1 des abgeschlossenen Vorhabens 4715R01345 wurde eine einheitliche Vorgehensweise für die Verifizierung von Analysesimulatoren entwickelt und manuell exemplarisch für eine Vorkonvoi-Anlage durchgeführt /PAL 18/. Durch eine sorgfältige Auswahl von Transienten bzw. Ereignissen aus der Sicherheitsebene 2 bis 4a /SIA 15/ wurde dabei ein Mindestsatz durchzuführender Ereignisanalysen definiert, mit dem Ziel die korrekte Nachbildung der betrieblichen Systeme/Komponenten, Begrenzungssysteme und der Sicherheitssysteme in ihren unterschiedlichen Zuständen zu zeigen.

Das im Rahmen des o. g. Vorhabens entwickelte Verifizierungsverfahren ermöglicht die Einhaltung eines hohen Qualitätsstandards der Datensätze und, falls erforderlich, eine Verbesserung der im Datensatz des Analysesimulators implementierten betrieblichen und sicherheitstechnischen Systeme bzw. Komponenten. Die wesentlichen Funktionen der Systeme, die für die Beherrschung von Transienten oder Störfällen von Bedeutung sind, werden vom entwickelten Verfahren abdeckend geprüft.

Bei der Entwicklung von Analysesimulatoren liegt eine nicht weiter quantifizierte Wahrscheinlichkeit zur Implementierung von Fehlern in Folge der Erstellung selbst oder im Zuge notwendiger Anpassungen der Datensätze für die Implementierung neuer

Funktionen bzw. für die Verfeinerung oder Neumodellierung einzelner Komponenten vor. Die Folgen solcher Fehler sind mögliche Abweichungen des Anlagenverhaltens bei der Berechnung von spezifischen Ereignissen, die sich ggf. auch nur bei wenigen Ereignissen auswirken. Der Anspruch an ein geeignetes Verifizierungsverfahren muss sein, das Auffinden derartiger Fehler zuverlässig und innerhalb eines angemessenen Zeitrahmens zu ermöglichen. Ein derartiges Verifizierungsverfahren sollte weiter einen modularen Charakter besitzen, um durch eine Erweiterung von verwendeten Ereignislisten und eine Erhöhung der Anzahl bzw. Frequenz von Simulationen, die Effektivität des Verfahrens bei der Entdeckung von Fehlern in Datensätzen steigern zu können. Dabei stellt der Effektivitätsgewinn aus einer Erweiterung der Ereignisliste ein Optimierungsproblem gegenüber des erhöhten Rechen- und Analyseaufwandes dar, welches in die Ausarbeitung des Verfahrens einzubeziehen ist. Eine Automatisierung des im Vorhaben 4715R01345 /PAL 18/ vorgeschlagenen Verifizierungsverfahrens kann eine sachgerechte Lösung der oben beschriebenen Problematiken darstellen.

Ziel der hier vorgestellten Arbeiten ist es, ein automatisiertes Verifizierungsverfahren zu entwickeln, durch dessen Anwendung auf einen Analysesimulator-Datensatz mögliche Fehler im Anlagenmodell effektiv und effizient gefunden und behoben werden können und somit die Einhaltung eines hohen Qualitätsstandards der in der GRS entwickelten Simulatoren kerntechnischer Anlagen sicherzustellen. Weiter ist im Zuge der Arbeiten etwaiger Optimierungsbedarf an den einbezogenen Analysesimulator-Datensätzen zu identifizieren, um eine umfassend verifizierte Datenbasis für Störfallanalysen zu erreichen. Der Einsatz des automatisierten Verifizierungsverfahrens wird es dabei auch ermöglichen, Änderungen im Anlagenmodell, welche beispielsweise infolge von Anlagenänderungen in der Leittechnik bzw. infolge einer angepassten Modellierungstiefe erfolgen, zeitnah und im Hinblick auf die Auswirkungen auf den gesamten Satz der zur Verfügung stehenden Bewertungsbasis umfassend zu bewerten. Durch die Implementierung und Verwendung eines Meldungsgenerators wird sichergestellt, dass künftig beim Eintreten von Fehlern in einzelnen Teilen des Analysesimulator-Datensatzes eine schnelle Anpassung durchgeführt werden kann. Bei der Umsetzung kommen aus der Softwareentwicklung abgeleitete Methoden zur kontinuierlichen Integration (englisch „Continuous Integration“; CI) zum Einsatz, welche in der GRS etablierte Prozesse zur Qualitätssicherung bei der (Weiter-)Entwicklung von Software-Tools darstellen. Das erarbeitete Wissen dient ferner als eine wesentliche Grundlage, um außerhalb der GRS erstellte Analysen (z. B. im Rahmen von Genehmigungs- und Aufsichtsverfahren) gezielter bewerten zu können.

1.2 Strukturierung des Arbeitsprogramms

Das abgeschlossene Vorhaben stellt eine Fortführung des vom BMUV geförderten Vorhabens 4715R01345 /PAL 18/ dar. Die in diesem Vorhaben durchgeführten Arbeiten gliedern sich inhaltlich entsprechend dem Arbeitsprogramm in folgende Arbeitspakete:

Arbeitspaket 1: Aufarbeitung des für das Vorhaben relevanten Standes von Wissenschaft und Technik

Der für die Bearbeitung des Vorhabens relevante Stand von Wissenschaft und Technik wird systematisch aufbereitet und um die im Projekt gewonnen Ergebnisse und Erfahrungen ergänzt. Die Aufbereitung des Standes von Wissenschaft und Technik bezieht sich u. a. auf:

- Methoden, Daten, Vorgehensweisen und Ergebnisse aus den bisherigen Forschungsarbeiten der GRS;
- Auswertung wichtiger Untersuchungen und Ergebnisse anderer Stellen (abgeschlossene sowie laufende Arbeiten, Literaturrecherche);
- Sichten der aktuellen relevanten Informationssysteme;
- Aufarbeitung der Ergebnisse aktueller Beratungen in einschlägigen nationalen und internationalen Gremien;
- vorhandene Bewertungsmaßstäbe, die dem Vorhaben zugrunde liegen;
- Einarbeitung in bewährte Vorgehensweisen („Best Practices“) sowie konkrete Anwendung von Plattformen zur kontinuierlichen Integration, wie Jenkins /PAT 17/ und Gitlab-CI /ARE 19/.

Arbeitspaket 2: Erstellung eines generischen ATHLET-Controllers zur Simulationsdurchführung mit automatisierten Maßnahmen inklusive Handmaßnahmen

Zur automatischen Durchführung von ATHLET-Simulationen wird die dazu vorgesehene Python-Schnittstelle von ATHLET benutzt. Diese erlaubt es, ein externes Steuerungsprogramm für ATHLET (ATHLET-Controller) zu erstellen und damit fallspezifische Abläufe und Tests festzulegen, um Anlagenänderungen, wie sie sich beispielsweise aus festgelegten Prozeduren der Betriebsdokumentation ergeben, zu integrieren. Dieses Steuerprogramm wird möglichst generisch aufgebaut und die Ablaufsteuerung in Form

einer einfachen Spezifikationsdatei verarbeitet, um das Steuerprogramm künftig für alle Arten von Test-Ablaufsteuerungen einsetzen zu können. Hierdurch kann dann die Simulationsmatrix einfach und in vergleichsweise kurzer Zeit um neue Testfälle erweitert werden.

Arbeitspaket 3: Erstellung von CI-Konfigurationen zur Durchführung von Analysen unter Verwendung von Analysesimulatoren

Eine GRS-Plattform zur kontinuierlichen Integration wurde im Rahmen des Vorhabens RS1538 /SCH 18/ auf Basis des Softwarepakets Jenkins /PAT 17/ aufgebaut. In der Vergangenheit wurde in der GRS ein Jenkins-Server verwendet, um die verschiedenen Versionen des GRS-Systemcodes ATHLET automatisch zu kompilieren und zu testen. Ab 2020 fand eine Umstellung der GRS internen Software-Verwaltung auf Gitlab /CHO 20/ statt. Gitlab ist eine Anwendung zur Versionsverwaltung für Softwareprojekte und stellt zukünftig die zentrale Plattform für sämtliche programmtechnische Entwicklungen der GRS dar. Es verfügt auch über einen Server zur kontinuierlichen Integration (CI), der auf einer Hardware installiert wurde, deren Leistungsfähigkeit höher ist als die des Jenkins-Systems und die zukünftig weiter gewartet und erweitert wird. Eine Migration des anfänglich auf Basis von Jenkins entwickelten Verfahrens zur kontinuierlichen Integration auf Gitlab-CI wurde im Zuge des Vorhabens umgesetzt. Diese führte zu einer

- Sicherung der Zukunftsfähigkeit durch fortwährende zentrale Wartung und Pflege der Server-Infrastruktur,
- Erhöhung der Rechenkapazität und damit Verkürzung des Verifizierungsdurchlauf,
- Harmonisierung der CI-Abläufe mit sonstigen programmtechnischen Entwicklungen in der GRS und
- perspektivisch zu einer Verwendung der offiziellen Installationsdateien von ATHLET auf Gitlab und damit zur Sicherstellung der Konsistenz (derzeit eigener Build-Prozess auf CI-Server).

Arbeitspaket 4: Protokollierung und Umsetzung der simulationsspezifischen Maßnahmen zur Durchführung der Rechnungen

Für die Verifizierung des vorrangig einbezogenen Analysesimulator-Datensatzes werden nach /PAL 18/ insgesamt 19 Simulationen für die wesentlichen Ereignisse der Sicherheitseben (SE) 2 bis 4a durchgeführt. Für die Durchführung der ausgewählten

Rechnungen wurde im vorangegangenen Vorhaben 4715R01345 /PAL 18/ eine Funktion des Simulator-Tools ATLAS benutzt, welche eine Protokollierung der für die Initiierung der Transienten oder Störfällen notwendigen Maßnahmen ermöglichte, dabei aber weiterhin manuelle Eingriffe erforderte und in Umfang und Funktionalität stark limitiert war. Eine derartige Protokollierung der transientenspezifischen Maßnahmen wird im hier vorgestellten Vorhaben durch Erstellung einer Spezifikationsdatei umgesetzt, welche von dem neu entwickelten Steuerprogramm (ATHLET-Controller) eingelesen werden kann. Hierdurch ist eine automatische Durchführung und detaillierte Steuerung von ATHLET-Simulationen möglich, welche auch in der Lage ist, erweiterte Steuerungsszenarien, wie z. B. regelmäßig wiederkehrende oder zustandsabhängige Handmaßnahmen, abzuarbeiten.

Arbeitspaket 5: Definition der Bandbreite für die wesentlichen simulationsspezifischen Anlagenparameter

Damit die Verifizierung der Datensätze der Analysesimulatoren automatisch auf die GRS-Plattform zur kontinuierlichen Integration erfolgen kann, ist eine Überprüfung der erzielten Ergebnisse anhand des Vergleichs mit vorgegebenen Referenzkurven erforderlich. Dafür wird eine sorgfältige Auswahl relevanter simulationsspezifischer Anlagenparameter für jeden Fall der in /PAL 18/ entwickelten Simulationsmatrix benötigt. Um die Arbeit für die Auswahl der simulationsspezifischen Anlagenparameter zu unterstutzen, werden die Ergebnisse aus dem vorangegangenen Vorhaben 4715R01345 /PAL 18/ herangezogen. Die Referenzkurven werden danach als Bewertungsbasis für die automatische Verifizierung gespeichert und für die Bestimmung von Grenzkurven verwendet, welche als Akzeptanzkorridore der Verifizierung dienen.

Arbeitspaket 6: Aufbau eines Meldungsgenerators für die Post-Processing Phase

Neben den trivialen Ausführungsfehlern der Simulation wie Programmabsturz oder vorzeitige Beendigung der Rechnung müssen weitere Prüfungen implementiert werden, welche z. B. Rechenergebnisse oder Daten des Simulationszustands für die Bewertung zugrunde legen. Hierfür sind folgende Prüfungen geplant:

- **Plausibilitätsprüfungen**: Der Zustand der Simulation wird während der Berechnung bewertet. Wird eine festgelegte Bedingung nicht erfüllt, wird die Simulation vorzeitig abgebrochen, was als fehlerhaftes Ergebnis der Verifikation erkannt wird.

- **Integrale Ergebnisprüfungen:** Die Simulationsergebnisse werden anhand des Verlaufs ausgewählter Ergebnisgrößen automatisch bewertet. Hierzu werden von Datensatzverantwortlichen Grenzkurven spezifiziert, welche die zu akzeptierenden Abweichung im Ergebnisverlauf festlegen. Werden diese Grenzkurven verletzt bzw. überschritten, gilt der Test als nicht bestanden.

Die dabei entwickelten Skripte zur Durchführung der Prüfungen in der Post-Processing-Phase basieren auf der Programmiersprache Python.

Arbeitspaket 7: Exemplarische Durchführung der automatischen Verifizierungsprozedur und Auswertung der Ergebnisse der Simulationen

Die entwickelte Vorgehensweise zur automatischen Verifizierung von Datensätzen wird exemplarisch angewendet, um die Qualität, der in Rahmen der AP 2 bis AP 6 entwickelten Prozeduren prüfen zu können. Neben einer vollständigen Umsetzung der automatischen Verifizierung anhand des Datensatzes eines Vorkonvoi-Analysesimulators, welcher im Rahmen des vorangegangenen Vorhabens 4715R01345 /PAL 18/ weitgehend verifiziert wurde, wird das Verfahren zusätzlich exemplarisch auf einen Konvoi-Datensatz angewendet, um eine Übertragbarkeit sicherzustellen.

Es werden dabei beispielhaft Modifikationen von betrieblichen Systemen in den Eingabedatensätzen vorgenommen, welche u. a. im Rahmen der Arbeit für das Vorhaben 4717R01334 /PAL 17/ als Verbesserungsmaßnahmen identifiziert wurden.

Nach der Anpassung des Datensatzes wird die Durchführung der Verifizierungsprozedur innerhalb der CI-Infrastruktur der GRS automatisch gestartet. Es wird für alle Simulationen der Simulationsmatrix geprüft, ob die Schritte zur Protokollierung und Umsetzung der simulationsspezifischen Maßnahmen und zur Definition der Bandbreite für die wesentlichen simulationsspezifischen Anlagenparameter automatisch durchgeführt werden. Gleichzeitig wird geprüft, ob die implementierten Checks beim Aufbau eines Meldungsgenerators ausreichend sind, um einen Rechenlauf weitestgehend automatisch auswerten zu können. Werden einzelne Prüfungen nicht bestanden ist eine sorgfältige Analyse der Simulationsergebnisse notwendig, um sicherzustellen, dass der von ATHLET berechnete Verlauf der Anlagenparameter physikalisch und technisch plausibel ist.

2 Standes von Wissenschaft und Technik

Zur Qualitätssicherung von Sicherheitsanalysen empfiehlt die IAEA ein standardisiertes Verfahren, das zum einen die Güte der Simulationsergebnisse sicherstellt und zum anderen die Anforderungen an die Anlage berücksichtigt /IAE 09/. Nach Verifikation der verwendeten Anlagen- bzw. Eingabedaten bedarf es laut IAEA einer Über- sowie Gegenprüfung des Datensatzes, um Fehler aus der Entwicklungsphase zu korrigieren. Zudem sind die angewandten Anlagenmodelle zu validieren /IAE 09/. Die praktische Vorgehensweise für die Verifizierung eines Datensatzes wird von der IAEA offengelassen.

In /PET 08/ sowie in /DAU 04/ wird eine Übersicht über Validierungsverfahren von in der Sicherheitsanalyse angewendeten Rechencodes sowie anlagenspezifischer Datensätze vorgestellt. Die Autoren stellen fest, dass bezüglich der Validierung der Datensätze eine Kontrolle über die implementierte Nodalisierung notwendig ist, um die Wirkung vieler verschiedener Näherungsquellen wie den folgenden zu berücksichtigen:

- Die Daten der Referenzanlage, die dem Benutzer zur Verfügung stehen, sind in der Regel nicht ausführlich, um eine perfekte „Schematisierung“ der Referenzanlage zu reproduzieren;
- Aus den verfügbaren Daten leitet der Benutzer eine approximierte Nodalisierung der Anlage ab, wodurch der Detailgrad reduziert wird.

Demzufolge wird in den o. g. Veröffentlichungen eine Prozedur für die Validierung der Datensätze vorgeschlagen, mit dem Ziel, die Qualitätssicherung des Datensatzes zu gewährleisten. Das Verfahren erfasst unterschiedliche Arbeitsschritte, die sich auf die korrekte Nachbildung eines breiten Spektrums von thermohydraulischen Phänomenen fokussieren. Am Ende des Verfahrens wird ein qualifizierter anlagenspezifischer Datensatz erreicht, der für die Analyse von spezifischen Transienten bzw. Störfällen geeignet ist. Damit der Datensatz ein breiteres Spektrum von Transienten bzw. Störfällen abdecken kann, wird der in /PET 08/ definierte „on transient level qualification“-Arbeitsschritt im Verfahren mehrmals durchgeführt. Das führt zu einer Steigerung des Analyseaufwands aufgrund der kontinuierlichen Prüfung der Ergebnisqualität gegenüber einer Bewertungsbasis sowie zu einer Wiederholung der im Verfahren vorgesehenen Arbeitsschritte aufgrund des darauffolgenden Verbesserungsbedarfs im Datensatz (z. B. Anpassung bzw. Verfeinerung der Nodalisierung einzelner Thermofluidobjekte).

Die in /PET 08/ sowie in /DAU 04/ vorgeschlagene Vorgehensweise befasst sich mit der *Validierung* der Datensätze. Die Validierung eines Eingabedatensatzes soll sicherstellen, dass das entwickelte Modell alle Funktionen der einzelnen modellierten Systeme adäquat darstellen kann. Die Validierung erfolgt durch einen Vergleich zwischen den Anlagedaten und Analyseergebnissen der mit dem Analysesimulator nachberechneten Ereignisse nach dem Ablauf einer Transiente bzw. eines Störfalls. Im Gegensatz dazu stellt die *Verifizierung* eines Datensatzes sicher, dass die thermohydraulische Modellierung sowie die Nachbildungen der leittechnischen Funktionen für jedes einzelne System im Eingabedatensatz den Anlagenspezifikationen bzw. der Anlagendokumentation entsprechen. Eine abdeckende Verifizierung der Eingabedatensätze ist für die Analysesimulatoren erforderlich, um die korrekte Funktion aller Systeme der Analysesimulatoren sicherzustellen.

Im Rahmen des abgeschlossenen Eigenforschungsvorhabens 4715R01345 /PAL 18/ wurde von der GRS eine einheitliche Vorgehensweise entwickelt, mit der die Analysesimulatoren weitestgehend abdeckend verifiziert werden können. Hierzu wurde ein Mindestsatz an Ereignisanalysen definiert, mit dem es möglich ist, die korrekte Nachbildung der betrieblichen Systeme / Komponenten, Begrenzungs- und Sicherheitssysteme in ihren unterschiedlichen Zuständen zu zeigen. Eine Liste von Ereignissen wurde erstellt, auf deren Basis das Verifizierungsverfahren durchgeführt wurde. Aufgrund des hohen Komplexitätsgrads des Eingabedatensatzes wurde das Verifizierungsverfahren im Wesentlichen in drei Schritte eingeteilt, mit dem Ziel, die unterschiedlichen Teile des Datensatzes bzw. die einzelnen Systeme des Analysesimulators schrittweise mit einem höheren Untersuchungsgrad zu analysieren und zu verifizieren. Als Bewertungsbasis für die Überprüfung der Funktionalität der im Analysesimulator-Eingabedatensatz implementierten Betriebs- und Sicherheitssysteme wurden Informationen aus folgenden Quellen herangezogen:

- Vorliegende anlagenspezifische Dokumentationen (Betriebs- und Sicherheitssysteme),
- Ergebnisse durchgeführter Rechnungen,
- Informationen aus verfügbaren Schulungsunterlagen bzw. vom vergangenen Störfallanalysenhandbuch-Projekt (Vorhaben 3612R01335).

Insgesamt 19 Simulationen wurden für das Verifizierungsverfahren für die wesentlichen Ereignisse der Sicherheitsebenen (SE) 2 bis 4a durchgeführt. Nach der Durchführung

jeder Rechnung erfolgte die manuelle Gegenüberstellung der Ergebnisse mit den Daten aus der Bewertungsbasis. Waren die wesentlichen Anlageparameter bzw. deren zeitliche Verläufe in der Simulation qualitativ vergleichbar mit denen aus der Bewertungsbasis, war keine weitere Anpassung spezifischer Funktionen im existierenden Modell des Analysesimulators notwendig. Bei der Feststellung von Abweichungen zwischen den berechneten und den aus der Bewertungsbasis herausgezogenen Ergebnisse erfolgt eine Anpassung des Modells und die Behebung der Abweichungen.

Bisherige Praxis der GRS bei der Validierung von Analysesimulatoren war im Wesentlichen das Nachrechnen von Experimenten und realer in Anlagen abgelaufener Transienten, für welche Messergebnisse zur Verfügung standen. Die Anzahl der nachgerechneten Ereignisse erscheint jedoch nicht ausreichend für eine systematische Funktionsprüfung aller betrieblichen und der sicherheitstechnisch relevanten Systeme in den Analysesimulatoren. Vielmehr ist ein Verfahren zu erarbeiten, welches effizient (bei einer großen Anzahl zu berücksichtigender Transienten und Störfälle) automatisiert Abweichungen vom erwarteten oder bekannten Anlagenverhalten detektiert, um zeitnah und zielgerichtet Gegenmaßnahmen in Form verfeinerter Modellierung zu ermöglichen.

Eine Methode, welche die oben aufgeführten Anforderungen an Automatisierungsgrad, Reproduzierbarkeit und Prüfungsumfang erfüllt, ist die kontinuierliche Integration, welche in der Softwareentwicklung eingesetzt wird. Methoden zur kontinuierlichen Integration sind mittlerweile Stand der Technik zur Qualitätssicherung in der Softwareentwicklung. Ziel ist es dabei, die Entwickler durch das automatische Kompilieren der in der Entwicklung befindlichen Software und die automatische Durchführung von Tests mit entsprechenden, automatisch generierten Berichten zu unterstützen. Diese Tätigkeiten zusammen werden als kontinuierliche Integration (englisch „Continuous Integration“; CI) bezeichnet.

Im Rahmen des Vorhabens RS1538 /SCH 18/ hat die GRS eine Plattform zur kontinuierlichen Integration auf Basis der Software Jenkins /SMA 11/ für die Entwicklung und Validierung von Simulationsprogrammen aufgesetzt. Die Anforderungen der Architektur und die Anwendung dieser Plattform für die Bestandteile des GRS Codesystems AC² sind in /HER 18/ beschrieben. Ab 2020 fand eine Umstellung der GRS internen Software-Verwaltung auf Gitlab /CHO 20/ statt. Gitlab ist eine Anwendung zur Versionsverwaltung für Softwareprojekte und stellt zukünftig die zentrale Plattform für sämtliche programmtechnische Entwicklungen der GRS dar. Es verfügt auch über einen Server zur kontinuierlichen Integration (CI), der auf einer Hardware installiert wurde, deren

Leistungsfähigkeit höher ist als die des Jenkins-Systems und für die geplant ist, sie in Zukunft weiter zu warten und auszubauen.

Die Anwendung eines CI-Verfahrens erlaubt die automatisierte Durchführung von Tests zur Qualitätssicherung. Die Aufgabe solcher Tests ist es, zunächst nachzuweisen, dass sich ein neuer Code oder Code-Überarbeitungen wie spezifiziert verhalten. Besonders wichtig ist aber auch sicherzustellen, dass die bestehende Qualität auch bei Änderungen am Code erhalten bleibt. Nur durch eine ausreichende Abdeckung eines bestehenden Programms mit Tests ist es möglich, Änderungen und Weiterentwicklungen durchzuführen und dabei das bestehende Qualitätsniveau aufrechtzuerhalten.

Für alle Tests ist es dabei notwendig, Soll- oder Grenzwerte vorzugeben, mit denen die Testergebnisse verglichen werden können. In der Code-Entwicklung wird dabei zwischen verschieden Testarten unterschieden /SCH 18/. Bei der Verifizierung von Analysesimulatordatensätzen sind davon insbesondere Regressions-, Einzeleffekt und Integraltests von Bedeutung. Hierfür müssen die Ergebnisdateien nach Ausführen der Simulation analysiert werden. Dazu werden für zeitabhängige Ergebnisgrößen der Simulation Grenzwertkurven durch den Entwickler spezifiziert. Die Grenzwertkurve besteht aus Punkten, zwischen denen linear interpoliert wird. Die Basis zur Erstellung der Grenzkurven kann hierbei vielfältig sein und beispielsweise auf reale Vergleichsdaten (Experimente, Anlagendaten), Erfahrung der Entwickler oder bekannten Simulationsergebnisse zurückgehen.

Für den hier vorgeschlagenen Anwendungsfall der automatisierten Verifizierung von Analysesimulatoren sind eine Vielzahl von Grenzkurven für simulationsspezifische Anlagenparameterverläufe abzuleiten. Die Darstellung der dabei angewendeten Vorgehensweise erfolgt in Kapitel 0.

3 Erstellung eines generischen ATHLET-Controllers zur Simulationsdurchführung mit automatisierten Maßnahmen

3.1 Konzept eines generischen ATHLET-Steuerungsprogramms (ATHLET-Controller)

Eine Automatisierung der Durchführung von ATHLET-Simulationen unter Anwendung detaillierter anlagenspezifischer Analysesimulatoren stellt hohe Anforderungen an den Funktionsumfang und die Flexibilität eines passenden Steuerungsprogramms. ATHLET verfügt über eine Schnittstelle, um mit Hilfe der Programmiersprache Python in den Simulationsablauf eingreifen zu können. Diese Schnittstelle erlaubt es, ein Steuerungsprogramm für ATHLET (ATHLET-Controller) mit den notwendigen Eigenschaften zu erstellen und dieses zu nutzen, um fallspezifische Abläufe und Tests festzulegen. Mit Anwendung dieses ATHLET-Controllers sollen so Zustandsänderungen in der Anlage innerhalb des Simulationsablaufs umgesetzt werden, wie sie sich beispielsweise aus festgelegten Prozeduren der Betriebsdokumentation ergeben. Das Steuerprogramm soll dabei möglichst generisch aufgebaut sein und die Ablaufsteuerung in Form einer einfachen Spezifikationsdatei verarbeiten, um es künftig für alle Arten von Testablauf- und Simulationssteuerungen einsetzen zu können. Dies ermöglicht es zum einen eine datensatzspezifische Simulationsmatrix mit zu verifizierenden Ereignissen bei Bedarf einfach und schnell zu erweitern und zum anderen ggf. Datensätze zusätzlicher Analysesimulatoren mit überschaubarem Aufwand in das automatisierte Verifikationsverfahren zu integrieren.

Es wäre unverhältnismäßig aufwendig, eine direkte Steuerung eines Rechenlaufs mit der Programmiersprache Fortran, auf welcher ATHLET basiert, zu implementieren und darüber während einer Simulation in die Datenstruktur einzugreifen. Dies führt allerdings zu einem erheblichen Nachteil, da grundlegende Aufgaben wie das Überwachen (Auswerten von Simulationsparametern) häufig ausgeführt werden müssen, die sonst nicht verfügbar oder nur zu spärlich geschrieben sind. Ergebnisse der ATHLET-Simulation werden für eine Steuerung standardmäßig nicht ausreichend oft nach außen gegeben bzw. herausgeschrieben, weshalb Steuerungsbefehle von außen durch einen Controller eine inhärente Unsicherheit in der Ausführungszeit aufweisen würden. Für eine gezielte und eindeutige Steuerung muss deshalb auf jeden einzelnen ATHLET-Zeitschritt zugegriffen werden können, um die Daten „online“ auswerten und ändern zu können. Ziel der Entwicklungsarbeiten war es demnach, ein Python-basiertes Programm bereitzustellen,

welches das Starten und Steuern von Simulationsläufen in ATHLET reproduzierbar und eindeutig ermöglicht. Seine Funktionalität basiert stark auf der generischen Schnittstellenbibliothek (Fortran Development Extensions; libfde) /ZOR 21/, erweitert sie jedoch um Funktionen, die auf eine einfachere Interaktion mit ATHLET zugeschnitten sind.

Die Durchführung einer Simulation mit dem ATHLET-Controller erfolgt in folgenden Schritten:

- Ein ATHLET-Binary (.dll) wird eingeladen.
- Der Eingabedatensatz wird geladen und von ATHLET auf Konsistenz überprüft. Nach erfolgreichem Einlesen des Datensatzes wird ATHLET pausiert und wartet auf weitere Anweisungen des Steuerprogramms (ATHLET-Controller).
- Der Controller lädt die Datei mit den Steuerungsmaßnahmen (Ablaufprotokoll) und versucht diese auf den bereits eingeladenen ATHLET-Datensatz anzuwenden. Hierbei erfolgt eine Prüfung der Konsistenz auf die in den Steuerungsmaßnahmen referenzierten ATHLET-Daten, wie z. B. Signale, TFOs und HCOs.
- Die in den Steuermaßnahmen definierten Schritte zur Überwachung des Simulationszustandes werden aktiviert, sodass diese nach jedem ATHLET-Zeitschritt ausgeführt werden und ggf. die festgelegten Aktionen auslösen können.
- Der Controller gibt ATHLET die Anweisung zur weiteren Durchführung der Simulation, welche dann gemäß den definierten Maßnahmen überwacht abläuft.

Abb. 3.1 zeigt schematisch die Komponenten und das Zusammenwirken der Programmteile, welche für die Durchführung einer Simulation mit dem ATHLET-Controller notwendig sind.

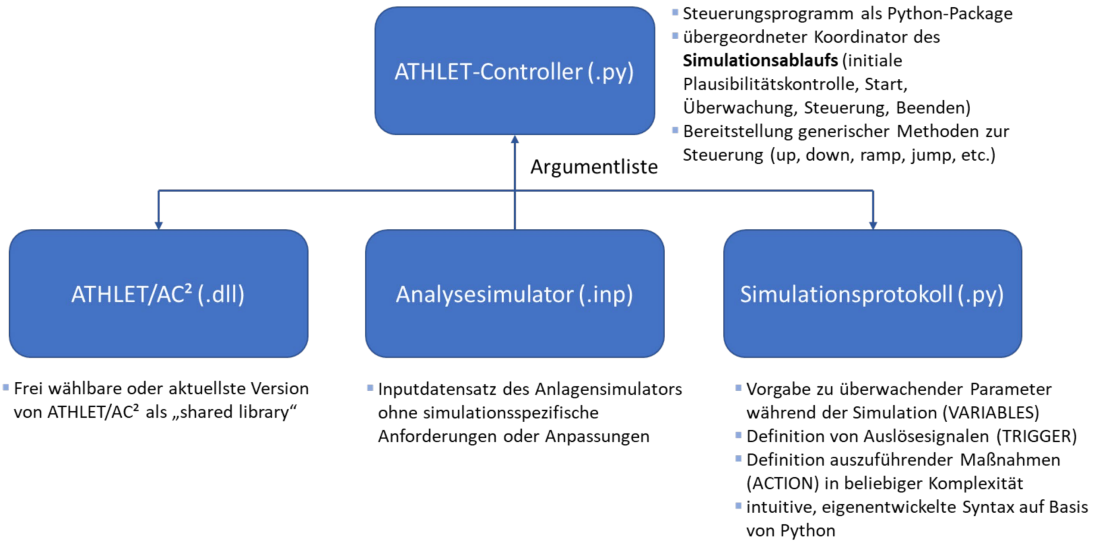

Abb. 3.1 Komponenten und Steuerungsschema des ATHLET-Controllers

Der ATHLET-Controller bietet drei Hauptklassen, um die Kommunikation zwischen den Python-basierten Spezifikationsdatei und ATHELT zu ermöglichen. Diese drei Klassen sind:

- Variablen: Abruf von und Zugriff auf Signale aus ATHLET
- Trigger: Erstellung von Bedingungen durch Anwendung logischer Operationen auf die definierten Variablen
- Action: Ausführung von Maßnahmen durch Zugriff und Änderung der zugehörigen Signalwerte in ATHLET

Als Format und Syntax der Spezifikationsdatei wurde eine Untermenge der Programmiersprache Python festgelegt. Dadurch kann ein Pythoninterpreter zum Einlesen verwendet werden, was die Wartung und mögliche Weiterentwicklungen sehr begünstigt. Außerdem bietet diese Lösung eine solide Basis für mathematische und logische Ausdrücke und ermöglicht große Flexibilität bei der Definition von Eingabekonstrukten und -überprüfungen.

Dem Anwender werden bei der Durchführung von ATHLET-Simulationen mit dem ATHLET-Controller Programmabbrüche in Form von Errorcodes signalisiert. Es ist nicht zwingend erforderlich, beim Starten des Controllers eine Spezifikationsdatei (Ablaufprotokoll) anzugeben, wodurch auch über den Controller ein zur Stand-Alone-Version von

ATHLET kompatibler Simulationsablauf durchgeführt werden kann. Dies vereinfacht die Reproduktion und Bewertung von Analyseergebnissen.

Um den ATHLET-Controller breiter anwendbar zu machen und den Bedarf an Rechenressourcen für das Verifizierungsverfahren zu minimieren, sind Simulationen mit dem ATHLET-Controller grundsätzlich restartfähig, es besteht also die Möglichkeit Simulationen auf zuvor erzeugte Ergebnisse aufzusetzen. Um diese Funktionalität sicherzustellen, waren auch Anpassungen in ATHLET selbst notwendig. ATHLET verhinderte bisher durch einen Abgleich der verwendeten Signaltypen in Eingabedatensatz und Restart-File, dass externe Signaländerungen, wie sie vom ATHLET-Controller vorgenommen werden, akzeptiert wurden. Nach den durchgeführten Änderungen akzeptiert ATHLET neben unveränderten Signaltypen in Abgleich mit dem Eingabedatensatz auch den vom ATHLET-Controller verwendeten Signaltyp „External“. Damit ist eine Restartfähigkeit von Ergebnissen, welche mit dem ATHLET-Controller erzeugt wurden, gegeben. Der ATHLET-Controller besitzt einen hohen Funktionsumfang im Umgang mit Restarts. Restart-Punkte können durch Funktionsdefinitionen in den Ablaufprotokollen und damit abhängig vom Anlagenzustand geschrieben und eingelesen werden. Dies hilft bei unerwarteten Systemverhalten, die Problemanalyse zu beschleunigen, da so abhängig vom betreffenden Ereignis lange Anlaufrechenzeiten deutlich verkürzt werden können.

Der ATHLET-Controller ist in der Lage, sowohl numerische wie auch boolesche Signalwerte in den Ablaufprotokollen zu verarbeiten, sowie auf im Datensatz spezifizierte Tabellenwerte zuzugreifen und diese bei Bedarf anzupassen. Auch hierzu waren Eingriffe in den ATHLET-Code erforderlich. Dieser Funktionsumfang des Controllers erlaubt es beispielsweise, Pumpenkurven oder Ventilöffnungscharakteristiken in bestimmten Störfallabläufen anzupassen und somit u. a. den Einfluss von Komponentenschädigungen zu untersuchen.

3.2 Beschreibung der Anwendung des ATHLET-Controllers

Im Folgenden wird ein Überblick über die verfügbaren Funktionen des ATHLET-Controllers geben und grundlegendes zu dessen Verwendung beschrieben. Anhand konkreter Beispiele wird gezeigt, wie eine Simulation in ATHLET über den Controller gestartet und das Systemverhalten gesteuert und überwacht werden kann.

3.2.1 Systemvoraussetzungen

Für die Verwendung des ATHLET-Controllers ist eine Python-Installation obligatorisch. Dabei kann beispielsweise auf die Distribution *Miniconda* (https://docs.conda.io/en/latest/miniconda.html) zurückgegriffen werden. Es wird empfohlen, eine eigenständige Python-Umgebung (Conda Enviroment) ausschließlich für die Verwendung des Controllers zu erstellen. Fehlende Pakete (z. B. docopt) können dann über den Befehl

```
conda install PACKAGENAME
```

einfach nachinstalliert werden. Als Folge der in Kapitel 3.1 beschriebenen notwendigen Anpassungen am ATHLET-Code ist der Controller erst ab ATHLET Version 3.2 oder höher inkl. der zugehörigen Shared Libraries (.dll) kompatibel.

3.2.2 Spezifikationsdateien (Ablaufprotokolle)

Die auszuführenden Steuerungsvorgänge werden in externen Python-Skripten (.py) definiert, welche als Spezifikationsdateien oder Ablaufprotokolle (protocols) bezeichnet werden. Sie werden während der Simulationslaufzeit vom ATHLET-Controller gelesen und ausgeführt. Der Benutzer gibt die Protokolldatei über die Befehlszeilenoption

```
--protocol="~\PATH\TO\PROTOCOLNAME.py"
```

bei Aufruf des Startbefehls des ATHLET-Controllers an. Die Protokolle bestehen aus drei bis vier Abschnitten:

- Variablendefinition
- Trigger-Definition
- Action-Definition
- Funktionsdefinition (optional)

Ein Beispiel für Aufbau und Inhalt einer solchen Spezifikationsdatei ist in List. 3.2 gegeben.

3.2.2.1 Variablendefinition

Variablen werden als Objekte definiert, die während des Simulationslaufs referenziert und/oder verarbeitet werden können. Die Struktur im Definitionsbereich der Spezifikationsdatei ist wie folgt:

```
VARNAME = TYPE('string', options)
```

wobei `VARNAME` ein frei wählbarer, aber eindeutiger Name der Variable, `TYPE` einer von vier vordefinierten Variablentypen ist und `'string'` den Namen des aufzurufenden Signals enthält (z. B. GCSM-Signalname), gefolgt von zusätzlichen Optionen (`options`):

- **`prnt – prnt=1:`** Ausgabe des Parameterwertes des Signals `'string'` in die Konsolenausgabe während der Simulationslaufzeit [default 0]. Die Werte werden dabei jeden n-ten Zeitschritt ausgegeben, wobei n im Startkommando des ATHLET-Controllers über die Option `--every=n` angegeben (mit $n \in \mathbb{N}$).

Es werden vier vordefinierte Variablentypen `TYPE` bereitgestellt:

- **`Var( path ):`** generische Variablenklasse; erlaubt den Zugriff auf jede verfügbare ATHLET-Variable, die der angegebenen Pfadzeichenfolge (HDF5 path) entspricht.

  Einfaches Beispiel:

  ```
  var1 = Var( 'model gcsm t', prnt=1 )
  ```

  Komplexes Beispiel:

  ```
  var2 = Var(
             'model physical TFO system TFsys_1
              TFYA01HL01 - NODE1 TFLUID'
            )
  ```

  ACHTUNG: `model physical [...]` stellt eine alternative Zugriffsmöglichkeit auf den Plotdatenvektor in ATHLET dar, in der Form wie sie in der HDF5-Ausgabedatei von ATHLET strukturiert ist. Der Plotdatenvektor wird unmittelbar vor dem Schreiben des ersten Zeitschritts erstellt. Das bedeutet, dass zuvor definierte Werte nicht verfügbar sind und alle Daten hierin nur gemäß der Plothäufigkeit aktualisiert werden. Für die meisten Zwecke, die eine ATHLET-Simulation

steuern, ist dies nicht die richtige Wahl. Hier muss direkt auf die Parameterwerte in `STATE DATA` zugegriffen werden. Dabei ist folgende Syntax einzuhalten: `state <ModuleName> <VariablenName>`. Ein Beispiel zur Anwendung ist in List. 3.1 gegeben. Erläuterungen und Listen verfügbarer Größen sind im ATHLET Programmer‘s Manual (/JAC 21/: 5-1ff) hinterlegt.

- **`GCSM( id ):`** direkter Zugriff auf GCSM-Signalwerte über die Signalwertbezeichnung (GCSM ID). Diese Klasse stellt den häufigsten Anwendungsfall dar.

  Beispiel:

```
var3 = GCSM( 'P-L1BEFS' , prnt=1 )
```

- **`TFO( id ):`** Zugriff auf das `TFOHandle` für die angegebene Thermofluidobjekt (TFO) ID. Ein `TFOHandle` ermöglicht den Zugriff auf jede TFO-Variable über ein numpy-Array entlang der Knoten eines TFO. Dies ermöglich den direkten Zugriff auf z. B. den Massenstrom (`GJ`), die Fluidtemperatur (`TFLUID`), etc. eines TFO.

  Beispiel:

```
tfo = TFO( 'PUM00A', prnt=1 )
tfo.tl[0:5]          #< get TL array along TFO-nodes 0,1,2,3,4
any( tfo.tl > 300 ) #< any TFO-node with TL > 300.?
tfo.tl -= 10.0       #< really want to cool down TFO nodes?
```

  HINWEIS: Aufgrund der internen Datenstrukturen und von Namenskonventionen in ATHLET weiß ein `TFOHandle` nicht im Voraus, welche Variablen existieren oder für TFOs sinnvoll und von Bedeutung sind. Es verwendet einfach den **kleingeschriebenen** Namen der Variablen (hier Liquidtemperatur; TL), auf die zugegriffen werden soll, um das interne Datenarray für diese Variable nachzuschlagen. Hier sucht es die Knoten des angegebenen TFO und bildet sie als numpy-Array aus dem Datenarray ab. Erläuterungen und Listen verfügbarer Variablen sind im ATHLET Programmer‘s Manual (/JAC 21/: 5-1ff) hinterlegt.

  ACHTUNG: Die zurückgegebenen numpy-Arrays sind tatsächliche Verweise auf die ursprünglichen ATHLET-Daten. Die Manipulation dieser Arrays wie im obigen Beispiel ändert demnach auch die internen ATHLET-Daten, welche für die Fortsetzung der Simulation verwendet werden.

- **`HCO( id ):`** Zugriff auf das `HCOHandle` für die angegebene Wärmestrukturobjekt (Heat Conduction Object; HCO) ID. Ein `HCOHandle` ermöglicht den Zugriff auf jede HCO-Variable über ein numpy-Array entlang der Knoten des HCO.

  Beispiel:

```
hco = HCO( 'HCPUM00A', prnt=1 )`
hco.hc_tsr          #< get TSR array (2-dim) over HCO-nodes
hco.hc_tsl.shape    #< get shape-tuple of HCO
hco.hc_qhr[0:2,1:3] #< get QHR-subslice of HCO
```

  HINWEIS: Aufgrund der internen Datenstrukturen und von Namenskonventionen in ATHLET weiß ein `HCOHandle` nicht im Voraus, welche Variablen existieren oder für HCOs sinnvoll und von Bedeutung sind. Es verwendet einfach den **kleingeschriebenen** Namen der Variablen (hier Strukturtemperatur auf der rechten Seite des HCO; HC_TSR), auf die zugegriffen werden soll, um das interne Datenarray für diese Variable nachzuschlagen. Hier sucht es die Knoten des angegebenen HCO und bildet sie als 2D-numpy-Array aus dem Datenarray ab. HCO-Variablen werden i. d. R. mit `HC_<name>` benannt. Erläuterungen und Listen verfügbarer HCO-Variablen sind im ATHLET Programmer‘s Manual /JAC 21/ hinterlegt.

  ACHTUNG: Die zurückgegebenen numpy-Arrays sind tatsächliche Verweise auf die ursprünglichen ATHLET-Daten. Die Manipulation dieser Arrays wie im obigen Beispiel ändert demnach auch die internen ATHLET-Daten, welche für die Fortsetzung der Simulation verwendet werden.

3.2.2.2 Trigger-Definition

Trigger dienen der Erstellung von Bedingungen durch Anwendung logischer Operationen auf die definierten Variablen. Sie werden wie folgt initialisiert:

```
TRGNAME = Trigger('string', options)
```

wobei `TRGNAME` ein frei wählbarer, aber eindeutiger Name ist, `'string'` eine boolsche Definition der Steuerungslogik enthält, welche sich auf die oben definierten Variablen bezieht (z. B. `'var1 < 100.0'`) und `options` die Möglichkeit bietet zusätzliche

Optionen für die Behandlung der logischen Operationen vorzugeben. Optionen werden durch Kommas getrennt. Implementierte Optionen sind:

- **`prnt – prnt=1:`** Ausgabe des Parameterwertes des Signals `'string'` in die Konsolenausgabe während der Simulationslaufzeit [default 0]. Die Werte werden dabei jeden n-ten Zeitschritt ausgegeben, wobei n im Startkommando des ATHLET-Controllers über die Option `--every=n` angegeben (mit $n \in \mathbb{N}$) ist.

- **`duration`**: enthält eine Angabe in Sekunden, wie lange die logische Bedingung ununterbrochen andauern muss, bevor der Trigger ausgelöst wird. Die Angabe ist optional und der default-Wert wird mit 0 s gesetzt. Es ist nur die Eingabe von Zahlen erlaubt (z. B. `duration=2.0` -> logische Bedingung muss mindestens 2.0 s andauern, bevor der Wert des Triggers `TRUE` annimmt).

- **`delay`**: enthält eine Angabe in Sekunden um welche Zeitspanne die Trigger-Auslösung nach Erfüllen der logischen Bedingung verzögert werden soll. Die Angabe ist optional und der default-Wert wird mit 0 s gesetzt. Es ist nur die Eingabe von Zahlen erlaubt (z. B. `delay=2.0` -> die Triggerwirkung wird um 2.0 s nach Erfüllen der logischen Bedingung verzögert).

- **`message`**: eine vom Benutzer vorgegebene Nachricht in Form eines Strings wird bei Erfüllen der logischen Bedingung in die Konsole ausgegeben (z. B. `message='Bedingung TRG1 erfüllt'`).

In `'string'` werden logische Bedingungen definiert, welche Steuerungsaktionen auslösen können. Dabei besteht ein hoher Freiheitsgrad bei der Eingabe von logischen Befehlsketten und der Form der Verweise auf relevante Variablen. So können innerhalb der Logikdefinition in `'string'` Variablen verwendet werden, welche im Variablendefinitionsabschnitt der Spezifikationsdatei definiert wurden oder direkt auf ATHLET-interne Parameter zugegriffen werden, indem der vollständige ATHLET-interne Verweispfad angegeben wird. Erläuterungen und Listen verfügbarer ATHLET-interner Parameter sind im ATHLET Programmer‘s Manual /JAC 21/ hinterlegt. Beispiele für den Umgang mit und die Definition von Triggern und logischen Bedingungen in Spezifikationsdateien sind in List. 3.1 aufgeführt.

List. 3.1 Beispiele zur Definition von Triggern

```
#name              #variable #limit #wert  #bool #variable #limit  #wert     #option
 TRG0  = Trigger('  VAR1       >=    1.5    and   VAR1        <     3.0  ' ,  prnt=1)

#name                          #< logische bedingung >                    #option        #option
 TRG1  = Trigger(' "model gcsm t" < 50.0 and "state cdnw iilo 3" < 8 ' , duration=0.0 , delay=5.0)

#name                          #< logische bedingung >
 TRG2  = Trigger(' VAR1 < 600 and VAR2 > 200 or math.sin(VAR1) < 0.56' )

#name                          #< logische bedingung >                                    #option
 TRG3  = Trigger(' VAR1 >= 0.1 ', duration=0.1, delay=0.05, message='TRG3 was triggered!', prnt=1 )
```

3.2.2.3 Action-Definition

Actions dienen der Ausführung von Steuerungsmaßnahmen durch Zugriff und Änderung der zugehörigen Signalwerte in ATHLET. Die Initialisierung einer Action erfolgt mit folgender Syntax:

```
ACT1= Action('<GCSM_SIGNAL_NAME>','<TRIGGER>',Action.<ACTION>,options)
```

wobei `'TRIGGER'` ein einzelner Trigger (z. B. `'TRG0'`) oder eine boolesche Kombination von Triggern sein kann, z. B. `'TRG0 and TRG1'`, `'TRG0 or TRG1'`, `'TRG0 and (TRG1 or TRG2)'`. Innerhalb des ATHLET-Controller sind eine Reihe häufig verwendeter Steuerungsbefehle (`ACTION`) vordefiniert. Eine Übersicht zu diesen vordefinierten Actions und den zugehörigen Optionen (`options`) bietet Tab. 3.1.

Neben diesen im Controller explizit vorgesehenen Steuerungsbefehlen steht es dem Anwender frei, nahezu beliebig komplexes Verhalten als Wirkung auf die Zielsignalgröße bei Auslösung einer Action zu definieren. Dazu können im Abschnitt „Funktionsdefinition“ der Spezifikationsdatei Python-Funktionen definiert werden, welche bei Auslösung einer Action aufgerufen und ausgeführt werden.

Tab. 3.1 Vordefinierte Steuerungsbefehle und zugehörige Optionen

Action	Optionen	Beschreibung
`up`	-	setzt den Zielsignalwert auf 1
`down`	-	setzt den Zielsignalwert auf 0
`jump`	target value	setzt den Zielsignalwert auf den angegebenen Target-Wert
`ramp`	target value, gradient	ändert Zielsignalwert linear auf den Target-Wert mit dem angegebenen zeitlichen Gradienten in [1/s]
`terminate`	-	beendet den ATHLET-Lauf regulär, wenn ausgelöst
`writeRestart`	filename	weist ATHLET an, bei Auslösung einen Restart-Punkt zu erstellen
`loadRestart`	n [index]	weist ATHLET an, den n-ten Restart-Punkt zu laden ($n \in \mathbb{Z}$; für negative Werte von n wird der n-letzte Restart-Punkt geladen)
-	prnt	Ausgabe des Action-Zustandes (True/False) in die Konsolenausgabe; prnt=1/prnt=0
-	tBeg	Startzeitpunkt eines Zeitfenster in dem die Action aktiv werden kann; tBeg={time value} (time value $\in \mathbb{Q}$)
-	tEnd	Endzeitpunkt eines Zeitfensters in dem die Action aktiv werden kann; tEnd={time value} (time value $\in \mathbb{Q}$)

3.2.2.4 Funktionsdefinition

Die Protokolle funktionieren als eigenständige Python-basierte Programmabschnitte. Dies ermöglicht neben der Anwendung vordefinierter Funktionalitäten wie der Definition von Objekten der Klassen `Variable`, `Trigger` und `Action` auch die Definition eigener Funktionen und Klassen innerhalb der Protokolle, welche dann bestimmte Aufgaben zur Steuerung und Überwachung einer Simulation übernehmen können. Dabei steht prinzipiell der gesamte Funktionsumfang der Programmiersprache Python zur Verfügung, was dem Anwender größtmögliche Freiheit bei der Entwicklung von Steuerungs- und Kontrollmaßnahmen eröffnet.

Ein Beispiel zum Speichern des Auslösezeitpunktes einer bestimmten `Action` ist in List. 3.2 dargestellt. Der damit ermittelte Wert kann anschließend zur Steuerung nachfolgernder Auslösungen verwendet werden.

List. 3.2 Beispiel eines Ablaufprotokolls in ATHLET-Anwendungskonvention

```
-# -*- coding: utf-8 -*-
# ========================================================================== #
##############################################################################
####             Definition eigener Funktionalitäten                      ####
##############################################################################

# Beispiel: Speichern der Auslösezeit einer Aktion
ttime  = 9E99
tcause = True
def lockTime( action ):
   global vt
   global ttime
   global tcause
   if tcause and action.cause.value:
      ttime  = vt.value
      tcause = False

##############################################################################
####                         Variablendefinition                          ###
##############################################################################

#### Trigger-Variablen:
Vt  = Var(  'model gcsm t'  , prnt=1 )

#### Info-Variablen:
I1  = GCSM( 'CSRL41S011'    , prnt=1 )

#### Target-Variablen:
SGPLOT = GCSM( 'SGPLOT'     , prnt=1 )
TAR1   = GCSM( 'CHYP10S231' , prnt=1 )
TAR2   = GCSM( 'YP10CF300'  , prnt=1 )

##############################################################################
####                        Logikdefinition                               ###
##############################################################################

# erhält den Wert 'TRUE' bei Überschreiten von t = 7001 s
t0   = Trigger( ' Vt >= 7001 '    , prnt=1 )

# Beenden der Simulation bei Überschreiten von t = 10500.0 s
tend = Trigger( ' Vt >= 10500.0 ' , prnt=1 )

##############################################################################
####                         Action-Definition                            ###
##############################################################################

# Anpassung der Zeitschrittweite zur Ausgabe in den Plot-Vektor
a0   = Action( 'SGPLOT'    ,'t0'  , Action.jump, 0.1, tBeg=600, tEnd=620)

# Öffnen des DH-Sprühventils
a1   = Action( 'CHYP10S231','t0'  , Action.jump, 1.0, prnt=1 )

# Aufruf der Beispielfunktion <lockTime> zur Speicherung der Auslösezeit
a2   = Action( 'PROB.TIME' ,'t0'  , lockTime , prnt=1)

# Reguläres Beenden der Simulation
aend = Action( 'RUNEND'    ,'tend', Action.terminate, prnt=1 )

# ---------------------------------------------------------------------- #
```

3.2.3 ATHLET-Controller-Befehle in der Kommandozeile

Ein vollständiges generisches Beispiel für die Ausführung einer ATHLET-Simulation mit dem ATHLET-Controller aus der Kommandozeile ist in List. 3.3 dargestellt.

List. 3.3 Startbefehl zur Ausführung einer Simulation mit dem ATHLET-Controller

```
python ~\PATH\TO\ATHLET_Controller
          --lib="LIB"
          --input="INPUT"
          --runId="RUNID"
          --protocol="ACTION"
          --every=TIMESTEPS
Optional:
          --opts="OPTS"
          --tKill=TKILL
          --showtime
          --verbosity=LEVEL
          --version
          --workdir="DIR")
```

Tab. 3.2 Erläuterungen zu verwendbaren Optionen des Startbefehls

Option	Beschreibung
LIB	Shared Library (.dll) des zu verwendenden ATHLET-Binarys
INPUT	ATHLET input file (Eingabedatensatz des Analysesimulators; *.inp)
RUNID	ATHLET's Problem und Run ID [default: pid.rid]
ACTION	Spezifikationsdatei/Ablaufprotokoll [default: ./action.py]
TIMESTEPS	Ausgabe von Informationen aus dem Controller in die Kommandozeile alle x Zeitschritte ($x \in \mathbb{Z}$) [default: -1 (keine Ausgabe)]
OPTS	Liste optionaler ATHLET-Argumente (insb. auch für gekoppelte Analysen mit Quabox-Cubbox oder Cocosys) [default:]
TKILL	Vorgabe eines Simulationsabbruchs über die Kommandozeile t ($t \in \mathbb{Q}$) [default: -1.0 (verwenden von TE im Input bzw. der Vorgabe im Ablaufprotokoll)]
--showtime	Ausgabe der Problemzeit in die Konsolenausgabe
LEVEL	Verbosity Level x ($x \in \mathbb{N}$); Umfang der Informationsausgabe in die Kommandozeile [default: 1]
-- version	Ausgabe verwendeter Programmversionsbezeichnungen bei Ende der Simulation
DIR	Arbeitsverzeichnis zur Ausführung des Prozesses [default: .]

Da der ATHLET-Controller eine vollumfängliche Restart-Fähigkeit besitzt, können Simulationen auch auf im Vorfeld erzeugter Simulationsergebnisse aufgesetzt werden. Restart-Punkte können durch die unter Kapitel 3.2.2.4 beschriebenen Funktionsdefinitionen in den Ablaufprotokollen und damit abhängig vom Anlagenzustand ausgeschrieben und eingelesen werden. Dies hilft bei unerwarteten Systemverhalten die Problemanalyse zu beschleunigen, da so abhängig vom betreffenden Ereignis lange Anlaufrechenzeiten deutlich verkürzt werden können.

Für das Starten von Restart-Simulationen wird die ATHLET-Controller-Option

```
--opts="OPTS"
```

verwendet. Dazu müssen die Restart-Informationen wie im folgenden Beispiel übergeben werden:

```
--opts="-rf ~\PATH\TO\RESTARTFILE\PID.RID.re -iwber 999"
```

Also Ablageort und Name des Restart-Files (`-rf`) der Vorläufersimulation und Identifikationsnummer des gewünschten Restart-Punktes. Im o. g. Beispiel wird mit der Codierung „`999`“ der letzte verfügbare Restart-Punkt gewählt. Nähere Information zum Umgang mit Restarts finden sich um ATHLET User‘s Manual /AUS 21/.

4 Erstellung von CI-Konfigurationen zur Durchführung von Analysen unter Verwendung von Analysesimulatoren

4.1 Einführung in das Konzept der kontinuierlichen Integration

Kontinuierliche Integration (Continuous Integration, CI) beschreibt ein Verfahren aus der Softwareentwicklung, bei der die Entwickler Codeänderungen in einem zentralen Repository zusammenführen. Die Software wird dann mit diesen Änderungen automatisiert erstellt und getestet. Die zentrale Ablage sowie die Durchführung der Tests erfolgt innerhalb einer dafür bereitgestellten Serverinfrastruktur mit spezialisierter Software, welche im Folgenden als CI-Plattform bezeichnet wird. Das Hauptziel einer CI-Plattform ist es, die Entwickler durch das automatische Kompilieren der entwickelten Software und die automatische Durchführung von Tests mit entsprechenden, automatisch generierten Berichten zu unterstützen und somit zur Qualitätssicherung der Entwicklungsergebnisse beizutragen.

Eine GRS-Plattform zur kontinuierlichen Integration wurde im Rahmen des Vorhabens RS1538 /SCH 18/ auf Basis des Softwarepakets Jenkins /PAT 17/ aufgebaut. In der Vergangenheit wurde in der GRS dieser Jenkins-Server verwendet, um die verschiedenen Versionen des GRS-Systemcodes ATHLET automatisch zu kompilieren und zu testen. Ab 2020 fand eine Umstellung der GRS internen Software-Verwaltung auf Gitlab /CHO 20/ statt. Gitlab ist eine Anwendung zur Versionsverwaltung für Softwareprojekte und stellt zukünftig die zentrale Plattform für sämtliche programmtechnische Entwicklungen der GRS dar. Es verfügt auch über einen Server zur kontinuierlichen Integration (CI), die zukünftig von allen GRS Softwareentwicklungsprojekten eingesetzt wird.

4.2 Umsetzung des CI-Konzeptes zur Anwendung im Kontext der Verifizierung von Analysesimulatoren

4.2.1 Konzeptbeschreibung und Vorstellung des Ablaufs

Für die Umsetzung einer automatischen Verifikation von Analysesimulatoren im Rahmen einer CI-Infrastruktur ist ein Zusammenspiel von Entwicklereingaben, Speicherinfrastruktur und CI-Umgebung erforderlich. An jede einzelne Komponente dieser Verifikationskette werden dabei verschiedenartige, hohe Anforderungen gestellt.

Die Bearbeitung und Weiterentwicklung von Datensätzen, muss parallel durch mehrere Entwickler möglich sein, wobei Konflikte bei der Bearbeitung zu vermeiden sind. Insbesondere um sicherzustellen, dass jederzeit eine lauffähige und verifizierte Version des Analysesimulatordatensatzes für die Durchführung von Analysen zur Verfügung steht, auch während laufender Entwicklungsarbeiten. Eine zeitgemäße Lösung für diese Anforderung ist die Anwendung von Verwaltungssystemen mit Versionskontrollmanagement (Repositorys). Gitlab /CHO 20/ bietet hierzu eine stabile Lösung mit großem Funktionsumfang an. So bietet es ein hohes Maß an Kontrolle über die Zugriffsrechtevergabe und zusätzliche Optionen zur Projektplanung und Teamkommunikation, was einer effektiven und effizienten Durchführung von (Weiter-)Entwicklungsarbeiten zuträglich ist.

Da die Verifikation von Analysesimulatoren auf der Durchführung einer großen Zahl, teilweise sehr komplexer und langwieriger Simulationen basiert, fallen dabei große Datenmengen an. Diese sind zu Zwecken der Überprüfung und Reproduzierbarkeit für gewisse Zeit vorzuhalten. Aus diesem Grund muss die verwendetet Speicherinfrastruktur eine entsprechende Größe aufweisen. Außerdem muss sie sowohl von allen Entwicklern als auch durch automatische Prozeduren des CI-Servers zentral und zuverlässig zugänglich sein. Für die Ablage der Daten ist eine einheitliche Logik und Nomenklatur vorzusehen, welche durch auf der CI laufende Programme interpretiert werden kann.

Die CI-Umgebung schließlich muss Software zur Verfügung stellen, welche die Kommunikation mit den Repositorys und der Speicherinfrastruktur unterstütz und in der Lage ist, komplex verschachtelte Aufgaben abzuarbeiten. Zusätzliche Anforderungen im konkreten Fall bestehen bzgl. der Kompatibilität mit der Programmiersprache Python und der Möglichkeit, zusätzliche Software-Pakete einzuladen und bestimmte Codes in der Vorbereitung einer Verifikation zu kompilieren.

Im Rahmen des vorgestellten Projektes wurde für die automatische Verifikation von Analysesimulatoren ein entsprechendes Konzept erstellt. Eine schematische Darstellung der fortlaufenden und automatisierten Verifikation ist in Abb. 4.1 gegeben. Dabei erarbeitet ein Entwickler oder ein Team von Entwicklern einen Datensatz für einen Analysesimulator oder führt an diesem Wartungen oder Erweiterungen durch. Diese Arbeiten werden zentral in einem Git-Repository auf dem GRS Gitlab-Server gespeichert. In einem Git-Repository auf demselben Server liegen datensatzspezifische Spezifikationsdateien (Ablaufprotokolle), wie sie in Kapitel 5.2 vorgestellt werden, die zur Steuerung des Datensatzes dienen und die Durchführung von Simulationen eines vordefinierten Satzes an Ereignissen ermöglichen. Die Ergebnisse dieser Ereignissimulationen können gegen

zugehörige Grenzkurven abgeglichen werden, welche als Akzeptanzkorridore für Abweichungen im Systemverhalten dienen. Die Informationen zu diesen Grenzkurven sind ebenfalls in dem Git-Repository hinterlegt.

Für die Durchführung der Verifikation werden Programmabläufe innerhalb der GitCI-Serverinfrastruktur in Form von YAML-Files definiert, welche die Steuerung von Pipelines und Jobs übernehmen. YAML ist eine vereinfachte Auszeichnungssprache, die es erlaubt, Daten wie die Konfiguration einer Pipeline, d. h. Konfigurationsvariablen und Befehlsskripte, in einem maschinenlesbaren Format abzuspeichern, das gleichzeitig einfach von Nutzern erstellt und angepasst werden kann. Eine Pipeline beinhaltet eine Reihe von Jobs, wobei Jobs automatisiert regelbasierte Aufgaben abarbeiten. Jobs werden demnach innerhalb von Pipelines ausgeführt. In Abb. 4.1 ist eine CI-Pipeline schematisch für die Anwendung im automatisierten Verifikationsverfahren dargestellt. Sie beinhaltet Jobs, welche den Import und das Kompilieren von Software, die Auslösung, Steuerung und Überwachung von ATHLET-Simulationen mit Hilfe des ATHLET-Controllers sowie die Ergebnisüberprüfung, Reporterstellung und Ablage der Ergebnisse in der Speicherinfrastruktur übernehmen.

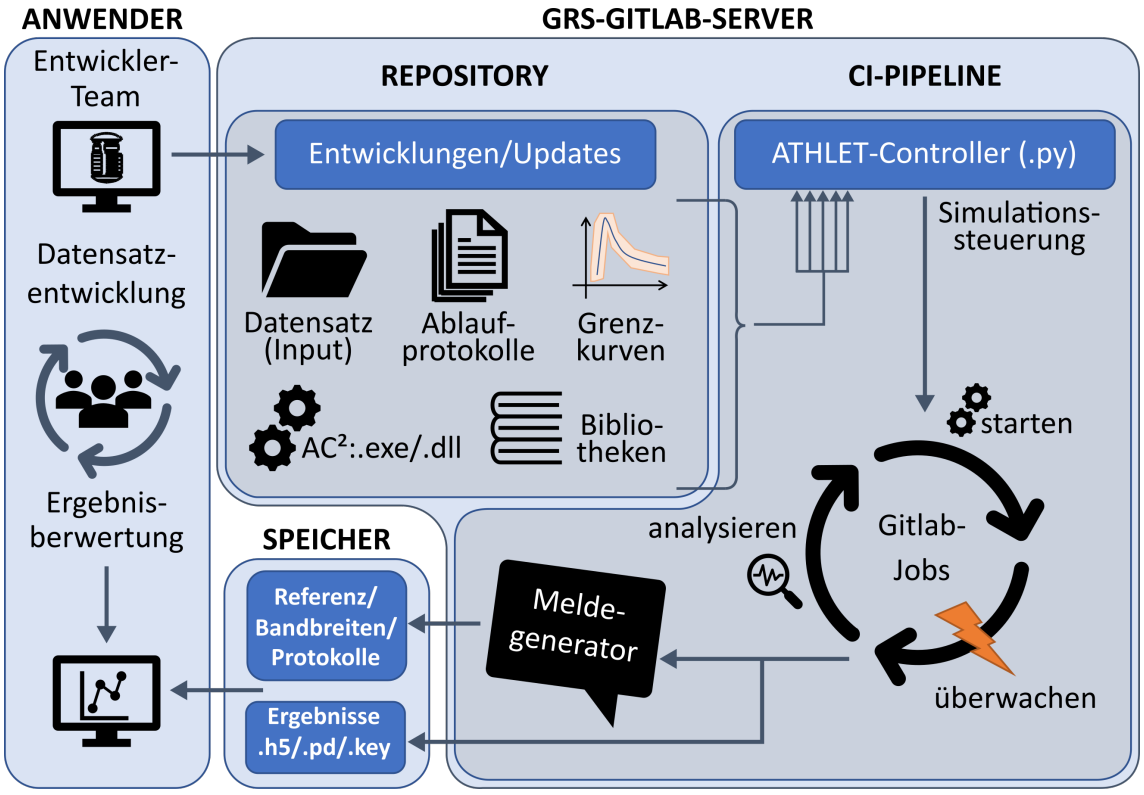

Abb. 4.1 Konzept der automatischen Verifizierung von Analysesimulatoren

Eine detaillierte Aufschlüsselung einzelner Jobs, ihrer Aufgaben und Hierarchie im entwickelten automatischen Verifikationsverfahren erfolgt schematisch in Abb. 4.2. Der Ablauf besteht aus sechs Stufen („Stages“):

- **Prepare (I)**: Je nach Startgrund des Jobs (nach einen Checkin automatisch bzw. manuell durch einen Nutzer) und gegebenenfalls nach Vorgaben des Nutzers wird der weitere Ablauf der Pipeline automatisch generiert, indem eine neue YML-Datei erstellt wird, die die Einstellungen bzw. den Code für die Jobs der weiteren Stufen der Pipeline enthält.
- **Trigger**: Es wird eine nachgeordnete Pipeline („child pipeline“) auf Basis der in der vorhergehenden Stufe generierten YAML-Datei gestartet.
- **Prepare (II)**: In mehreren, parallellaufenden Jobs werden die notwendigen Programme (ATHLET und Plugins), die Eingabedatensätze der zu simulierenden Reaktoren, sowie alle anderen Hilfsprogramme geklont, gegebenenfalls kompiliert oder anderweitig vorbereitet und als Artefakte für die weitere Verwendung innerhalb der Pipeline gespeichert. So werden immer die gleichen Eingabedaten bzw. Programme für alle Jobs einer Pipeline verwendet, um die Reproduzierbarkeit der Ergebnisse sicherzustellen.
- **Bases_cases**: Parallele Durchführung der Simulationen, die neu und ohne Restart durchgeführt werden. Die Restartdateien dieser Simulationen werden als Artefakte gespeichert, um gegebenenfalls in der nächsten Stufe verwendet werden zu können.
- **Cases**: Parallele Durchführung der Simulationen, die auf Basis von Restarts ausgeführt werden.
- **Summary**: Erstellen des Ergebnisreports für die Entwickler

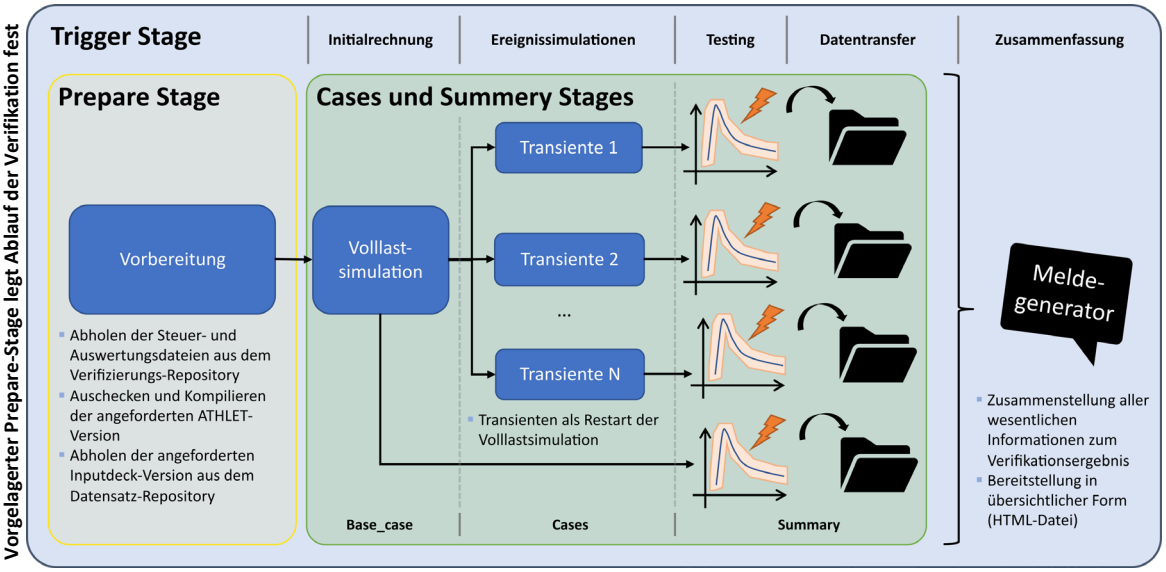

Abb. 4.2 Schematische Darstellung der programmtechnischen Umsetzung der Verifizierungsprozedur in Gitlab-CI

4.2.1.1 Die einzelnen Stufen und Jobs der Verifizierungsprozedur

Nach einem manuellen oder automatischen Start der Pipeline (siehe Abschnitt 4.2.2) wird in der ersten Stufe (Prepare-I) auf Basis von Parametern die neue YAML-Datei `child-ci.yml` generiert, die die Anweisungen für alle weiteren Stufen enthält. Für alle Parameter sind Standardwerte definiert, die bei einem automatischen Start der Pipeline zum Einsatz kommen und die bei einem manuellen Start vom Nutzer überschrieben werden können. Die Beschreibung der ersten Stufe „Prepare“ der Pipeline ist fest kodierter Bestandteil der Projektrepositorys (`.gitlab-ci.yml`). Die neu zu erstellende dynamische YAML-Datei `child-ci.yml` wird aus verschiedenen Musterdateien zusammengebaut.

Der erste Teil von `child-ci.yml` besteht aus dem Inhalt der Datei `.ci/child-ci-header.yml`, in der die Jobs der Stufe „Prepare“ der neu erzeugten Pipeline definiert werden[1]. Um den Speicherbedarf der Artefakte der Pipelines auf dem Gitlab-CI-Server zu reduzieren, wird bei der dynamischen Generierung der Stufe „Prepare“ festgelegt,

[1] Die Stufe „Prepare“ ist also zweimal vorhanden: Einmal als Bestandteil der fest kodierten Pipeline in der Datei `.gitlab-ci.yml` und einmal in der dynamisch generierten Pipeline definiert in `.child-ci.yml`.

dass die gespeicherten Artefakte dieser Stufe für die verkürzte Ausführung der Pipeline (RUN_MODE „fast“) nur für einen Tag gespeichert werden, während die Speicherfrist für eine vollständige Ausführung auf drei Wochen gesetzt wird. Die lange Dauer im zweiten Fall ist notwendig, um sicherzustellen, dass auch bei einer unerwartet langen Dauer der Pipeline, z. B. weil durch eine Datensatzänderung oder durch eine Überlastung der Gitlab-CI-Serverinfrastruktur die Simulationen plötzlich langsamer laufen, allen Jobs der Pipeline die Artefakte auch wirklich zur Verfügung stehen.

Als nächstes wird auf Basis der Datei `case_spec.json` (siehe weiter unten) und der Musterdatei `.ci/child-ci-case.yml` die Definitionen der Jobs für die einzelnen Rechenfälle generiert. Der Namen der einzelnen Jobs wird aus dem Reaktornamen und der Fallnamen generiert.

Da die Gitlab-CI nur ein sehr rudimentäres System zur Verwaltung von parallelen Jobs zur Verfügung stellt und die Serverinfrastruktur nicht überlastet werden sollte, wurde ein eigener Mechanismus entwickelt, der die Anzahl der parallel ausgeführten Simulatorjobs begrenzt. Die maximale Anzahl paralleler Jobs wird durch den Pipelineparamter `NUM_RG` vorgegeben. Um die Ressourcen möglichst effizient auszuschöpfen, werden die Jobs in umgekehrter Reihenfolge ihrer erwarteten Laufzeit gestartet. Langlaufende Jobs starten also möglichst früh. Dies wird erreicht, indem die Jobs in dieser Reihenfolge in die Datei `child-ci.yml` geschrieben werden. Obwohl die Jobs in der Oberfläche der Gitlab-CI in alphabetischer Reihenfolge angezeigt werden, erfolgt die Ausführung in der Reihenfolge der Eintragung in der YAML-Datei. Damit wird sichergestellt, dass langlaufende Jobs nicht sehr spät gestartet werden und ein Teil der Rechenressourcen leer steht, da schon alle schnelllaufenden Jobs abgearbeitet wurden.

In der GRS-Gitlab-Serverinfrastruktur erlauben die vorhandenen Rechnerknoten unterschiedlich lange Laufzeiten für Jobs. Es wird deshalb bei der Generierung der einzelnen Jobs sichergestellt, dass sie nur auf Rechnerknoten laufen, die eine ausreichend lange Laufzeit erlauben.

Je nachdem, ob der Rechenfall auf Basis eines Restarts durchgeführt wird oder frisch startet, wird er entweder der Stufe „Base_cases“ oder „Cases“ zugewiesen.

Die Parameter des Aufrufs des ATHLET-Controllers (siehe Abschnitt 3.2.3) werden ebenfalls auf Basis der Daten in der Datei `case_spec.json` generiert. Der Aufruf des Controllers, sowie die Durchführung der Tests und das Speichern der Simulations-

dateien ist dabei im PowerShell-Skript `.ci/run_case.ps1` gekapselt. Letzteres erlaubt es, die Simulation und die Tests auch außerhalb der Gitlab-CI-Serverinfrastruktur, z. B. auf dem PC eines Datensatzentwicklers, durchzuführen. Dazu muss beim Aufruf des Skripts das Kommandozeilenflag `-debug` gesetzt werden.

Abhängig von der Rechenzeit in den bisherigen Simulatorläufen (gespeichert in der Variable `eta` („estimated time of arrival“) in der Datei `case_spec.json`) und einem Zuschlag von 25 % wird für jeden Job ein Timeout vorgegeben. Durch Änderungen im Datensatz oder auch eine neue Version von ATHLET kann es unter Umständen dazu kommen, dass ein Job plötzlich sehr viel länger läuft, da z. B. die Zeitschrittweite von ATHLET sehr kleine Werte annimmt. Bricht man einen solchen Job manuell ab, so kann die nächste Stufe der Pipeline nicht ausgeführt werden und damit auch kein Ergebnisreport generiert werden. Die Ergebnisse der anderen Jobs wären damit quasi verloren. Durch einen gezielten Abbruch des ATHLET-Laufs, ohne dass der Job als solches als gescheitert gekennzeichnet wird, können die Daten aller Jobs immer noch ausgewertet werden. Der Abbruch eines einzelnen Jobs wird im Meldegenerator automatisch erkannt, da überprüft wird, ob die vorgegebene Endzeit der jeweiligen Simulation erreicht wird. Ein Abbruch einer Simulation erscheint deshalb als ein Fehler im Testbericht. Ein weiterer Vorteil eines Timeoutmechanismus ist, dass die Gitlab-CI-Serverinfrastruktur nicht übermäßig beansprucht wird.

Als letztes wird der Datei `child-ci.yml` für die Stufe „Summary“ für jeden zu bearbeitenden Reaktor ein Job zur Ausführung des Meldegenerators auf Basis der Musterdatei `.ci/child-ci-summary.yml` hinzugefügt. Beim Meldegenerator handelt es sich um das PowerShell-Skript `.ci/create_report.ps1`. Dieses wertet die Testergebnisdateien aus und generiert den in Kapitel 7 beschriebenen Testbericht. Dieses Skript kann ebenfalls außerhalb der Gitlab-CI-Serverinfrastruktur ausgeführt werden, z. B., um schon die Ergebnisse von bereits beendeten Simulationsläufen einer noch nicht beendeten Pipeline vorab auszuwerten[2].

Der Job „tigger-cases-jobs“ in der Stufe “Trigger” startet eine neue Pipeline auf Basis der Datei `child-ci.yml`.

[2] Um die Artefakte bereits beendeter Jobs einer Pipeline vom Gitlab-Server herunterzuladen, liegt im Verify-Repository das Pythonskript `download_all_artifacts.py` bereit.

In der Stufe „Prepare“ der dynamisch generierten Pipeline in der Datei `child-ci.yml` werden im Job „athlet-job“ die Binarys von ATHLET und den verwendeten Plugins (teleperm, atlas_process, hdf5_writer und QUABOX/CUBBOX) auf Basis der jeweils als Parameter übergebenen Git-Versionen gebaut und als Artefakte für die weiteren Jobs der Pipeline zwischengespeichert. Im Job „controller-job“ werden die Skriptdateien des ATHLET-Controllers, die von ihm verwendeten Bibliotheken sowie die Grenzkurven aus den Git-Repositorys geklont und zwischengespeichert. Im Job „input-job“ werden die Eingabedatensätze für die zu simulierenden Reaktoren geklont, aus ihren Einzeldateien zusammengebaut und zwischengespeichert. Im Job „python-libs-job“ werden die Pythonbibliotheken für die Durchführung der automatisierten Tests geklont und zwischengespeichert. In jedem der Jobs werden auch Angaben über die jeweilige so genannten Hashes der geklonten Versionen mitgespeichert, um Simulationsergebnisse jederzeit eindeutig den verwendeten Codeversionen und Modulen zuordnen zu können. Dieses Vorgehen stellt sicher, dass alle nachfolgend gestarteten Simulationen auf den gleichen Versionen von ATHLET, des Eingabedatensatzes bzw. der Protokolldateien und der Grenzkurven basieren, da diese nur einmal für jeden Lauf der automatischen Verifikation zusammengestellt werden. Die Vergleichbarkeit aller innerhalb einer Pipeline erzeugten Ergebnisse ist somit stets gegeben. Dieses Vorgehen ist notwendig, da der gesamte Verifizierungsprozess mehrere Tage in Anspruch nimmt. Es ist demnach denkbar, dass sich sonst während der Laufzeit der Verifizierung Änderungen in den verwendeten Komponenten ergeben könnten, die zu inkonsistenten bzw. nicht reproduzierbaren Ergebnissen führen könnten.

In der Stufe „Base_cases” werden die ihr zugeordneten Jobs durchgeführt, die jeweils eine Simulation ohne Restart durchführen. Es werden auch die Tests durchgeführt und die Ergebnisdaten auf einem Netzlaufwerk gespeichert (siehe Abb. 4.1 und Abb. 4.3). Wurden alle Tests bestanden, so werden ältere Ergebnisdateien gelöscht, um Speicherplatz zu sparen.

In der Stufe „Cases“ werden die Jobs ausgeführt, die mit einem Restart beginnen. Dabei werden die Abhängigkeiten der Jobs explizit gesetzt, um nur auf den Jobs aus der Stufe „Bases_cases“ zu warten, durch den die jeweils notwendige Restartdatei generiert wird. Die dadurch erreichte lange Einschwingzeit in Form der Volllastsimulation garantiert einen stabilen Ausgangszustand für die Transienten. Gleichzeit wird für alle transienten Ereignisse aus der Simulationsmatrix die simulierte Echtzeit um 7.000 s verkürzt, was einer Zeitersparnis pro Simulation in der Größenordnung von vier Stunden entspricht.

Wie für die „Bases_cases“ werden auch hier die Tests durchgeführt, die Ergebnisse gespeichert und gegebenenfalls alte Daten gelöscht.

In der abschließenden Stufe „Summary“ werden die reaktorspezifischen Jobs dann gestartet, wenn alle reaktorspezifischen Simulationsjobs beendet sind.

Über Start, Ende und relevante Zustandänderungen bei Durchführung eines Verifikationslaufs werden die Entwickler über ihre GRS-Gitlab-Accounts und per E-Mail informiert.

4.2.1.2 Einbinden neuer Fälle in die Testmatrix

Für die Einbindung eines neuen Analysesimulators in den Verifikationsprozess müssen folgende Voraussetzungen erfüllt sein:

- Ein Unterverzeichnis in `/Ablaufprotokolle` des Verify-Repositorys[3] ist benannt mit der Abkürzung (`ABK`) des zu verifizierenden Anlagendatensatzes. Darin enthalten sind:
  - Spezifikationsdateien (Ablaufprotokolle) zur Steuerung der Ereignisse in der Simulationsmatrix mit einer systematischen aber frei wählbaren Benennung (z. B. `D2-01_FD_SiVentil`, etc.),
  - eine Signalliste (`<ABK>_Signals.xlsx`) mit Informationen zu den simulationsspezifischen Anlagenparametern zur Verifikation,
  - die Datei `case_spec.json` mit der Liste der Ereignisse in der anlagenspezifischen Simulationsmatrix,
  - die Datei `simulation_times.json` mit der Liste der Start- und Endzeiten der Simulationen der einzelnen Ereignisse in der Simulationsmatrix.
- Auf dem zentralen Speicher für Ergebnisdateien ist eine Verzeichnisstruktur entsprechend der Darstellung in Abb. 4.3 mit:
  - einem Unterverzeichnis in `GitlabCI` benannt nach der Abkürzung des entsprechenden Anlagendatensatzes (`<ABK>`) mit jeweils einem Verzeichnis für Test

[3] https://gitlab.grs.de/grs/AnalysisSimulators/dsa_public/verify

„fast“ und Test „normal“ (wird automatisch von den Gitlab-CI-Jobs erzeugt, falls nicht vorhanden),

  - einem Verzeichnis mit Referenzrechnungen zur Erstellung der Grenzkurven zur Bewertung benannt nach der Abkürzung des zu verifizierenden Anlagendatensatzes (`<ABK>`) (ist manuell auf fs-gar08 anzulegen).

- Ein Unterverzeichnis in `/prepare_tests` des Verify-Repositorys ist benannt mit der Abkürzung (`<ABK>`) des zu verifizierenden Anlagendatensatzes. Darin enthaltenen sind:

  - die Datei `prepare_test_eta.ipynb` mit einem Funktionspaket zur Ermittlung der erwarteten Simulationsdauer aus den Referenzrechnungen (**Anpassungen** durch den Entwickler erforderlich),

  - die Datei `prepare_test_limits.ipynb` mit einer Liste der einzubeziehenden anlagenspezifischen Setups aus den Referenzen (vgl. Abb. 4.3) in `cases_root` (**Anpassungen** durch den Entwickler erforderlich),

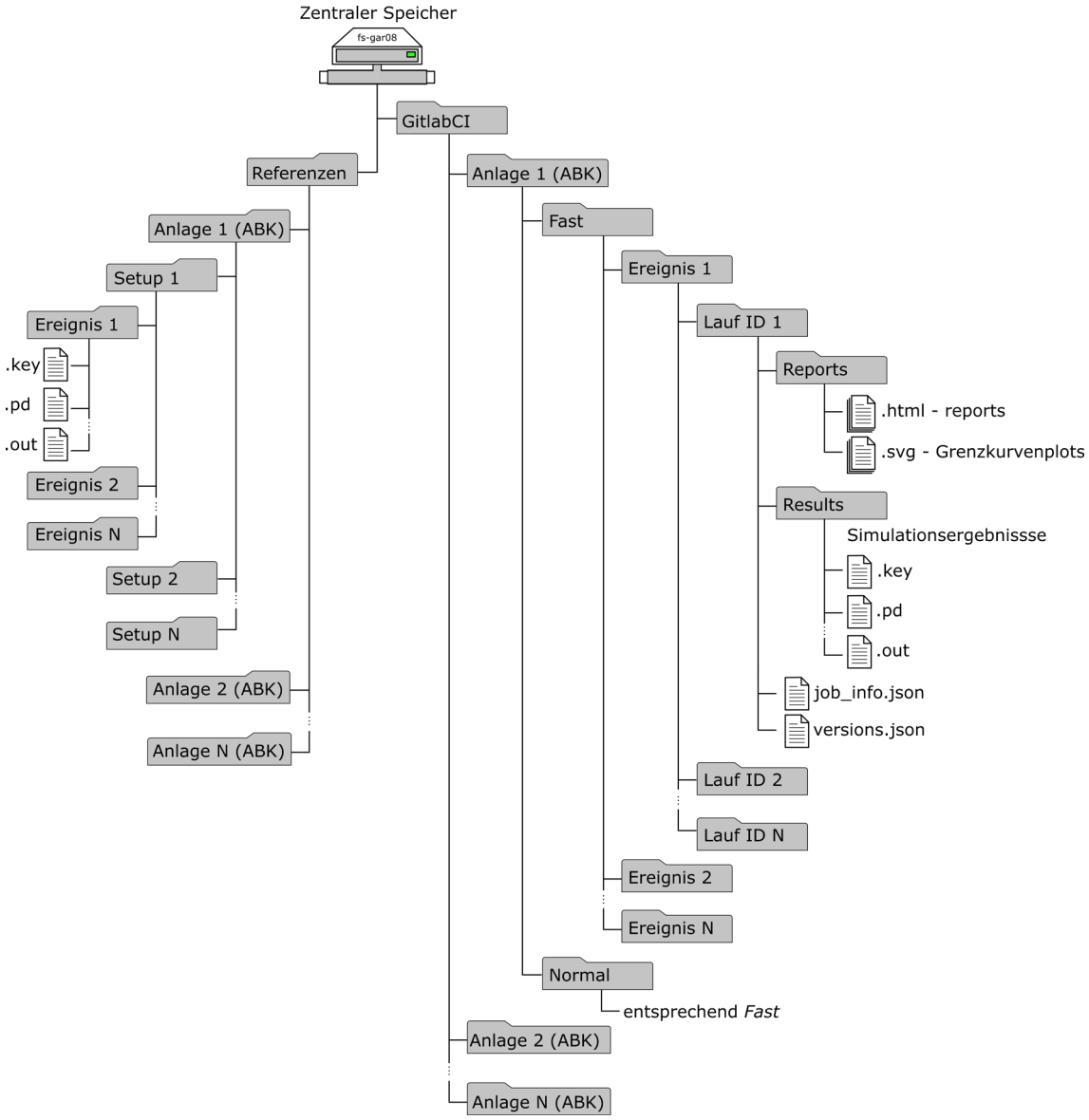

Abb. 4.3 Speicherstruktur zur Ablage der Verifizierungsergebnisse

4.2.2 Auslösung der Verifikation

Für den Start einer Verifikation steht eine Reihe von Optionen zur Verfügung. Grundsätzlich erfolgt eine vollständige Datensatzverifikation automatisch im monatlichen Intervall, sofern es seit dem letzten Verifikationslauf Änderungen im jeweiligen Datensatz-Repository gegeben hat. Die vollständige Verifikation überprüft die vollständigen Transientenverläufe aller Ereignisse der datensatzabhängigen Simulationsmatrix und dauert etwa 7 Tage, abhängig von der Größe der Simulationsmatrix und der Art der berücksichtigten Transienten.

Zusätzlich zur vollständigen Verifikation (Test „normal“), wurde ein Schnelltestverfahren implementiert (Test „fast“). Das Schnelltestverfahren wird automatisch bei jeder Änderung in einem Datensatz-Repository auf dem Gitlab-Server der GRS ausgelöst und prüft die Rechenfähigkeit der „gepushten“ Änderungen für alle Ereignisse der jeweiligen Simulationsmatrix für die ersten 0,1 s. Dieser Schnelltest ist in weniger als 1 h abgeschlossen und stellt sicher, dass die aktuelle Version des Datensatzes jederzeit rechenfähige ist, d. h. zum Beispiel keine Syntaxfehler im Eingabedatensatz oder den Protokolldateien vorhanden sind.

Neben der automatischen Auslösung der Verifikation kann ein Entwickler den Verifikationsprozess auch bei Bedarf jederzeit manuell starten. Der manuelle Start kann über eine Benutzeroberfläche der Gitlab-CI erfolgen. Bei einem manuellen Start kann die Variable `RUN_MODE` auf den Wert „normal“ gesetzt, so dass die Simulationsläufe mit der vollständigen Simulationszeit durchgeführt werden.

In der erstellten Eingabemaske in Gitlab-CI ist eine Reihe von weiteren Eingaben zur Steuerung der Verifikation einzugeben. Dies betrifft die jeweils zu verwendenden Versionen des Sourcecodes der zu bauenden Binarys, falls diese von den Standardeinstellungen (jeweils letzte Version) abweichen. Es kann dabei ein bestimmter Commit in dem jeweiligen Repository, ein Branch-Name, dessen aktueller Commit dann verwendet wird, sowie ein Tag-Name angegeben werden. Die Angabe zur Anzahl der zu verwendender Rechenressourcen bestimmt, wie viele Simulationen parallel durchgeführt werden (`NUM_RG`). Über die Variable `NPP_NAME` wird bestimmt, für welche(n) Reaktor/Reaktoren die Datensätze verifiziert werden. Eine Ausdehnung des Verfahrens auf die Datensätze weiterer Reaktoren ist dabei prinzipiell mit geringem Aufwand möglich. Hierfür ist die Nomenklaturkonvention[4] der Datensatz-Repositorys auf dem GRS-Gitlab-Server einzuhalten, um sie in die Verifizierung einzubeziehen. Zusätzliche Anpassungen an den YAML-Skripten der Verifikationsprozedur sind nicht erforderlich. Eine Auflistung der Voraussetzungen für die Einbindung eines neuen Analysesimulators in den Verifikationsprozess ist in Kapitel 4.2.1 gegeben.

[4] Nomenklaturkonvention für Datensatzrepositories auf dem GRS Gitlb-Server in AnalysisSimulators/dsa_explicit: `Analysesimulator_<ABK>` (`<ABK>`… Abkürzung der Anlage). Aufruf zur Verifikation über Angabe von `<ABK>`.

4.2.3 Erfahrung bei der Implementierung dynamisch generierter Pipelines auf Gitlab-CI

Die Möglichkeit, Pipelines dynamisch zu generieren und Daten zwischen hart kodierten und dynamisch generierter Pipelines auszutauschen, wird vom Hersteller von Gitlab als einsatzreife Fähigkeit der Serversoftware beworben. Im Rahmen dieses Projektes, das bisher das umfangreichste in dieser Beziehung innerhalb der GRS war, zeigte sich aber, dass es eine größere Menge, durch Bugreports beim Hersteller von Gitlab dokumentierte Probleme gibt. Diese sollen hier dokumentiert werden, um bei zukünftigen Erweiterungen oder ähnlich gelagerten Projekten nicht wieder zu einem unnötigen Aufwand zu führen.

Die Jobs laufen auf der Gitlab-CI auf Basis des Betriebssystems Windows und die Befehle in den Pipelinebeschreibungen werden vom PowerShell-Interpreter ausgeführt. In der PowerShell neu erzeugte Textdatei werden im UTF-Format gespeichert, wobei jede Datei mit einer so genannten Byte-Order-Markierung beginnt. Wird auf diese Weise eine Steuerungsdatei für eine Gitlab-CI-Pipeline erzeugt, so kann der Gitlab-Server diese nicht verarbeiten und bricht mit einer unklaren Fehlermeldung ab. Als Alternative wurde nur die Möglichkeit gefunden, die Datei im ASCII-Format abzuspeichern, was zu Problemen führen wird, falls zukünftig nicht-ASCII Zeichen wie Umlaute verwendet werden, z. B. für die Namen von Simulationsfällen.

Prinzipiell ist es möglich, aus einer übergeordneten Pipeline auf Jobs zu warten, die in einer „Kind“-Pipeline ablaufen. Es ist aber zum aktuellen Zeitpunkt immer noch ein dokumentiertes, bestehendes Problem, dass Artefakte aus einem Kind-Job nicht in den Eltern-Job übernommen werden können. Das zwang im aktuellen Projekt dazu, nicht nur die Jobs für die Simulationsläufe automatisch generiert in der Kind-Pipeline durchzuführen, sondern auch die Auswertung der Simulationsergebnisse. Dies erhöhte die Komplexität der dynamischen Jobgenerierung.

Auch konnte der automatische Mechanismus der Gitlab-CI zur Auswertung von Testergebnisse und deren Darstellung in der Benutzeroberfläche der Gitlab-CI nicht verwendet werden, da auch dieser Datenaustausch nicht zwischen Eltern- und Kind-Pipelines funktioniert.

Es wurden sporadische, nicht-reproduzierbare Jobabbrüche beobachtet, die im Zusammenhang mit der Git-Erweiterung für große Dateien (Git LFS) stehen, bzw. es traten

Fehlermeldungen auf, die auf Probleme mit dem Aufräumen der während der Jobs generierten Dateien hindeuteten. Beide Probleme sind seit längerer Zeit bekannt, es existieren aber nur Workarounds und keine Bugfixes.

Darüber hinaus wurden folgende Einschränkungen, die insbesondere bei einem Vergleich mit den Fähigkeiten der Jenkinssoftware festgestellt werden mussten, beobachtet. Diese sind teilweise seit mehreren Jahren dem Hersteller der Gitlab-Server-Software bekannt. So ist es momentan auf der Übersichtsseite der Pipelineläufe auf der Gitlab-Benutzeroberfläche nicht möglich, Pipelineläufe durch benutzerdefinierte Hinweise zu markieren. Damit könnte man z. B. einfach erkennen, ob ein Pipelinelauf nur im Schnelltestverfahren oder vollständig durchgeführt wurde. Auch führt ein Überschreiten des Timeouts eines Gitlab-Jobs dazu, dass dieser als fehlerhaft markiert wird und keine Artefakte gespeichert werden. Die davon abhängigen Jobs in einer nachgelagerten Stufe können dann gar nicht bzw. nicht sinnvoll ausgeführt werden. Es war deshalb notwendig, einen eigenen, zweiten Timeoutmechanismus zu implementieren. Wie oben beschrieben, war es notwendig, einen eigenen Warteschlangenmechanismus zu implementieren, da der in Gitlab vorhandene entweder nur die serielle Ausführung oder die parallele Ausführung auf allen vorhanden Rechnerknoten gleichzeitig erlaubt. Ebenfalls ein bekanntes Problem sind Einschränkung bei der Speicherung von vertraulichen Daten, wie z. B. dem Passwort für ein Netzlaufwerk. Auch scheint keine einfache, programmtechnische Möglichkeit zu bestehen, Artefakte mit der Git-Version zu verbinden, auf deren Basis sie erzeugt wurden. Im aktuellen Projekt wurden deshalb diese Daten separat in den Artefakten gespeichert.

Als umständlich erwies sich auch, dass Informationen über einen Git-Server für normale Nutzer nicht auf einfache Weise und zentral abrufbar sind. Dies betraf z. B. die auf den einzelnen Rechnerknoten eingestellten Jobtimeouts oder die momentane Verfügbarkeit und Auslastung der Knoten.

5 Protokollierung und Umsetzung der simulationsspezifischen Maßnahmen zur Durchführung der Rechnungen

5.1 Einführung in die Protokollierung von Steuerungsbefehlen bei Anwendung des ATHLET-Controllers

Für die Verifizierung eines Analysesimulator-Datensatzes ist die Erarbeitung einer geeigneten Simulationsmatrix notwendig, um die korrekte Nachbildung der betrieblichen Systeme und Komponenten, Begrenzungssysteme sowie der Sicherheitssysteme in ihren unterschiedlichen Zuständen aufzuzeigen. Im Rahmen des Vorhabens 4715R01345 /PAL 18/ wurde dazu eine sorgfältige Auswahl von Transienten bzw. Ereignissen aus der SE 2 bis 4a /SIA 15/ getroffen mit dem Ziel, einen Mindestsatz durchzuführender Ereignisanalysen zur Datensatzverifizierung zu definieren. Für einen Druckwasserreaktor (DWR) deutscher Bauart werden nach /PAL 18/ insgesamt 19 Simulationen für die wesentlichen Ereignisse der SE 2 bis 4a vorgeschlagen. Diese Ereignisliste zur Datensatzverifizierung wurde für die Umsetzung des automatisierten Verifizierungsverfahrens übernommen und wird in Tab. 5.1 mit der jeweiligen transientenspezifischen Zielsetzung vorgestellt.

Eine maschinenlesbare Protokollierung der transientenspezifischen Maßnahmen aller in Tab. 5.1 vorgestellten Simulationen wird in einer Spezifikationsdatei (gen. Ablaufprotokoll) umgesetzt, welche von dem in Kapitel 3 vorgestellten Steuerungsprogramm (ATHLET-Controller) eingelesen und verarbeitet werden kann. Diese Ablaufprotokolle basieren auf der Programmiersprache Python und verwenden die eigenentwickelte, ATHLET-spezifische und in Kapitel 3.2 beschriebene Syntax zur Festlegung der Steuerungsbefehle. Mit Hilfe der so erstellten Spezifikationsdateien ist eine automatische Durchführung und detaillierte Steuerung der ATHLET-Simulationen zur Verifikation möglich, welche auch in der Lage ist, erweiterte Steuerungsszenarien, wie z. B. regelmäßig wiederkehrende oder zustandsabhängige Handmaßnahmen, abzuarbeiten. Weiter können durch Anwendung des ATHLET-Controllers transientenspezifischen Systemzustände, Randbedingungen und/oder auslösende Ereignisse kontrolliert werden. Diese werden ebenfalls innerhalb der Ablaufprotokolle festgehalten.

Die Ablaufprotokolle folgen in ihrem Aufbau einem standardisierten Schema mit einheitlichen Konventionen in Inhalt und Aufbau. Dazu wird ein Protokoll in drei Bereiche unterteilt:

- Variablendefinition:

  Es erfolgt die Zuweisung von für die Simulationssteuerung relevanten Parametern aus ATHLET zu Steuerungsvariablen des Python-basierten ATHLET-Controllers. Hierbei wird weiter unterschieden in

  - Trigger-Variablen: Parameter, welche während der Laufzeit ausgelesen werden und auf Basis derer Veränderung Steuerungsaktionen auszulösen sind,
  - Info-Variablen: informative Parameter, anhand deren Verlauf in der Konsolenausgabe die grundlegende Funktionalität des Ereignisverlaufs geprüft werden kann,
  - Target-Variablen: Parameter welche auf Basis der Veränderungen von Trigger-Variablen und deren logischer Verknüpfung Steuerungseingriffe innerhalb des Simulationsprozesses erfolgen.

- Logikdefinition:
  Es erfolgt die Definition logischer Bedingungen in Form von Wertabfragen von Trigger-Variablen und deren logischer Verknüpfung auf Grundlage boolescher Algebra.

- Action-Definition:
  Es werden Steuerungsaktionen für Zielparameter (Target-Variablen) festgelegt, welche bei der Erfüllung der zugeordneten Bedingungen aus dem Bereich „Logikdefinition“ ausgelöst werden. Diese Aktionen können innerhalb des Protokolls selbst definiert werden oder es kann auf die in Kapitel 3.2 vordefinierten Aktionen zurückgegriffen werden.

Ein Beispiel für die Anwendung der ATHLET-bezogenen Konvention bei der Erstellung eines Ablaufprotokolls ist in List. 3.2 in Kapitel 3.2 dargestellt.

Tab. 5.1 Ereignisliste für die Verifizierung eines Vorkonvoi-Analysesimulators nach /PAL 18/

#	Ereignisse	Kurzbezeichnung	Ziel der Rechnung	Typ
1	Rechnung eines stationären Zustandes (Vollastbetrieb)	Volllast	Überprüfung des Gleichgewichts der Energie- und Massenbilanzen zwischen Primär und Sekundärseite anhand der anlagenspezifischen thermohydraulischen Parameter.	-
2	Abfahren der Anlage bis auf den Zustand „Unterkritisch kalt“	Abfahren_UH_UK	Verifizierung des dynamischen Verhaltens der Anlage anhand des stationären Teillastdia-gramms und Prüfung der Funktionalität der Leistungsregelung und der betrieblichen Aufgabe der aktivierten Anlagensysteme (z. B. nukleares Nachkühlsystem).	-
3	Fehlöffnen eines DH-Sprühventils	D2-12_DH_SpVentil	Prüfung der Implementierung der KMD-Sollwertbegrenzung und der Funktionalität des Druckhaltersystems.	SE2 (D2-12)
4/5	Fehlöffnen der HD-Reduzierstation bzw. Fehlschließen der GBA-Armatur	D2-14_HD_Reduzier_AUF D2-16_HD_Reduzier_ZU	Prüfung der Implementierung der KMD- und KMM-Sollwertbegrenzung und der Funktionalität des Volumenregelsystems im Fall einer Massenbilanzstörung im Primärkreislauf. Prüfung der implementierten LOOP-RELEB-Grenzwerte und der Druckhalter-Abblaseregelung.	SE2 (D2-16/ D2-14)
6	Störung in der KMT-Regelung, die zu einem unkontrollierten Ausfahren von Steuerstäben führt	D2-21_KMT	Prüfung des Anlageverhaltens infolge einer unkontrollierten Leistungszufuhr im Leistungsbetrieb (Reaktivitätsstörung). Verifizierung der Implementierung der L-RELEB-Grenzwerte.	SE2 (D2-21)
7	Fehlerhafte Inbetriebnahme eines Strangs des Chemikalieneinspeisesystems, die zur Einspeisung von Deionat in PKL führt (externe Deborierung)	D2-24_Deionat	Prüfung des Anlagenverhaltens infolge einer Änderung der Reaktivität. Verifizierung der Implementierung der Stabfahrbegrenzung (STAFAB), Fahrgeschwindigkeitsbegrenzung (FAGEB) sowie der D-Bank-Stellungsregelung (D-BARE). Überprüfung der Steuerungsbefehle für die Betätigung der Komponenten in Chemikalieneinspeisesystem und Volumenregelsystem.	SE2 (D2-24)
8	Fehlöffnen eines FD-Sicherheitsventils	D2-01_FD_SiVentil	Prüfung des Anlagenverhaltens infolge einer Fehlfunktion im Frischdampf-System, die zu einer ungeplanten Temperatur-/Druckabsenkung im DE bzw. im PKL führt. Verifizierung der Frischdampfdruckregelung	SE2 (D2-01)

#	Ereignisse	Kurzbezeichnung	Ziel der Rechnung	Typ
			sowie der implementierten Funktionen für die Absperrung der betroffenen Dampferzeuger in Reaktorschutzsystem.	
9	Lastabwurf auf Eigenbedarf	D2-07_LAW-MAN	Prüfung des Anlagenverhaltens infolge einer schnellen Leistungsabsenkung. Verifizierung der Implementierung des Kriteriums für Lastabwurf (LAW) von der Stabeinwurffunktion STEW und der Korrektheit der Nachbildung der Dampferzeuger-Füllstandregelung (Vollast- und Schwachlastregelung). Prüfung der implementierten Umleitstationsregelung.	SE2 (D2-07)
10	Ausfall aller in Betrieb befindlichen Hauptspeisewasserpumpen mit Zuschaltung der Reservepumpe	D2-09_HspW_Pumpen	Prüfung des Anlagenverhaltens infolge eines vollständigen bzw. partiellen Ausfalls der Speisewasserversorgung. Verifizierung der Begrenzungseinrichtungen für die Beherrschung des Störfalls (SPEISE-RELEB und STEW) sowie der Implementierung sekundärseitiger RESA-Anregekriterien im Reaktorschutzsystem.	SE2 (D2-09)
11/ 12	Ausfall der Hauptwärmesenke (TUSA ohne FDU) / Fehlöffnen aller FDU	D2-02-06_HWSenke	Prüfung der Funktionalität der implementierten FD-Abblaseregelung nach Auslösung des Teilabfahrens. Verifizierung der korrekten Nachbildung der FD-Armaturenstation und der Ansprechwerte der FD-Sicherheitsventile.	SE2 (D2-06/ D2-02)
13	Ausfall einer Hauptkühlmittelpumpe (PUMA 1v4)	D2-10_PUMA1v4	Prüfung des Anlageverhaltens beim Teilloop-Betrieb infolge eines Ausfalls einer Hauptkühlmittelpumpe (PUMA). Verifizierung der Implementierung des PUMA-RELEB Grenzwerts und der nachfolgenden STEW-PUMA-Funktion für die schnelle Reduktion der Reaktorleistung. Überprüfung des Störfallerkennungssignals aus MADTEB für die Aktivierung störfallspezifischer Fahrbereich und Maßnahmen für die KMD- und KMM-Begrenzung.	SE2 (D2-10)
14	Notstromfall gleich oder kürzer als 10 Stunden	D2-28_Notstrom	Prüfung des Anlageverhaltens infolge eines Ausfalls der Eigenbedarfsversorgung. Verifizierung des implementierten Modells für die Inbetriebnahme der Notstromversorgung durch Diesel-Start und Belastungsprogramm in Reaktorschutzsystem.	SE2 (D2-28)

#	Ereignisse	Kurzbezeichnung	Ziel der Rechnung	Typ
15	Versagen eines Dampferzeuger-Heizrohres (größer als betrieblich zulässige Leckagen und bis maximal 2F)	D3-31_DE_Heizrohr	Verifizierung der implementierten Einsatzlogik des KMD- und KMM-Befehlsdiagramms „DE-Heizrohrleck“ für die Leistung- und Druckabsenkungsphase. Überprüfung des Zusammenspiels diverser betrieblichen und Sicherheitssysteme für die Beherrschung des Störfalles.	SE3 (D3-31)
16	Mittleres Leck innerhalb des Sicherheitsbehälters (Leckquerschnitt ≤ 0,1F)	D3-23_KMV_01F	Überprüfung der implementierten RESA-Anregekriterien und Notkühlkriterien im Reaktorschutzsystem für die Störfallerkennung. Prüfung der im Datensatz implementierten Prozesssignale für die Betrachtung der Übertragung von Energie und Masse aus der Primärseite im Sicherheitsbehälter. Verifizierung der Funktionalität der implementierten Komponenten des Not- und Nachkühlsystems. Prüfung der Einsatzlogik der KMD-Befehlsdiagramme „Kühlmittelverlust“.	SE3 (D3-23)
17	Fehlöffnen und Offenbleiben eines Druckhalter-Sicherheitsventils	D3-30_DH_SiVentil	Prüfung der korrekten Nachbildung von DH-Sicherheitsventilen und des Abblasebehälters (ABB). Verifizierung des implementierten Modells für die ABB-Kühler. Prüfung der Auslösung der Notkühlkriterien	SE3 (D3-30)
18	Leck in Frischdampfsystem innerhalb des Sicherheitsbehälters	D3-05_FD_Leck	Prüfung des Anlageverhaltens infolge eines Bruchs einer Frischdampfleitung hinter der Frischdampfabschlussarmatur (FD-AA). Verifizierung der Implementierung des Druckabfall-Kriteriums (DAF) und des Teilabfahren-Signals im Reaktorschutzsystem	SE3 (D3-05)
19	Ausfall der Hauptspeisewasserversorgung und mechanisches Verklemmen aller Steuerstäbe (ATWS)	D4a-04_ATWS	Verifizierung des implementierten RESA-Kontrollsignals zur Erkennung von ATWS-Störfällen sowie der Einsatzlogik des KMD- und KMM-Befehlsdiagramms „ATWS“. Überprüfung des Zusammenspiels diverser betrieblichen und Sicherheitssysteme für die Beherrschung des Störfalles.	SE4a (D4a-04)

5.2 Beschreibung erstellter Protokolle für die Ereignisse aus der Simulationsmatrix

Im Folgenden werden die in den Ablaufprotokollen festgelegten Steuerungsbefehle für alle in die Verifizierung des Vorkonvoi-Datensatzes einbezogenen Simulationen aufgeführt. Teilweise stimmen die Darstellungen außerdem mit den Anwendungen zur Übertragung des Verifikationsverfahrens auf den Konvoi-Datensatz überein. Entsprechende Verweise sind in Kapitel 8.3 angeführt.

Nummerierung und Kurzbezeichnungen der Simulationsfälle beziehen sich im Folgenden auf die Angaben in Tab. 5.1. Die Kurzbezeichnungen entsprechen außerdem der verwendeten Nomenklatur für die berücksichtigten Ereignisse sowohl in der Speicherstruktur (siehe in Kapitel 4.2, Abb. 4.3) als auch für die erstellten Ablaufprotokolle. Ausführliche Erläuterungen zu den Simulationsergebnissen sowie ihrer Verifizierung finden sich in /PAL 18/.

5.2.1 Ereignis #1 - Volllast

Die Berechnung eines stationären Zustandes (Vollastbetrieb) erfolgt für 7.000 s simulierte Laufzeit und dient der Überprüfung des Gleichgewichts der Energie- und Massenbilanzen zwischen Primär und Sekundärseite. Im Ablaufprotokoll wird diese Laufzeit und somit der Abbruchzeitpunkt festgelegt. In Tab. 5.2 und Tab. 5.3 sind dazu notwendigen und im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) aufgelistet. Der stationäre Endzustand der Simulation stellt den Startzeitpunkt der nachfolgenden transienten Simulationen aus der Ereignisliste (Ereignisse #2 bis #19 in Tab. 5.1) dar. Diese werden als *Restart* auf das Ergebnis der Volllastrechnung aufgesetzt und verwenden dieses als Initialbedingung. Dadurch wird ein stabiler Ausgangszustand für die Transienten sichergestellt und gleichzeitig für die 18 transienten Ereignisse aus der Simulationsmatrix die simulierte Echtzeit um eine Mindesteinschwingzeit von 600 s verkürzt, was einer Zeitersparnis für die Durchführung einer Simulation von je mindestens 1 h entspricht.

Tab. 5.2 Datensatzeingriffe (Actions) im Fall Volllast

Maßnahme / Ereignis	Bedingung	Action	Parameter
Ende der Simulation	TEND	terminate	-

Tab. 5.3 Kontrollbedingungen (Trigger) im Fall Volllast

Bedingung	Beschreibung	Logik	Wert	Parameter
TEND	simulierte Zeit [s]	>=	7.000	-

5.2.2 Ereignis #2 - Abfahren_UH_UK

Die Simulation des vollständigen Abfahrvorgangs der Anlage aus Volllast über den Zustand „unterkritisch heiß“ auf „unterkritisch kalt“ ermöglicht eine umfassende Betrachtung und Verifizierung einer Vielzahl betrieblicher Anlagensysteme. Im Sinne einer Verifizierung des Verhaltens der dabei beteiligten Systeme ist der Abfahrvorgang darüber hinaus auch von besonderem Wert, da umfangreiche Anlagendaten und Angaben aus der Anlagendokumentation vorliegen.

Zum Abfahren wird die Anlage dazu zunächst aus dem Leistungsbetrieb entsprechend der Beschreibungen in Betriebshandbüchern in den Zustand „unterkritisch heiß“ überführt:

- Hilfssysteme in Betrieb nehmen
- Abfahren auf Mindestlastpunkt
- Abfahren auf 0 % Reaktorleistung
- Hauptspeisewasser-Förderung und Kondensatsysteme abfahren

Anschließend wird die Anlage aus dem so erreichten Zustand „unterkritisch heiß“ in den Zustand „unterkritisch kalt“ überführt. Dieser Vorgang wird in folgende Schritte unterteilt:

- Fahrweise der Borkonzentration anpassen
- Abfahren mit 50 K/h über FDU
- KMT-Absenkung auf 120 °C
- Übernahme der Wärmeabfuhr durch das Nachkühlsystem

5.2.3 Ereignis #3 - D2-12_DH_SpVentil

Anhand der Simulation des Ereignisses lässt sich die Güte der Umsetzung des Druckhalter- und Abblasesystems (YP) im Reaktorkühlsystem sowie der Kühlmittel-Massen-, Druck- und Temperaturgradienten-Begrenzung (MADTEB) im Analysesimulator prüfen.

Des Weiteren kann die Funktionsweisen der Schnellabschaltsysteme mit der Reaktor- und Turbinenschnellabschaltung geprüft werden.

Zur Umsetzung der automatisierten Steuerung des Transientenablaufs wird aus dem stabilen Volllastzustand heraus das Fehlöffnen und Offenbleiben eines Sprühventils im Druckhalter forciert. Die dazu notwendigen und im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.4 und Tab. 5.5 aufgelistet.

Tab. 5.4 Datensatzeingriffe (Actions) im Fall D3-30_DH_SiVentil

Maßnahme / Ereignis	Bedingung	Action	Parameter
Fehlerhaftes Verhalten eines Sprühventils im Druckhalter	T0	up	-
Ende der Simulation	TEND	terminate	-

Tab. 5.5 Kontrollbedingungen (Trigger) im Fall D3-30_DH_SiVentil

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	10.500	-

5.2.4 Ereignis #4 - D2-14_HD_Reduzier_AUF

Die Simulation dieser Transiente eignet sich zur Prüfung der Funktionsweise des Volumenregelsystemmodells im Analysesimulator im Fall einer Massenbilanzstörung sowie für die Prüfung der implementierten MADTEB-Befehlsdiagramme für die Erkennung anormaler Betriebssituationen sowie von Störfällen. Die innerhalb der Transienter angeforderten MADTEB-Maßnahmen betreffen fast ausschließlich die KMD- und KMM-Begrenzungen.

In der für die Verifizierung herangezogenen Transiente wird das Fehlfahren bzw. fehlerhafte Verharren von Armaturen in ihrer Ausgangsposition aufgrund leittechnischer Fehler unterstellt. Etwaige Fehler im implementierten Verhalten des TA-Systems können hierbei durch Nichteinhalten des DH-Füllstandsollwertes im Leistungsbetrieb erkannt werden. Außerdem wird im Verlauf der Transiente bei der Entnahme bzw. Einspeisung mit einem Ansprechen von Temperaturverriegelungen und einem Ausfall des gesamten TA-Systems gerechnet. Die zur Steuerung der Transiente im Ablaufprotokoll fest-

gehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.6 und Tab. 5.7 aufgelistet.

Tab. 5.6 Datensatzeingriffe (Actions) im Fall D2-14_HD_Reduzier_AUF

Maßnahme / Ereignis	Bedingung	Action	Parameter
Fehlerhaftes Öffnen der HD-Reduzierstation	T0	jump	1
Ende der Simulation	TEND	terminate	-

Tab. 5.7 Kontrollbedingungen (Trigger) im Fall D2-14_HD_Reduzier_AUF

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	11.000	-

5.2.5 Ereignis #5 - D2-16_HD_Reduzier_ZU

Die Durchführung dieser Transiente eignet sich ebenso wie das in Kapitel 5.2.4 beschriebene „Fehlöffnen der HD-Reduzierstation“ für die Prüfung der Funktionalität des Volumenregelsystems im Fall einer Massenbilanzstörung sowie für die Prüfung der Implementierung von MADTEB-Befehlsdiagrammen für die Erkennung verschiedener Betriebssituationen und Störfälle. Bei dem hier vorgestellten „Fehlschließen einer GBA-Armatur in der Entnahmeleitung des Volumenregelsystems“ wird zusätzlich die Nachbildung der LOOP-RELEB Grenzwerte geprüft. Die LOOP-RELEB Grenzwerte dienen in Ergänzung zu der KMT-RELEB zur Begrenzung des Primärkreisenergieinhaltes bei schnellen Leistungstransienten. Die hierbei unterstellte Störung wird von Armaturen verursacht, die aufgrund leittechnischer Fehler fehlfahren bzw. fehlerhaft in ihrer Position verharren. Dabei wird eine Störung in der Entnahme durch das Volumenregelsystem unterstellt.

Die zur Steuerung der Transiente im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.8 und Tab. 5.9 zusammengefasst.

Tab. 5.8 Datensatzeingriffe (Actions) im Fall D2-16_HD_Reduzier_ZU

Maßnahme / Ereignis	Bedingung	Action	Parameter
Blockierung der HD-Reduzierstation 1	T0	jump	-1
Schließen der GBA-Armatur	T0	jump	-1
Vorsteuerung der GBA-Armatur	T0	jump	-1
Ende der Simulation	TEND	terminate	-

Tab. 5.9 Kontrollbedingungen (Trigger) im Fall D2-16_HD_Reduzier_ZU

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	8.500	-

5.2.6 Ereignis #6 - D2-21_KMT

Die Simulation dieses Ereignisses dient der Verifizierung des Anlagenverhaltens infolge einer Störung in der Kühlmitteltemperatur-(KMT)-regelung. Postuliert wird ein sprunghafter Anstieg des KMT-Sollwertes um 9 K was zu einem unbeabsichtigten Ausfahren der Steuerstäbe führt. Folgende Funktionen des Analysesimulator werden anhand der Simulation dieser Transiente verifiziert:

- L-Bank-Stellungsregelung (L-STABS);
- Koordination der L- und D-Bank Regelbefehle;
- Begrenzung der Reaktorleistung (L-RELEB).

Tab. 5.10 und Tab. 5.11 listen die im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) zur Simulationssteuerung auf.

Tab. 5.10 Datensatzeingriffe (Actions) im Fall D2-21_KMT

Maßnahme / Ereignis	Bedingung	Action	Parameter
Änderung KMT-Sollwert	T0	jump	317
Ende der Simulation	TEND	terminate	-

Tab. 5.11 Kontrollbedingungen (Trigger) im Fall D2-21_KMT

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	8.000	-

5.2.7 Ereignis #7 - D2-24_Deionat

Simuliert wird eine fehlerhafte Inbetriebnahme eines Stranges des Chemikalieneinspeisesystems, was zu einer Einspeisung von Deionat in den PKL und somit zu einer externen Deborierung führt. Anhand dieses Ereignisses wird das Anlagenverhaltens infolge einer Änderung der Reaktivität im Kern verifiziert. Für die Rechnung wird angenommen, dass einer der zwei Stränge des Chemikalieneinspeisesystems fehlerhaft in Betrieb genommen wird und Deionat über die Zuleitungen des Volumenregelsystems in den Primärkreislauf eingespeist wird. Die D-Bank-Regeleinrichtung (D-BARE) soll in diesem Szenario gegen eine Deborierung des Kühlmittels ansteuern, um eine ausreichende Abschaltreaktivität sicherzustellen.

Folgende Funktionen des Analysesimulator werden anhand der Simulation dieser Transiente verifiziert:

- Steuerstabfahrbegrenzung (STAFAB);
- D-Bank-Regeleinrichtung (D-BARE);
- Begrenzung bei zu hohem Energieinhalt im Primärkreis (LOOP-RELEB);
- Komponenten des Volumenregel- (TA-System) und Chemikalieneinspeisesystems (TB-System).

Die zur Steuerung der Transiente im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.12 und Tab. 5.13 aufgelistet.

Tab. 5.12 Datensatzeingriffe (Actions) im Fall D2-24_Deionat

Maßnahme / Ereignis	Bedingung	Action	Parameter
Fehlerhaftes Einschalten Deionatpumpe	T0	jump	1
Öffnen Deionateinspeiseventil	T0	jump	1
Teilsteuerung aller Pumpen TA aus	T0&T1	down	-
Anforderung der 2. HD-Förderpumpe	T0&T1	up	-
Ende der Simulation	TEND	terminate	-

Tab. 5.13 Kontrollbedingungen (Trigger) im Fall D2-24_Deionat

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
T1	Massenstrom Deionatpumpe [kg/s]	>	8	-
TEND	simulierte Zeit [s]	>	8.900	-

5.2.8 Ereignis #8 - D2-01_FD_SiVentil

Bei dieser Transiente wird ein fehlerhaftes Auffahren des FD-Sicherheitsventils in der FD-Station im Dampferzeuger angenommen. Hauptziel dieser Rechnung ist die Verifizierung des im Reaktorschutzsystem gebildeten Signals für die Absperrung der Absperrarmaturen vor den FD-Sicherheitsventilen. Die Steuerung der Transiente mit Hilfe des erstellten Ablaufprotokolls zum Festgehalten der Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) erfolgt über die in Tab. 5.14 und Tab. 5.15 dargestellte Signal-Logik.

Tab. 5.14 Datensatzeingriffe (Actions) im Fall D2-01_FD_SiVentil

Maßnahme / Ereignis	Bedingung	Action	Parameter
Fehlerhaftes Öffnen des FD-Sicherheitsventils	T0	up	-
Ende der Simulation	TEND	terminate	-

Tab. 5.15 Kontrollbedingungen (Trigger) im Fall D2-01_FD_SiVentil

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	8.200	-

5.2.9 Ereignis #9 - D2-07_LAW-MAN

Die Simulation dieser Transiente dient zur Verifizierung des Anlagenverhaltens bei Störungen in der Leistungsabgabe mit Lastabwurf auf Eigenbedarf (LAW-EB). Die Turbine wird über ihre Stellventile innerhalb von 2 s auf ca. 5 % der Nennleistung der Anlage abgefahren und das Kraftwerk in den Inselbetrieb überführt. Ziel der Gegenmaßnahmen bei LAW ist es, den Reaktor bei niedriger Leistung zu stabilisieren ohne Reaktorschutzgrenzwerte zu verletzen.

Anhand der Transiente wird die Stabeinwurffunktion (STEW) sowie die implementierte Umleitstationsregelung geprüft. Zusätzlich wird die korrekte Nachbildung der Dampferzeuger-Füllstandregelung (Vollast- und Schwachlastregelung) verifiziert. Letztere hat die Aufgabe, den Füllstand im Dampferzeuger (DE) unabhängig von Lastschwankungen und innerhalb fester Grenzen auf dem vorgegebenen Sollwert zu halten. Bei folgenden temporären Anlagenzuständen ist dieser Sollwert um bestimmte Beträge abzusenken und teilweise zeitabhängig wieder anzugleichen:

- Einschalten bzw. Zuschalten einer Hauptkühlmittelpumpe
- DE-Abschluss bei erkanntem Heizrohrbruch
- Schneller Leistungsreduzierung

Die zur Steuerung der Transiente im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.16 und Tab. 5.17 aufgelistet.

Tab. 5.16 Datensatzeingriffe (Actions) im Fall D2-07_LAW-MAN

Maßnahme / Ereignis	Bedingung	Action	Parameter
Steuerungsautomatik LAW-MAN	T0	jump	1
Ende der Simulation	TEND	terminate	-

Tab. 5.17 Kontrollbedingungen (Trigger) im Fall D2-07_LAW-MAN

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	7.600	-

5.2.10 Ereignis #10 - D2-09_HspW_Pumpen

Mit dieser Transiente wird der Ausfall aller in Betrieb befindlichen Hauptspeisewasserpumpen mit Zuschaltung der Reservepumpe simuliert. Die Nachrechnung eignet sich für die Verifizierung der im Simulator implementierten Begrenzungsfunktionen (z. B. SPEISE-RELEB) sowie der Funktionen einzelner Komponenten, wie der Gruppen- und Untergruppensteuerung der Speisewasserpumpen. Gleichzeitig ermöglicht sie die Prüfung der thermodynamischen Wechselwirkung zwischen Primär- und Sekundärseite.

Im Rahmen der Transiente lässt sich sowohl die Nachbildung der Reaktorleistungsbegrenzungssignale (RELEB), die den Speisewasserdurchsatz und die Reaktorleistung

überwachen, als auch die Berechnung der erlaubten Reaktor- und Generatorleistung prüfen. Des Weiteren wird das Auslösen automatischer Maßnahmen und Regeleingriffe beim Ansprechen bestimmter Begrenzungswerte getestet.

Zur Initialisierung der Transiente werden die zwei in Betrieb befindlichen Hauptspeisewasserpumpen aus dem stabilen Volllastbetrieb über den Komponentenschutz (KS) abgeschaltet. Die dazu notwendigen und im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.18 und Tab. 5.19 aufgelistet.

Tab. 5.18 Datensatzeingriffe (Actions) im Fall D2-09_HspW_Pumpen

Maßnahme / Ereignis	Bedingung	Action	Parameter
Abschalten HspW-Pumpe 1	T0	up	-
Abschalten HspW-Pumpe 2	T0	up	-
Ende der Simulation	TEND	terminate	-

Tab. 5.19 Kontrollbedingungen (Trigger) im Fall D2-09_HspW_Pumpen

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	8.000	-

5.2.11 Ereignis #11/12 - D2-02-06_HWSenke

Mit dieser Transiente wird ein Ausfall der Hauptwärmesenke (Turbinenschnellabschaltung bei verblockter Frischdampfumleitstation; TUSA ohne FDU) simuliert. Die Nachrechnung dieses Ereignisses eignet sich für die Verifizierung der der im Datensatz implementierten Begrenzungsfunktionen (z. B. STEW-RELEB, L-RELEB) sowie der Funktionen einzelner Komponenten der sekundärseitigen Wärmeabfuhr, wie der Steuerung der FD-Abblaseregelventile, der FD-Armaturenstation und der Ansprechwerte der FD-Sicherheitsventile. Für die Simulation der Transiente wurden Unverfügbarkeiten der Reaktorleistungsbegrenzung entsprechend der Vorgabe der Verifizierungsbasis (Referenzfall Ausfall HWS für eine Vorkonvoi-Anlage /GRS 06/) angenommen. Die zur Steuerung der Transiente im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.20 und Tab. 5.21 aufgelistet.

Tab. 5.20 Datensatzeingriffe (Actions) im Fall D2-02-06_HWSenke

Maßnahme / Ereignis	Bedingung	Action	Parameter
unverf. L-RELEB 09-18: wenn R.-Leist. < PERL-2 % bis + 8 %	T0	down	-
unverf. STEW-RELEB-Folgebetrieb	T0	down	-
unverf. Folge-Einwurf synchron	T0	down	-
unverf. Folge-Einwurf 1 - 7	T0	down	-
unverf. STEW-Folgebetrieb	T0	down	-
unverf. STEW-Zentralstab	T0	down	-
unverf Synchroneinwurf Steuerstäbe	T0	down	-
Auslösung TUSA	T0	up	-
FDU unverfügbar	T0	down	-
Ende der Simulation	TEND	terminate	-

Tab. 5.21 Kontrollbedingungen (Trigger) im Fall D2-02-06_HWSenke

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	7.400	-

5.2.12 Ereignis #13 - D2-10_PUMA1v4

Ziel der Simulation dieses Ereignisses ist die Verifizierung des Verhaltens des Analysesimulators bei Ausfall einer Hauptkühlmittelpumpe. Hierbei erfolgt die Prüfung der Implementierung folgender Funktionen im Datensatz:

- Begrenzung der Reaktorleistung (PUMA-RELEB);
- Begrenzung der Generatorleistung (PERG);
- DE-Füllstandregelung (Umschaltung auf Schwachlastregelung).

Die zur Steuerung der Transiente im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.22 und Tab. 5.23 aufgelistet.

Tab. 5.22 Datensatzeingriffe (Actions) im Fall D2-10_PUMA1v4

Maßnahme / Ereignis	Bedingung	Action	Parameter
Ausfall Hauptkühlmittelpumpe 1v4	T0	up	-
Ende der Simulation	TEND	terminate	-

Tab. 5.23 Kontrollbedingungen (Trigger) im Fall D2-10_PUMA1v4

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	8.000	-

5.2.13 Ereignis #14 - D2-28_Notstrom

Mit diesem Ereignis wird ein vollständiger Verlust der Eigenbedarfsversorgung über eine Dauer von weniger als 10 h und ein Fehlschlagen der automatischen Umschaltung auf Eigenbedarf simuliert. Die Durchführung dieser Simulation erlaubt die Prüfung der richtigen Implementierung folgender Funktionen und Systeme im Analysesimulator:

- Aufbau der Stromversorgungsnetz- sowie der Notstromversorgung;
- Ablauf des Notstromprogramms;
- Bildung des Störfallerkennungssignals „PUMA 4v4“;
- Bildung der Einsatzlogik „Ausfall aller HKMP“ für die Durchführung der Maßnahmen anhand der Kennlinie aus MADTEB.

Zur Beherrschung dieses Ereignisses werden vom Reaktorschutzsystem neben der Auslösung von RESA und TUSA die Notstromdiesel gestartet, mit welchen nur noch die Komponenten versorgt werden, die zu einem sicheren Abfahren der Anlage notwendig sind. Tab. 5.24 und Tab. 5.25 listen die im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) zur Steuerung dieser Simulation auf.

Tab. 5.24 Datensatzeingriffe (Actions) im Fall D2-28_Notstrom

Maßnahme / Ereignis	Bedingung	Action	Parameter
Diesel unverfügbar	T0	down	-
RESA-Anregekriterium 1 rücksetzen	T0	down	-
RESA-Anregekriterium 2 setzen	T0	up	-
Vorwahl TA-Pumpe	T0	up	-
Notstromfall auslösen	T1	up	-
Ende der Simulation	TEND	terminate	-

Tab. 5.25 Kontrollbedingungen (Trigger) im Fall D2-28_Notstrom

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
T1	simulierte Zeit [s]	>	7.100	-
TEND	simulierte Zeit [s]	>	8.000	-

5.2.14 Ereignis #15 - D3-31_DE_Heizrohr

Das doppelendige Versagen eines Dampferzeuger-Heizrohres (2F-Bruch) zeichnet sich in seinem Störfallablauf durch besondere Komplexität und hohe Anforderungen an das Zusammenwirken von betrieblichen und Sicherheitssystemen aus. Beim Auftreten eines Lecks in einem Dampferzeuger-Heizrohr kommt es aufgrund des hohen Druckunterschiedes zwischen Primär- und Sekundärkreislauf zu einem Übertritt von Radioaktivität in den Wasser-Dampfkreislauf. Die wesentliche Aufgabe der MADTEB ist es dabei, die Störfallauswirkungen derart zu begrenzen, dass kein radioaktiver Dampf über das Dach an die Umgebung abgegeben und der Kühlmittelverlust im Primärkreislauf minimiert wird. Hierzu soll die Reaktor- und Generatorleistung zunächst so schnell wie möglich abgesenkt werden. Ein gleichzeitiger Netzausfall ist dabei so weit als möglich zu vermeiden, da der damit einhergehende Notstromfall eine zusätzliche Schwächung der Resilienz der Anlage mit sich bringt. Durch einen Weiterbetrieb der HKMP wird ein Zwangsumlauf im Primärsystem aufrechterhalten, um die Ausbildung einer RDB-Deckelblase bei sinkendem Kühlmitteldruck zu verhindern. Mit dem Kondensator als Wärmesenke wird das Abblasen radioaktiven Frischdampfes über das Dach vermieden. Nach einer Leistungsabsenkung wird der Kühlmitteldruck und damit die Druckdifferenz zwischen Primär- und Sekundärseite abgebaut, um die Leckrate zu reduzieren.

Der Störfallablauf bei DEHEIRO lässt sich grundsätzlich in drei Phasen gliedern:

- **Phase 1: Leistungsabsenkung**
  Über RELEB- und MADTEB-Maßnahmen werden Reaktor- und Generatorleistung abgesenkt.

- **Phase 2: KMD-Absenkung**
  Über MADTEB-Maßnahmen wird der KMD gezielt abgesenkt.

- **Phase 3: Abfahren in den Nachkühlbetrieb**
  Über Hand-Maßnahmen werden Kühlmitteltemperatur und -druck so weit abgesenkt, dass das Nachkühlsystem in Betrieb genommen werden kann; in dieser Phase

werden KMD und DH-Füllstand von der MADTEB in den störfallspezifisch definierten Fahrbereichen gehalten.

Die Durchführung dieses Störfalls eignet sich für die Prüfung des Zusammenspiels von betrieblichen Systemen und von Sicherheitssystemen, die relevant für die primärseitige Beherrschung des Dampferzeuger-Heizrohrlecks sind. Die Funktionsweise der Implementierung folgender Systeme im Datensatz wird dabei geprüft:

- das vierfach redundante Zusatzboriersystem (TW);
- das betriebliche Boriersystem (TB);
- das Volumenregelsystem (TA);
- MADTEB, RELEB und Reaktorschutzsystem;
- Druckhaltersystem (YP).

Die zur Steuerung der Transiente im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.26 und Tab. 5.27 aufgelistet.

Tab. 5.26 Datensatzeingriffe (Actions) im Fall D3-31_DE_Heizrohr

Maßnahme / Ereignis	Bedingung	Action	Parameter
Bruchöffnung an einem DE-Heizrohr	T0	up	-
Ende der Simulation	TEND	terminate	-

Tab. 5.27 Kontrollbedingungen (Trigger) im Fall D3-31_DE_Heizrohr

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	8.000	-

5.2.15 Ereignis #16 - D3-23_KMV_01F

Die Simulation eines mittleren Lecks im Primärkreis innerhalb des Sicherheitsbehälters (Leckquerschnitt ≤ 0,1F) ermöglicht die Verifizierung der implementierten RESA-Anregekriterien und der Notkühlkriterien im Reaktorschutzsystem. Zudem können die im Datensatz implementierten Prozesssignale für die Energie- und Massenübertragung aus der Primärseite in den Sicherheitsbehälter überprüft werden.

Neben der Leckauslösung ist zur Steuerung der Simulation eine Anpassung der maximalen Zeitschrittweite (SETSGHM) erforderlich um dem numerischen Lösungsverfahren eine höhere Stabilität zu verleihen. Die Lecköffnung selbst erfolgt aus demselben Grund mit einem zeitlichen Gradienten. Die zur Steuerung erforderlichen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.28 und Tab. 5.29 aufgelistet.

Tab. 5.28 Datensatzeingriffe (Actions) im Fall D3-23_KMV_01F

Maßnahme / Ereignis	Bedingung	Action	Parameter
Anpassung der max. Zeitschrittweite	T0	jump	1,0E-03
Lecköffnung Heißer Strang	T1	ramp	0,1, 0,1
Anpassung der max. Zeitschrittweite	T2	jump	0,5
Ende der Simulation	TEND	terminate	-

Tab. 5.29 Kontrollbedingungen (Trigger) im Fall D3-23_KMV_01F

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>=	7.000,9	-
T1	simulierte Zeit [s]	>=	7.001,0	-
T2	simulierte Zeit [s]	>=	7.001,1	-
TEND	simulierte Zeit [s]	>=	9.000,0	-

5.2.16 Ereignis #17 - D3-30_DH_SiVentil

In dieser Transiente wird ein Fehlöffnen und Offenbleiben eines Druckhaltersicherheitsventils unterstellt. Sie dient der Prüfung einer richtigen Implementierung folgender Funktionen und Systeme:

- Notkühlkriterien in Reaktorschutzsystem;
- Bildung der Störfallerkennungssignale „Abf. 100 K/h“, „KMVÜ“ und „KMV-HD“;
- Einsatzlogik „Kühlmittelverluststörfall“ für die Durchführung der Maßnahmen anhand der KMV-Kennlinie aus MADTEB.

Weiter wird die korrekte thermohydraulische Nachbildung der DH-Sicherheitsventile sowie die Modellierung der Leitungen zur Verbindung zwischen Druckhalter und Abblasebehälter geprüft. Es erfolgt zusätzlich die Verifizierung des implementierten Modells des Abblasebehälterkühlers. Die dazu notwendigen und im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.30 und Tab. 5.31 aufgeführt.

Tab. 5.30 Datensatzeingriffe (Actions) im Fall D3-30_DH_SiVentil

Maßnahme / Ereignis	Bedingung	Action	Parameter
Generatorleistungssollwert erhöhen	T0	jump	1.522
Generatorleistungssollwert setzen	T0	up	-
RESA-Anregekriterium 1 rücksetzen	T0	down	-
RESA-Anregekriterium 2 setzen	T0	up	-
Diesel unverfügbar	T1	down	-
420 kV Netz unverfügbar	T2&T3	down	-
Generator unverfügbar	T2&T3	down	-
Fehlöffnen des DH-SiV	T3	jump	1
Ende der Simulation	TEND	terminate	-

Tab. 5.31 Kontrollbedingungen (Trigger) im Fall D3-30_DH_SiVentil

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>=	7.001	-
T1	simulierte Zeit [s]	>=	7.200	-
T2	RS-Signal TUSA	>	0,9	-
T3	simulierte Zeit [s]	>=	7.400	-
TEND	simulierte Zeit [s]	>=	9.000	-

5.2.17 Ereignis #18 - D3-05_FD_Leck

Eine Simulation eines Lecks im Frischdampfsystem innerhalb des Sicherheitsbehälters eignet sich zur Verifizierung von Schutzsignalen für die sekundärseitige Absperrung des Dampferzeugers sowie für die Verifizierung der Implementierung des Druckabfall-Kriteriums (DAF) und des Teilabfahren-Signals im Reaktorschutzsystem. Das Leck ist dabei in der Frischdampfleitung von Dampferzeuger hinter der Frischdampfabschlussarmatur (FD-AA) mit einer Querschnittsfläche von ca. 150 cm² implementiert. Insbesondere wird hier die Nachbildung der sekundärseitigen Absperr- und Abfahrsignale sowie Gebäudeabschluss- und Notspeisesignale geprüft. Des Weiteren erfolgt die Verifizierung des Analysesimulators bzgl. des Auslösens der Schnellabschaltsysteme mit der Reaktor- und Turbinenschnellabschaltung. Gleichzeitig ermöglicht die Simulation dieser Transiente die Prüfung der im Analysesimulator implementierter Ventile und Pumpen, welche im Rahmen der Störfallbeherrschung direkt vom Reaktorschutzsystem gesteuert werden.

Die zur Steuerung der Transiente im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 5.32 und Tab. 5.33 aufgelistet.

Tab. 5.32 Datensatzeingriffe (Actions) im Fall D3-05_FD_Leck

Maßnahme / Ereignis	Bedingung	Action	Parameter
Lecköffnung FD-Leitung	T0	up	-
Ende der Simulation	TEND	terminate	-

Tab. 5.33 Kontrollbedingungen (Trigger) im Fall D3-05_FD_Leck

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>=	7.001	-
TEND	simulierte Zeit [s]	>=	9.000	-

5.2.18 Ereignis #19 - D4a-04_ATWS

Der Ausfall der Hauptspeisewasserversorgung mit einem mechanischen Verklemmen aller Steuerstäbe (ATWS) wird zur Prüfung des Zusammenspiels von betrieblichen und Sicherheitssystemen durchgeführt, welche für die primärseitige Beherrschung von ATWS-Störfällen relevant sind. Dabei wird die Nachbildung folgender Systeme im Analysesimulator geprüft:

- vierfach redundantes Zusatzboriersystem;
- betriebliches Boriersystem;
- Volumenregelsystem;
- STEW, RELEB und STAFAB.

Die Begrenzungseinrichtungen spielen beim Erkennen des ATWS-Störfalls und bei der Einleitung entsprechender aktiver Gegenmaßnahmen zu seiner Beherrschung eine entscheidende Rolle. Ihr Verhalten wird bei der Analyse dieser Transiente verifiziert. In Tab. 5.34 und Tab. 5.35 sind die zur Steuerung der Transiente notwendigen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) aufgelistet, welche im zugehörigen Ablaufprotokoll festgehalten sind.

Tab. 5.34 Datensatzeingriffe (Actions) im Fall D4a-04_ATWS

Maßnahme / Ereignis	Bedingung	Action	Parameter
Blockieren aller Steuerstäbe	T0	up	-
Komponentenschutz SpWP 1 ein	T1	up	-
Komponentenschutz SpWP 2 ein	T1	up	-
Verfügbarkeit SpWP 3 aus	T1	down	-
Ende der Simulation	TEND	terminate	-

Tab. 5.35 Kontrollbedingungen (Trigger) im Fall D4a-04_ATWS

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>=	7.001	-
T1	simulierte Zeit [s]	>=	7.006	-
TEND	simulierte Zeit [s]	>=	7.700	-

6 Definition der Bandbreite für die wesentlichen simulationsspezifischen Anlagenparameter

Damit die Verifizierung der Datensätze der Analysesimulatoren automatisch auf der GRS-Plattform zur kontinuierlichen Integration erfolgen kann, ist eine Überprüfung der erzielten Ergebnisse anhand des Vergleichs mit vorgegebenen Referenzkurven erforderlich. Dafür ist eine sorgfältige Auswahl simulationsspezifischer Anlagenparameter für jedes Ereignis der anlagenspezifischen Simulationsmatrix (siehe z. B. Tab. 5.1 für den Vorkonvoi-Reaktor) notwendig. In /PAL 18/ wird eine Mindestzahl von sechs Anlageparametern pro Rechenlauf vorgeschlagen, um eine automatische Verifikation durchzuführen. Die notwendige Anzahl kann aber in Abhängigkeit vom betrachteten Ereignis stark nach oben abweichen.

Um die Arbeit für die Auswahl der simulationsspezifischen Anlagenparameter zu unterstützen, sind die Ergebnisse aus dem vorangegangenen Vorhaben 4715R01345 /PAL 18/ maßgeblich eingeflossen. Dort wurden Simulationsergebnisse und das Verhalten des betrachteten Analysesimulators eines Vorkonvoi-Reaktors anhand vorliegender anlagenspezifischer Dokumentation, Simulationsergebnisse von durch den Betreiber durchgeführten Rechnungen (Sicherheitsstatusanalyse) sowie Informationen aus den Schulungsunterlagen und dem Störfallhandbuch der GRS (Vorhaben 3612R01335) verifiziert. Die im Rahmen des Vorhabens 4715R01345 erzeugten Simulationsergebnisse bilden die erste Grundlage für die Erzeugung von Referenzkurven zur Bewertung von Verifikationsergebnissen des im laufenden Projekt vorrangig betrachten Vorkonvoi-Datensatzes. Die Auswahl simulationsspezifischer Anlagenparameter wurde im Zuge der Arbeiten zur exemplarischen Durchführung der automatischen Verifikation (siehe Kapitel 8) weiter ergänzt und entsprechende Signallisten erstellt. Eine Auflistung der simulationsspezifischen Signale und Anlagenparameter, welche zur Verifikation des Vorkonvoi- respektive des Konvoi-Datensatzes verwendet werden, wurde aufgeführt. Diese dient der Zuordnung der in den Datensätzen verwendeten Nomenklatur zu den jeweiligen Anlagengrößen und einer Vereinfachung der Reproduzierbarkeit. Diese Listen liegen in Form von Excel-Dateien (`<ABK>_Signals.xlsx`) in den Repositorys der anlagenspezifischen Ablaufprotokolle. Dort sind die gelisteten Signale zusätzlich denjenigen Ereignissen der Simulationsmatrix zugeordnet, für welche sie zur Bewertung relevant sind.

Referenzkurven stellen im hier beschriebenen Kontext also die direkten Ergebnisse verifizierter Ereignissimulationen in Form von zeitlichen Verläufen simulationsspezifischer

Anlagenparameter dar. Grenzkurven werden aus diesen Referenzkurven oder einem Ensemble an Referenzkurven abgeleitet und bilden einen Akzeptanzkorridor für Abweichungen in Ergebnisgrößen nachfolgender Verifikationsrechnungen. Das Vorgehen zur Ableitung von Grenzkurven für einen beliebigen Datensatz erfolgt grundsätzlich nach folgendem Schema:

1. Es wird für den zu verifizierenden Datensatz eine geeignete Bewertungsbasis identifiziert,
2. für die Verifikation relevante Ereignisse werden bestimmt und simuliert,
3. die Ergebnisse in Form simulationsspezifischer Anlagenparameter werden mit der Bewertungsbasis abgeglichen,
4. etwaige Abweichungen werden bewertet und ggf. durch Datensatzanpassungen behoben,
5. die finalen verifizierten Simulationsergebnisse bilden die erste Grundlage zur Erzeugung von Referenzkurven.

Die Referenzkurven werden danach für die automatische Verifizierung gespeichert und für die Bestimmung der Grenzkurven verwendet.

Um die Verifikationsbasis zu verbreitern und Tests robust zu gestalten, können Ergebnisse weiterer Verifikationssimulationen in eine Gruppe von Referenzkurven individuell für jedes zu untersuchende Ereignis einfließen. Die Notwendigkeit für dieses Vorgehen besteht insbesondere deshalb, weil es durch Datensatzanpassungen und Entwicklungsarbeiten am thermohydraulischen Code (ATHLET) zu Abweichungen kommen kann, welche akzeptabel sind oder die Ergebnisqualität sogar erhöhen, aber bei einzelnen Anlagengrößen zu Verletzungen der Grenzkurven führen können. Ein Beispiel für ein Ensemble an Referenzkurven, welche durch die Anwendung unterschiedlicher Setups (Kombinationen verwendeter Versionen des Analysesimulators und thermohydraulischen Codes) erzeugt wurde, ist in Abb. 6.1 dargestellt. Die Abbildung zeigt den Verlauf des Druck-Istwerts im Druckhalter bei der Untersuchung des Ereignisses „Ausfall der Hauptspeisewasserversorgung und mechanisches Verklemmen aller Steuerstäbe“ (#19 - D4a-04_ATWS; vgl. Tab. 5.1) in den qualifizierten Referenzsimulationen.

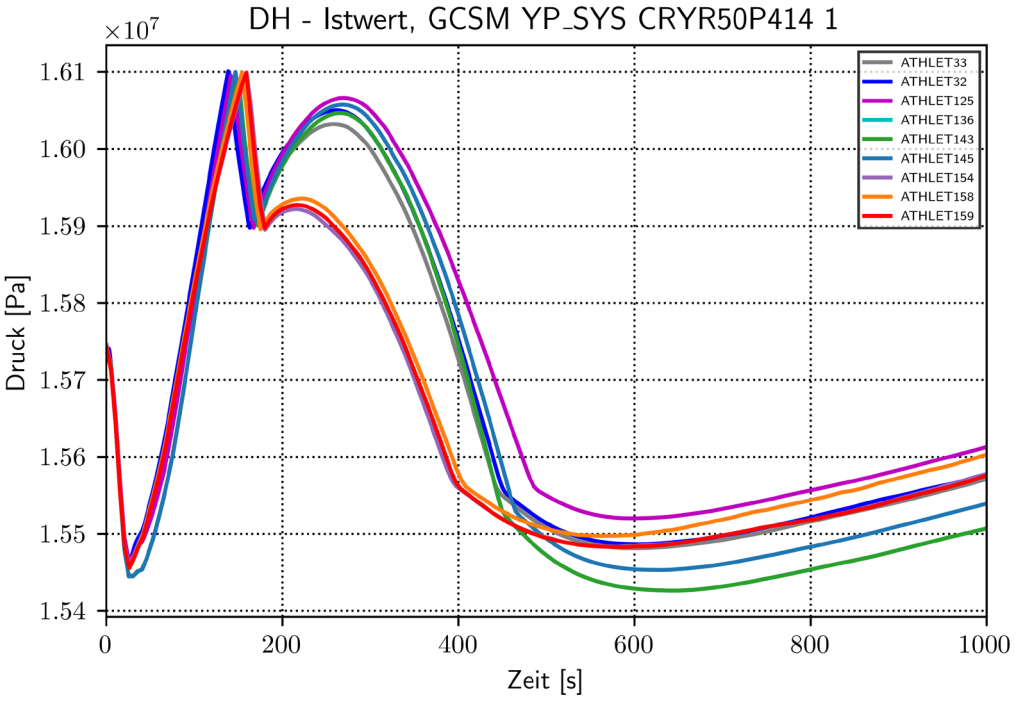

Abb. 6.1 Beispiel für ein Ensemble an Referenzkurven als Bewertungsbasis zur automatischen Verifikation

Für die Erstellung der Grenzkurven der ausgewählten Anlagegrößen wurden Skripte in der Programmiersprache Python entwickelt. Nach Erfahrung aus dem vorangegangenen Projekt /SCH 18/ werden für die erzeugten Ensemble an Referenzkurven für alle simulationsspezifischen Anlagenparameter jedes in der Simulationsmatrix berücksichtigten Ereignisses konvexe Einhüllende /GRA 72/ bestimmt, welche die Grenzkurven bilden.

In Abb. 6.2 ist dargestellt, wie aus den neun in Abb. 6.1 gezeigten Simulationsläufen unter Verwendung von unterschiedlichen ATHLET/Datensatz-Setups Grenzkurven für die physikalische Beispielgröße (DH-Druck Ist-Wert) ermittelt werden. Die Abbildung zeigt den tatsächlichen Verlauf des Maximums und Minimums des Referenzkurven-Ensembles in blau, den Verlauf des Simulationsergebnisses des aktuellen Verifikationslaufs in schwarz, den Verlauf des Simulationsergebnisses des vorangegangenen Verifikationslaufs in violett sowie das obere und untere Akzeptanzlimit (Grenzkurven) in grün. Die gezeigte Erweiterung des Grenzkurvenkorridors dient zur Vermeidung einer Fehlschlagsmeldung der Verifikation bei geringfügigen Abweichungen im Kurvenverlauf, wie er beispielsweise durch numerische Effekte sehr häufig auftreten kann. Verwendet man direkt Grenzwerte der Referenzkurven-Ensemble für die beschriebenen Integrationstests, so können bereits sehr kleinen Abweichungen zu einem Nichtbestehen der Tests

(Status: UNSTABLE) führen. Deshalb wurden die Grenzwertkurven entsprechend der Darstellung in Abb. 6.2 so verschoben, dass sie einen Abstand von 5 % zu den Referenzwertgrenzen (blau) besitzen, bezogen auf den Abstand zwischen Minimal- und Maximalwert der jeweiligen Größe und der Dauer der Transiente. Dies erfolgt auf Basis des Vatti-Clipping-Algorithmus /VAT 92/.

Die Einbeziehung des Ergebnisses des vorangegangenen Verifikationslaufs ermöglicht es, direkt die Auswirkungen etwaiger Datensatzanpassungen oder ATHLET-Entwicklungen auf die betrachtete Anlagengröße aufzuzeigen und brachte bereits großen Nutzen bei der exemplarischen Durchführung der automatischen Verifikation (siehe Kapitel 8).

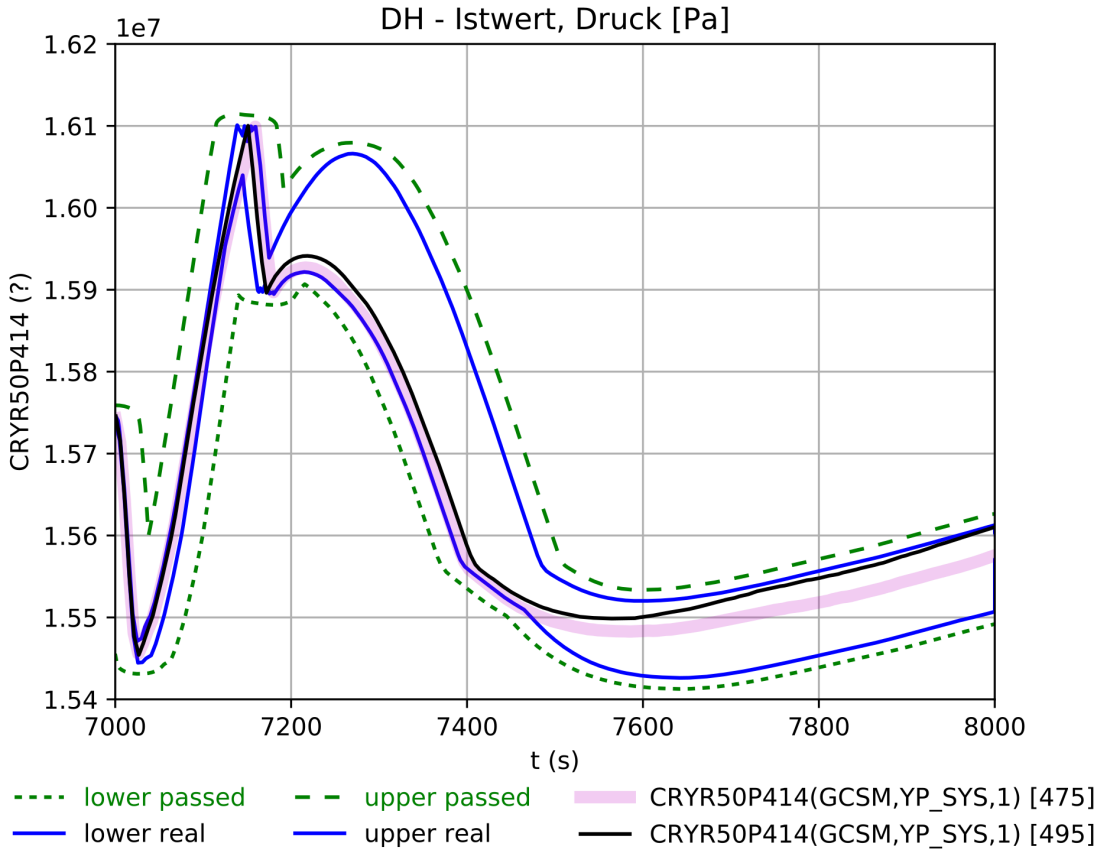

Abb. 6.2 Vergleich eines aktuellen Verifikationslaufergebnisses mit Grenzkurven und dem vorangegangenen Verifikationslaufergebnis

Die so erzeugten Grenzkurven werden dann für die automatisierte Überprüfung verwendet: Für jeden Zeitschritt der Simulation wir überprüft, ob die linear interpolierten Grenzwerte eingehalten werden. Wenn nicht, wird der Test als fehlgeschlagen markiert. Im Rahmen des Vorhabens wurden Skripte (unter Anwendung der Bibliothek *pytest*

/OKK 17/) weiterentwickelt, welche die Werte aus den Ergebnisdateien der Verifikationsläufe mit den vorgegebenen Grenzwertkurven vergleichen. Dazu werden für jeden gespeicherten Zeitschritt der Simulation interpolierte Grenzwerte ermittelt. Dann wird für jeden Simulationsschritt überprüft, dass keiner der Grenzwerte verletzt wird. Falls eine Verletzung festgestellt wird, so wird der Test als fehlgeschlagen (Status: UNSTABLE) markiert.

Eine solche Grenzwertverletzung ist in Abb. 6.3 exemplarisch dargestellt. In diesem Fall kam es durch eine Änderung am Datensatz zu einem früheren Ende des Druckabfalls und einem Wiederanstieg mit gleichem Gradienten, was zu einem Auslaufen des Druckwerts aus dem Grenzkurvenkorridor führte. Um den Entwicklern die Fehlersuche zu erleichtern, wird der Bereich der Grenzwertkurve, in dem die Grenzwertverletzung aufgetreten ist, rot markiert.

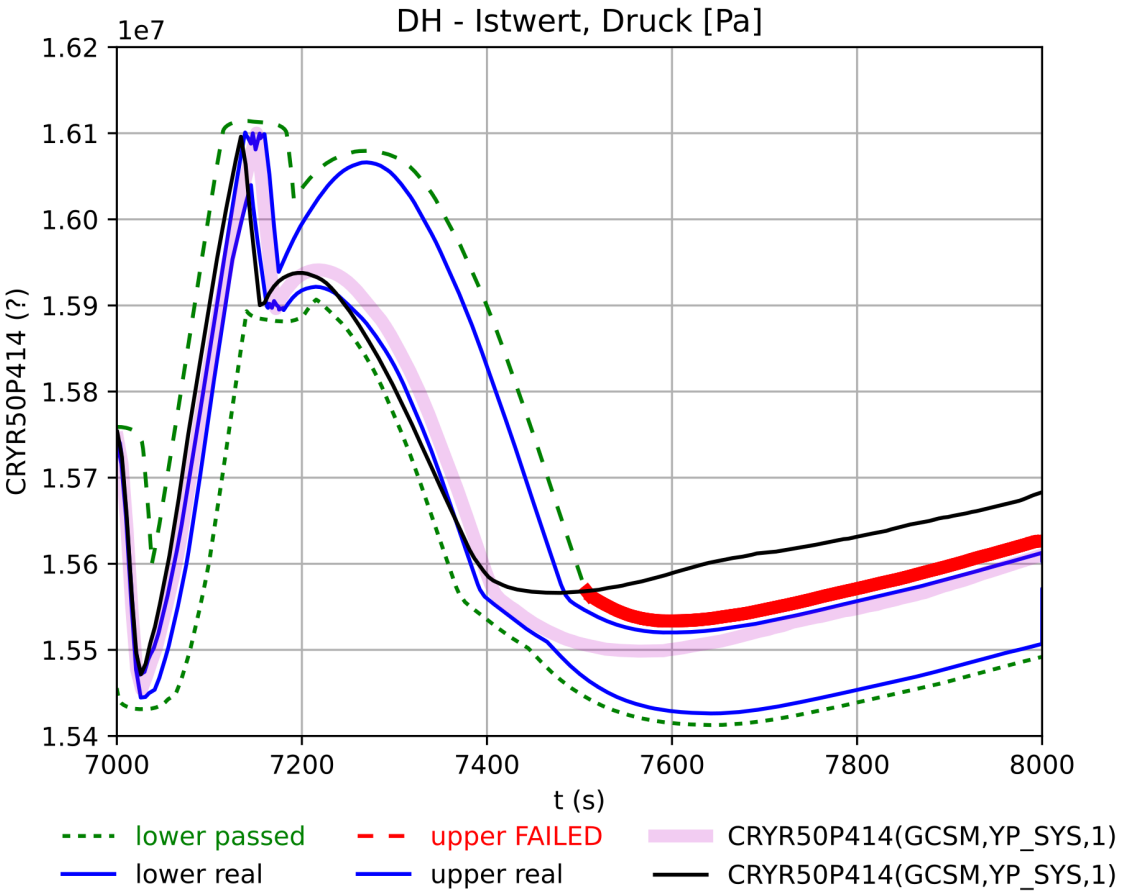

Abb. 6.3 Verletzung einer Grenzkurve infolge von Änderungen im Datensatz

Bei Grenzwertverletzungen werden die Entwickler automatisch per E-Mail informiert. Die Ergebnisse der Tests werden mit Hilfe des Meldungsgenerators (siehe Kapitel 7)

zusammengefasst und den Entwicklern in übersichtlicher Form zur Verfügung gestellt. Es ist dann ihre Aufgabe, zu überprüfen, ob die Änderungen auf erwünschte Effekte zurückzuführen sind oder ob möglicherweise ein Fehler im Datensatz oder der verwendeten ATHLET-Version vorliegt. Sollten erwünschte Effekte zu einer Verletzung der Grenzwertkurven geführt haben, so sind die Grenzwertkurven neu zu generieren, wobei die neue Rechnung in das Ensemble der verifizierten Rechnungen aufgenommen wird. Gegebenenfalls sind aber alte Rechnungen aus diesem Ensemble zu entfernen, falls die Grenzwertkurven zukünftig enger gezogen werden sollen.

7 Aufbau eines Meldungsgenerators für die Post-Processing Phase

Um eine effektive und effiziente Überprüfung von Verifikationsergebnissen durch die Datensatzentwickler zu ermöglichen, ist eine Sammlung und Bereitstellung aller dafür notwendigen Informationen in einer übersichtlichen Form von Nutzen. Hierfür wurde im laufenden Vorhaben ein Meldungsgenerator entwickelt, welcher diese Aufgabe übernimmt. Der Meldungsgenerator besteht auf dem PowerShell-Skript `create_report.ps1` und wird im Ablauf der Verifikationsprozedur auf Gitlab in der Stufe „Summary“ verwendet.

In dieser Phase werden neben den trivialen Ausführungsfehlern der Simulation wie Programmabsturz oder vorzeitige Beendigung der Rechnung weitere Prüfabfragen getätigt, welche z. B. Rechenergebnisse oder Daten des Simulationszustands für die Bewertung zugrunde legen. Folgende Prüfarten sind im automatisierten Verifizierungsverfahren umgesetzt:

- **Plausibilitäts- und Stabilitätsprüfungen**: Der Zustand der Simulation wird während der Berechnung bewertet. Werden in den Testspezifikationen festgelegte Bedingung nicht erfüllt, kann die Simulation vorzeitig abgebrochen werden. Die verkürzte Rechenzeit wird in der Summary-Stufe (erkannt und als instabile Durchführung (Status: UNSTABLE) oder bei Abbruch mit Fehlercode in ATHLET als fehlerhaft (Status: FAILURE) markiert.
- **Integrale Ergebnisprüfungen:** Die Simulationsergebnisse werden anhand des Verlaufs ausgewählter Ergebnisgrößen automatisch bewertet. Hierzu sind Grenzkurven entsprechend der Beschreibung in Kapitel 6 spezifiziert, welche die zu akzeptierenden Abweichung im Ergebnisverlauf festlegen. Werden einige oder alle dieser Grenzkurve verletzt bzw. überschritten, gilt der Test als instabil (Status: UNSTABLE) und die Ursache der Abweichung ist von den Datensatzentwickler zu untersuchen und zu bewerten, entsprechende Fehler ggf. zu beheben. Nur wenn eine Ereignissimulation vollständig abgeschlossen und keine der vorgegebenen Grenzkurven verletzt wurde gilt die Prüfung als bestanden (Status: SUCCESS).

In der Summary-Stufe wird auf weitere im Rahmen des Vorhabens entwickelte Skripte zurückgegriffen, welche die Grenzkurven der simulationsspezifischen Anlagenparameter (siehe Kapitel 6) sowie Ergebnisse des vorangegangenen Verifikationslaufs verwenden, um die integralen Ergebnisprüfungen durchzuführen. Dies erfolgt durch den Zugriff

auf das Repository `athlet_test_scripts`, welches auch über die Anwendung im vorgestellten Vorhaben hinaus im Rahmen von Projekten zur kontinuierlichen Integration bei der ATHLET-Entwicklung (vgl. /SCH 18/) Verwendung findet.

Anschließend an die durchgeführten Prüfungen erfolgt die grafische Auswertung und Zusammenstellung der Ergebniswerte sowie aller relevanten Informationen zur Beurteilung des Verifizierungserfolgs/-misserfolgs in einer HTML-Datei, welche als Test-Report bezeichnet wird. Das gewählte HTML-Dateiformat zeichnet sich durch eine gut strukturierbare Darstellungsform und vielseitige Lesbarkeit, beispielsweise mit einem beliebigen Browser, aus. Dadurch wird weiter eine plattformübergreifende Nutzbarkeit gewährleistet. Der Test-Report besteht aus vier Teilen:

Teil I beinhaltet Informationen zu Datum und Uhrzeit des Prozessstarts sowie die ID der Pipelines in welchen der Verifikationsprozess ausgeführt wurde. Abb. 7.1 zeigt den zugehörigen Ausschnitt des Test-Reports mit Informationen zu den durchgeführten Pipelines und Jobs auf dem GRS-Gitlab-Server. Der Abschnitt enthält Verlinkungen zur grafischen Nutzeroberfläche des Gitlab-Servers für die jeweiligen Pipelines. Über diese Links können alle Artefakte abgerufen werden, welche die Ergebnisse der Kompilierungs-Jobs enthalten, falls es sich um die neuesten Artefakte handelt oder die Speicherzeit der Artefakte noch nicht überschritten wurde (siehe Kapitel 4.2.1.1). Diese beinhalten die Binarys der für die Verifikation verwendete Software und die zugehörigen Bibliotheken.

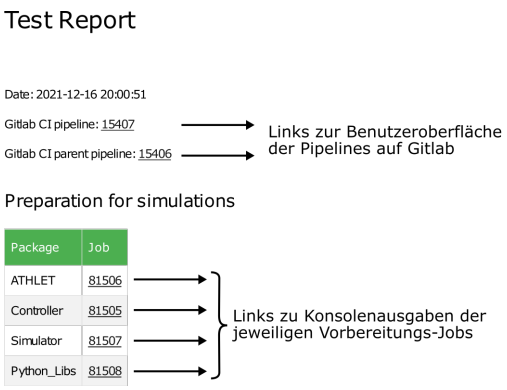

Abb. 7.1 Teil I und II des Test-Reports mit Informationen zu den durchgeführten Pipelines und Jobs auf Giltlab

Teil II (Preparation for Simulations) ordnet der Zusammenstellung und Kompilierung der für die Verifizierung notwendigen Komponenten die jeweilige Job-ID in der Gitlab-Pipeline zu. Der Abschnitt beinhaltet außerdem Verlinkungen zu den Konsolenausgaben der

Jobs sowie zu den zugehörigen Artefakten, welche hier einzeln für die jeweiligen Komponenten (ATHLET, Controller, Simulator und Python-Bibliotheken) heruntergeladen werden können. Diese Artefakte können genutzt werden, um nicht erfolgreiche Verifikationsläufe mit identischem Setup zu wiederholen und somit etwaige Fehler eindeutig identifizieren zu können. Der Teil II des Test-Reports ist ebenfalls in Abb. 7.1 dargestellt.

Teil III (Results of Simulations) nimmt den größten Bereich des Test-Reports ein und beinhaltet alle Informationen zu den einzelnen Ereignissimulationen. Abb. 7.2 zeigt ein Beispiel dieses Abschnittes des Test-Reports. In tabellarischer Form wird den Ereignissen mit einer Kurzbezeichnung nach Tab. 5.1

- der Status des Verifikationsergebnisses (SUCCESS, UNSTABLE, FAILURE),
- eine Statistik der simulationsspezifischen Tests,
- Verlinkungen zu den grafischen Auswertungen der Tests (Darstellung *aller* Testergebnisse oder *ausschließlich fehlgeschlagener* Testergebnisse), den Ergebnisdateien der Simulation (.pd, .key, .hd5, etc.) sowie zu den Konsolenausgaben der Child-Jobs auf dem GRS-Gitlab-Server und
- Metainformationen zu Start-/Endzeitpunkt und Laufzeiten der jeweiligen Jobs

zugeordnet. Über die Verlinkungen kann direkt auf die für die Analyse notwendigen Informationen und grafischen Auswertungen zugegriffen werden. Entwickler werden damit in die Lage versetzt Abweichungen im Datensatzverhalten auf einen Blick zu erkennen, direkt auf grafische Auswertungen der ereignisspezifisch wichtigen Anlagengrößen zurückzugreifen und diese mit den vorherigen Verläufen und Akzeptanzkorridoren zu vergleichen und die Simulationen ggf. mit identischem Setup schnell zu wiederholen. Informationen über die Laufzeiten können außerdem Hinweise auf die Stabilität oder numerischen Kosten bestimmter Anpassungen im Datensatz liefern.

Abb. 7.2 Teil III des Test-Reports mit allen Informationen zu den Ergebnissen der einzelnen Ereignissimulationen

Teil IV (Used Versions) erleichtert die Nachvollziehbarkeit des Einflusses von Anpassungen in den einzelnen zur Verifizierung verwendeten Komponenten und ermöglicht die Reproduktion der Simulationsergebnisse zu einem späteren Zeitpunkt, wenn verwendete Artefakte nicht mehr zur Verfügung stehen (z. B. nach Ablauf der in Teil I beschriebenen Vorhaltefrist). Dieser Abschnitt des Test-Reports ist in Abb. 7.3 dargestellt. Auch dieser Abschnitt enthält Verlinkungen zu den jeweiligen Informationen auf dem Gitlab-Server. So geht aus der dargestellten Tabelle hervor, welcher Branch des Repositorys der jeweiligen Komponente verwendet wurde. Außerdem kann auf den zu diesem Zeitpunkt aktuellen Commit zugegriffen werden. Damit ist für die Entwickler direkt ersichtlich, welche Änderungen im Vorfeld des Verifikationslaufs an den verwendeten Komponenten vorgenommen wurden. Die Effektivität und Effizienz der Analyse von Abweichungen in den Simulationsergebnissen werden damit erheblich gesteigert.

Used versions

Repository	Ref	Commit
ATHLET	v3.3	258284bd47adf5983aee0e260fa513a89ff0a21a
athlet_result_reader	master	f0cb8e29c32e1fb624384beae9381c11a79c27b3
athlet_test_scripts	master	87330a506e40c69f8a654cbcd4a5ad7c0ce10caf
atlas_plugin	master	2ce696464152a998a58c5c5f4ebd761a9ecf1f1e
Controller	duz/umzug_test	297815d94f048c3b2c43ac44bf60bf5ec3a7e8e8
qc_plugin	origin/kbr	c225c9b41deaf6467a2db2e0e68013609aecba9e
Simulator	master	51f4a524df35978196db6c65ebeea2f6c895a220
teleperm_plugin	master	354df3e1ab6c5b70d871aad8621ce748c5b84cb7

Links zu den Branches der verwendeten Repositories auf Gitlab

Links zu den Commits der verwendeten Versionen; z. B.:

Abb. 7.3 Teil IV des Test-Reports mit Informationen zu allen verwendeten Software-Versionen

8 Exemplarische Durchführung der automatischen Verifizierungsprozedur und Auswertung der Ergebnisse der Simulationen

8.1 Allgemeine Anwendung des automatisierten Verifizierungsverfahrens

Die entwickelte Vorgehensweise zur automatischen Verifizierung von Datensätzen wurde exemplarisch angewendet, um die entwickelten Prozeduren und den gesamten automatisierten Verifikationsprozess zu prüfen. Dazu wurde zunächst der Datensatz eines Vorkonvoi-Analysesimulators einbezogen, für welchen qualifizierte Simulationsergebnisse aus vorangegangenen Vorhabens 4715R01345 /PAL 18/ vorliegen. Diese werden im Rahmen der Verifikation als Referenzen verwendet. Zur Prüfung der Robustheit des Verfahrens wurden Modifikationen und Erweiterungen in den betrieblichen Systemen am Eingabedatensatz vorgenommen, welche im Rahmen des Vorhabens 4717R01334 /PAL 17/ als Optimierungsmaßnahmen identifiziert wurden. Dies betrifft insbesondere die Überarbeitung der implementierten Füllstandsmessung (siehe Kapitel 8.2.1) sowie eine Reihe von Anpassungen in der Modellierung der Leittechnik, darunter die folgenden:

- Anpassung der Logik für die Absenkung des FD-Max-Drucksollwertes für die FDU nach DEHEIRO,
- Anpassung der Logik für das Hochsetzten der DE-Druckabsicherung,
- Korrektur der Signalbildung „Verriegelung Sperre AUF von HD-Red. Station“ in Volumenregelsystem anhand der verfügbaren Systemdokumentation
- Anpassung der Logik für die Signalbildung der DE-Druckabsicherung
- Anpassung der Logik für die Sicherheitsgefahrenmeldung „DE-Heizrohrleck I“ und „DE-Heizrohrleck II“ im Reaktorschutz.

Nach abgeschlossenen Entwicklungsarbeiten am Datensatzes wird die Durchführung der Verifizierungsprozedur auf der Gitlab-CI automatisch gestartet. Es wird für alle Ereignisse der Simulationsmatrix (siehe Tab. 5.1) geprüft, ob die hinterlegten Grenzkurven aus den qualifizierten Referenzrechnungen weiterhin eingehalten werden. Bei Verletzungen der Grenzkurven erfolgt eine detaillierte Analyse der Ergebnisse und eine Bewertung der Abweichungen. Gegebenenfalls sind daran anschließend weiter Anpassungen im Datensatz oder, falls die Abweichungen auf erwünschte Effekte zurückzuführen sind,

Anpassungen an der Grenzkurven selbst notwendig. Ziel der exemplarischen Durchführung ist es außerdem zu prüfen, ob die implementierten Testabläufe und der Umfang der Auswahl simulationsspezifischer Anlagenparameter ausreichen, um ein Simulationsergebnis weitestgehend automatisch auswerten zu können.

Zusätzlich zur Integration des Vorkonvoi-Datensatzes die automatische Verifizierungsprozedur auf einen Konvoi-Datensatz übertragen. Dazu war eine Anpassung der Quellen-(Input) und Ablagestruktur (Output) notwendig in der Art, dass unter Einhaltung von Nomenklaturkonventionen das automatisierte Verifizierungsverfahren nun auf beliebige in der GRS entwickelte Datensätze übertragbar ist. Weiter wurden die für die Steuerung der automatisierten Verifizierung auf der Gitlab-CI verwendeten Skripte (YAML-Dateien; .yml sowie PowerShell; .ps1) überarbeitet und generalisiert. Folgende Erweiterungen wurden hier vorgenommen:

- Bei manuellem Start der Verifizierung ist zusätzlich der zu verifizierende Reaktor anzugeben. Für die zukünftige Berücksichtigung weiterer Datensätze ist dabei ausschließlich die Nomenklaturkonvention einzuhalten, um sie in die Verifizierung einzubeziehen. Zusätzliche Anpassungen an den YAML-Skripten sind nicht erforderlich.
- Bei dem Start der Verifizierung können mehrere Reaktoren angegeben werden. Die Verifizierung wird dann seriell für alle angegebenen Reaktoren durchgeführt.

Für die Überprüfung der Simulationsergebnisse gegen vordefinierte Grenzkurvenverläufe einer Auswahl transientenspezifischer Signale sind die Skripte zur Erstellung der Grenzkurven derart angepasst worden, dass hier eine List zu verifizierender Reaktoren (Datensätze) angegeben werden kann. Dateien im HDF5 Format, welche die anlagen- und transientenspezifischen Grenzkurven als Akzeptanzkorridor gegen Abweichungen enthalten, werden in der abschließend aufgelegten Ordnerstruktur (siehe Abb. 4.3 sowie Abb. 8.5) den jeweiligen zu verifizierenden Datensätzen zugeordnet und entsprechend abgelegt. Eine detaillierte Beschreibung der Übertragung der Verifikationsprozedur auf den Konvoi-Datensatz findet sich in Kapitel 8.3.

8.2 Anwendung an einem generischen Vorkonvoi-Datensatz und damit verbundene Anpassungen am Analysesimulator

8.2.1 Überarbeitung der implementierten Füllstandsmessung in Dampferzeuger und Druckhalter

Behälterfüllstände sind oftmals Eingangsgrößen für Regelungen. Zu den elementaren Zielen, welche in einem Leistungsreaktor einzuhalten sind, gehören die Kontrolle des Kühlmittelinventars und Sicherstellung der Nachwärmeabfuhr. Auch hierfür ist die Kenntnis von Behälterfüllständen von großer Bedeutung. Deshalb werden viele Behälterniveaus vom Reaktorschutz überwacht. Beispiele hierfür sind die Füllstände der Dampferzeuger (DE) und im Druckhalter (DH).

Bisher wurde für die Berechnung des Füllstandes in den DE und im DH mit den Analysesimulatoren das Collapsed-Level Model von ATHLET verwendet. Das Model fasst den Flüssigkeitsbestand einer vertikalen Rohrleitung zusammen und stellt ihn als Wasserpegelniveau dar. Die in den Anlagen vorhandenen Füllstandsmessungen für die DE und den DH basieren auf dem Prinzip der hydrostatischen Messung durch Differenzdruck mit außen- bzw. innenliegenden Messsäulen. Der durch das Collapsed-Level Model von ATHLET berechnete Füllstand im DH und in den DE weicht aufgrund des unterschiedlichen Berechnungsverfahrens leicht vom in der Anlage angezeigten Füllstand u. a. bei Leistungsvariationen bzw. während Transienten und Störfällen-Berechnung ab. Um diese Abweichungen und deren Einfluss auf dem Transientenverlauf zu quantifizieren, ist die Entwicklung eines Modells für die Simulation des realen und angezeigten Füllstandes im DH und DE erforderlich.

Simulation des realen und angezeigten Füllstandes in Druckhalter und Dampferzeuger

Bei der Messung des Füllstandes im DH ist zu beachten, dass das Medium im DH in zwei unterschiedlichen Phasen auftritt, nämlich als Wasser und als Dampf. Über den Anschluss der Vergleichssäule im Dampfraum des DH gelangt Dampf in ein Kondensatgefäß und kondensiert dort. Das Kondensatgefäß ist bis zur Einbindung der Dampfleitung mit Kondensat gefüllt. Dieser Wasserspiegel stellt die obere Begrenzung der Vergleichssäule dar. Die Wasserphase des DH ist an den Messanschluss der Bartonzelle angeschlossen. Die mit kaltem Kondensat gefüllte Vergleichssäule endet an der Bartonzelle.

Der Füllstand im DH ergibt sich nach folgender Formel (/KSG 07/):

$$h = \frac{(\rho_V - \rho_D) * g * H - \Delta p}{(\rho_W - \rho_D) * g} \qquad 8.1$$

Die Bedeutung der einzelnen Größe kann aus der Abb. 8.1 abgeleitet werden.

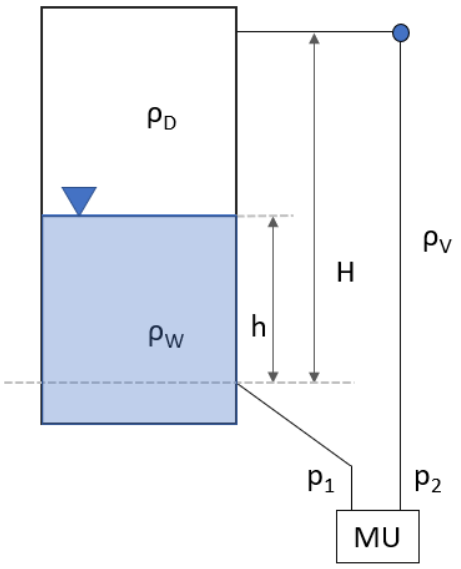

Abb. 8.1 Prinzip der Füllstandsmessung durch Differenzdruck bei siedendem Wasser

Aus der Formel ist ersichtlich, dass der gemessene Füllstand von den Dichten ρ_W, ρ_D und ρ_V, abhängt. Hiervon ist nur ρ_V während des Betriebes konstant, die beiden anderen Dichten andern sich, sobald sich der Kühlmitteldruck (KMD) ändert. Dies führt dazu, dass die DH-Niveaumessung nur für den Lastzustand gilt, bei dem die Messung kalibriert wurde. Für alle abweichenden Zustände zeigt die Messung falsche Werte an. Man kann diesen Effekt kompensieren, indem man in die Messkette eine automatische Temperaturkorrektur einführt. Mit dieser Korrektur zeigt die Messung dann bei allen Betriebszuständen den realen Füllstand an.

Die Füllstandmessung für den DH wurde als GCSM Block mit Hilfe des GRS-Tools AGM entwickelt. Als Eingangsgröße wurde der Druck am oberen bzw. unteren Objekt im Druckhaltermodell von ATHLET verwendet. Die Temperatur des Kühlmittels in der Messsäule wurde abgeschätzt und als konstanter Wert im Modell eingegeben. Für die Ermittlung des realen Füllstandes im Modell wurden die Korrekturfaktoren K1 und K2 auf Basis ausgewählter Kalibrier-Eichpunkte berechnet. Das entwickelte Modell erlaubt somit ein

Vergleich zwischen dem realen und angezeigten DH-Füllstand für das ganze Druckspektrum im Primärkreis.

Für die Füllstandsmessung in den DE gilt zunächst einmal das gleiche, was zur DH-Füllstandsmessung erläutert wurde. Die DE-Füllstandmessungen (Weitbereich- und Schmalbereichsmessungen) sind auf den Leistungsbetrieb kalibriert und zeigen daher beim Abkühlen der Anlage unterschiedliche Abweichungen zum Realwasserstand in den DE. Um das Verhältnis zwischen angezeigten und realen Wasserfüllständen zu simulieren, wurden Korrekturfaktoren auf Basis ausgewählter Kalibrier-Eichpunkte berechnet.

Ähnlich wie bei der Füllstandmessung im DH wurde der Füllstand (sowohl Schmal- als auch Weitbereich) als GCSM Block mit Hilfe des GRS-Tools AGM entwickelt. Als Eingangsgröße wurde der Druck an den Thermofluid-Objekten im DE-Fallraum bzw. DE-Dampfdom von ATHLET verwendet. Die genaue Höhe der Messstützen in den DE wurde abgeleitet. Die Temperatur des Kühlmittels in der Messsäule wurde abgeschätzt und als konstanter Wert im Modell eingegeben. Für die Ermittlung des realen Füllstandes im Modell wurden Korrekturfaktoren auf Basis ausgewählter Kalibrier-Eichpunkte berechnet. Das entwickelte Modell erlaubt somit ein Vergleich zwischen dem realen und angezeigten DE-Füllstand für das ganze DE-Druckspektrum.

8.2.2 Ausfall aller in Betrieb befindlichen Hauptspeisewasserpumpen mit Zuschaltung der Reservepumpe (D2-09)

Während eines frühen Verifikationslaufs in der Entwicklungsphase wurde im Fall #10 - D2-09_HspW_Pumpen ein ungewöhnlicher Sprung in der Generatorleistung CPPERGIST (PG-Ist) festgestellt. Abb. 8.2 zeigt die Verletzung der Grenzkurven im Verifikationslauf, welcher zu einer Meldung mit dem Status UNSTABLE im Meldungsgenerator führte.

Zur Ursachenklärung wurde die Simulation mit dem vom CI-Server bereitgestellten Artefaktpaket mit höherer Ausgabefrequenz im Bereich des auffälligen Sprungs wiederholt. Die Signalverfolgung zeigte, dass eine fehlerhafte Rücksetzung von dem Signal 86_GEN in SPEISE-PERL den Leistungssprung verursachte. Um das Fehlverhalten zu korrigieren wurde der Eingang des Speichers SPSPWB05 überarbeitet. Laut Vorgabe sollte ein Signal diesen Speicher rücksetzen. Durch eine Vereinfachung in der Implementierung war hier eine Konstante vorgegeben.

Die Vereinfachung führte zu einer Zielwert-Veränderung in der Generatorleistung, welche im Zusammenhang mit dem Setzen bzw. Rücksetzung des "Schnelle PERG-Reduktion"-Signals stand.

Um den Sprung des Generatorleistung-Ist- und Sollwerts zu minimieren, wurde weiter der PERG-Zielwert erhöht. Nach der Rücksetzung des "Schnelle PERG-Reduktion"-Signals erfolgt in der Simulation mit dem so überarbeiteten Datensatz eine quasi stoßfreie Umschaltung des Sollwertes.

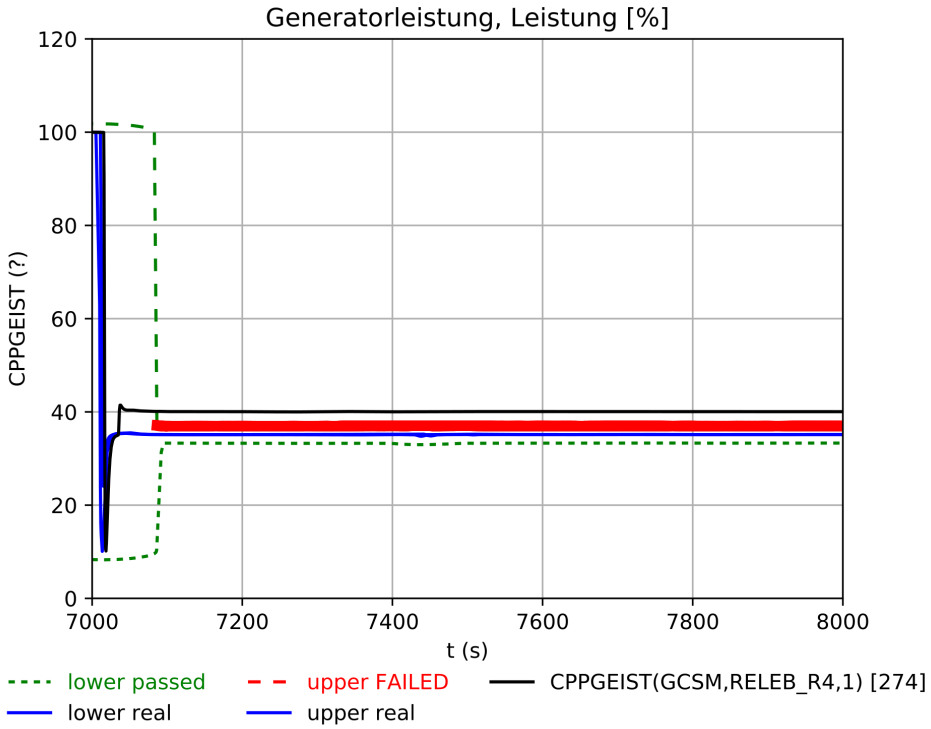

Abb. 8.2 Sprung in der Generatorleistung (PG-Ist) mit Verlassen des Akzeptanzkorridors in der automatischen Verifikation von D2-09_HspW_Pumpen

8.2.3 Fehlöffnen der HD-Reduzierstation (D2-16)

In einem Verifikationslauf des Ereignisses #5 “Fehlöffnen der HD-Reduzierstation“ (D2-16) (siehe Tab. 5.1) wurden starke Oszillationen im Druckverlauf des Primärkreises mit Verletzungen der Grenzkurven festgestellt (siehe Abb. 8.3; links). Die daraufhin erfolgte Analyse des Simulationsergebnisses zeigte außerdem abweichendes Verhalten im Verlauf der Generatorleistung auf. Die Generatorleistung verbleibt im betrachteten Fall in der Nähe des Nennwerts, während sie in der Referenz mit stabilem Gradienten absinkt. In der Referenzrechnung findet eine Umschaltung in der Turbinenregelung von

Leistungs- auf Druck-Regelung ab ca. 900 s nach Störfallinitialisierung statt. Im untersuchten Verifikationsergebnis schaltet die Regelung ständig zwischen Leistungs- und Druck-Regelung hin und her, was die Ursache der Druckoszillationen ist, und die Generatorleistung sinkt folglich weniger stark ab.

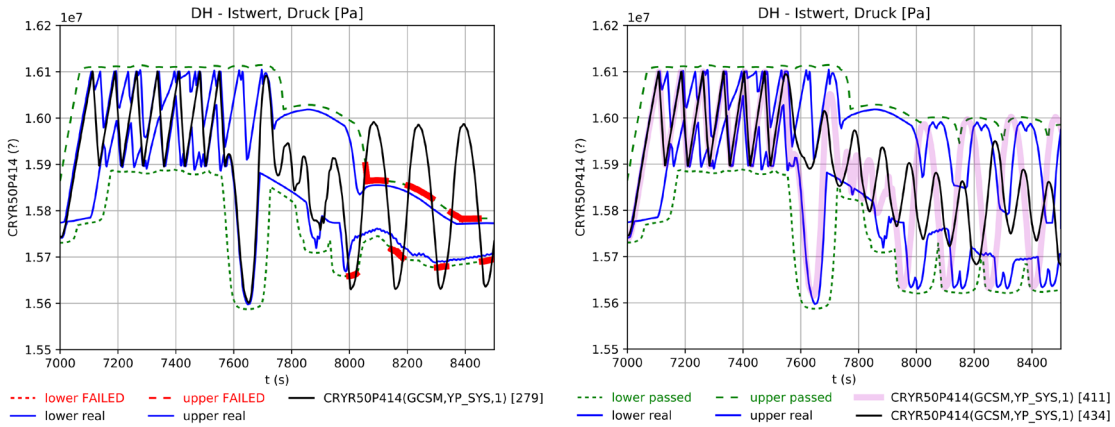

Abb. 8.3 Druckoszillationen im Primärkreis (links) und angepasste Grenzkurven mit aktualisiertem Verifikationsergebnis (rechts)

Ein weiterer Unterschied zwischen der Referenzrechnung und dem untersuchten Verifikationsergebnis wurde für das Signal SE01CSP11 identifiziert. In der Referenzrechnung zeigt das Signal einen positiven Wert, im Verifikationsergebnis verbleibt der Wert auf Null. Dieses Verhalten erscheint nach Prüfung der Steuerungslogik korrekt und das Signal SE01CSP11 wurde folglich mit einem konstanten Wert Null initialisiert.

Die auffälligen Druckoszillationen wurden mit dieser Änderung nicht beseitigt. Eine Gegenüberstellung des Verifikationsergebnis mit der Referenzrechnung deutet auf einen Effekt bei der Auslösung von LOOP-RELEB GW30 hin. In der Referenzrechnung bleibt das Grenzwertsignal GW30 ab ca. 900 s nach Störfallinitialisierung aktiv, denn der Rücksetzen-Wert des zugehörigen Signal-Switches wird nicht erreicht. Das Verhalten verursacht die Umschaltung von Leistungs- auf Druckregelung der Turbinenleistung und zurück und somit sinkt die Generatorleistung oszillierend um einen Mittelwert entsprechend dem sekundärseitigen Druckverlaufes.

Das Simulationsergebnis wurde nach dieser Prüfung in das Ensemble an Referenzkurven für dieses Ereignis übernommen. Das Ergebnis des Druckverlaufs im Primärkreis nach allen oben beschriebenen Anpassungen und unter Berücksichtigung des aktualisierten Grenzkurvenverlaufs ist in Abb. 8.3 (rechts) dargestellt.

8.2.4 Störung in der KMT-Regelung, die zu einem unkontrollierten Ausfahren von Steuerstäben führt (D2-21)

In einem Verifikationsergebnis des Ereignisses #6 „Störung in der KMT-Regelung, die zu einem unkontrollierten Ausfahren von Steuerstäben führt“ (D2-21) wurden starke Oszillationen der Kühlmitteltemperatur und das Verfahren von Steuerstäben mit Verletzungen der Grenzkurven festgestellt (siehe Abb. 8.4).

In den Referenzrechnungen wurden der Grenzwert GW20 der KMT-RELEB nicht erreicht. Im untersuchten Verifikationsergebnis wird dagegen sowohl GW20 als auch der höher liegende GW21 überschritten und führt zu den in Abb. 8.4 gezeigten Schwankungen der Kühlmitteltemperatur mit Einfahren der Steuerstäbe. Die Ursache ist hier wieder auf einen Effekt zurückzuführen. Wenn die Temperatur KMT_RELEB GW20 überschreitet kommt es zu einem Eingreifen der RELEB und dem Einfahren der Steuerstäbe. Im betrachteten Fall ist er Auslöser eine Überschreitung des GW20 um 0,07 K.

Im Zuge der Ursachenanalyse für die Abweichung im Verhalten wurde eine Aktualisierung in der Speisewasserregelung vorgenommen. Die Hauptspeisewasserventile haben nach einer Leistungserhöhung in der Referenzanlage keine Begrenzung auf 60 % Ventilöffnung mehr, wie im Datensatz noch implementiert war. Diese Begrenzung wurde entfernt, sodass die Hauptspeisewasserventile bei Anforderung nun vollständig öffnen können. Die Absenkung der DE-Füllstände fällt somit weniger stark aus.

Nach der Aufhebung dieser Begrenzung der Ventilöffnung (in RL-System) zeigt sich keine Verletzung des KMT-RELEB-Grenzwertes 20 mehr. Eine Anpassung der Grenzkurven erfolgte deshalb nicht. Es ist aber nicht auszuschließen, dass zukünftig dieser Effekt nochmal in Erscheinung tritt.

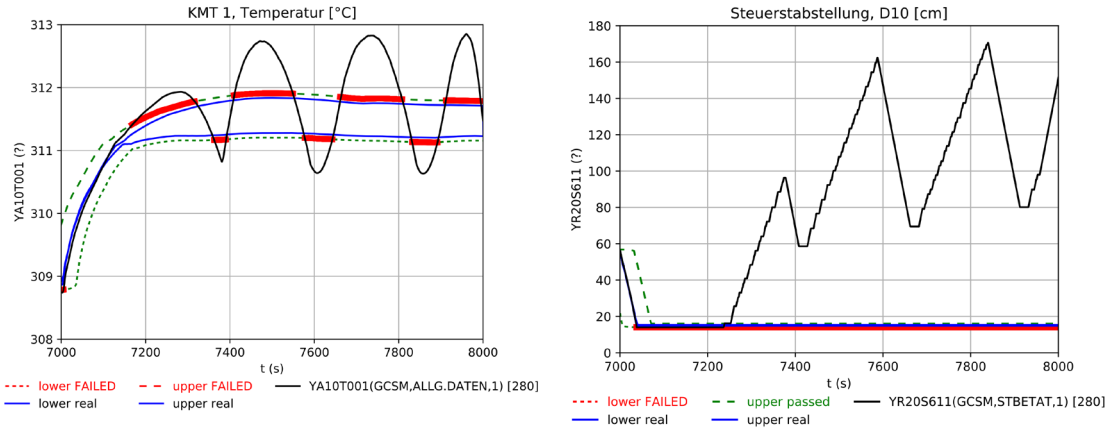

Abb. 8.4 Oszillationen der Kühlmitteltemperatur (links) und unerwartetes Verfahren der Steuerstäbe der D-Bank (rechts)

8.2.5 Ausfall der Hauptspeisewasserversorgung und mechanisches Verklemmen aller Steuerstäbe (ATWS)

Im Zuge der automatischen Verifikationsrechnungen kam es für das Ereignis #19 „Ausfall der Hauptspeisewasserversorgung und mechanisches Verklemmen aller Steuerstäbe (ATWS)“ (D4a-04) immer wieder zu druckbedingten Abbrüchen der Simulation. Die Simulation wurde durch das FEBE-Modul von ATHLET abgebrochen, wenn das Druckhalter-Sicherheitsventil schließt. Das entsprechende Signal steuert die Ventil-Position. ATHLET berechnete einen Druck vor dem Ventil von p > 230 bar. Dieser Druckwert liegt außerhalb des Gültigkeitsbereichs der physikalischen Modelle (überkritische Bedingungen) und ist nicht mehr in der CDR-Tabelle berücksichtigt.

Ursache war ein Fehler in den Vorgabewerten der Druckverlustbeiwerte von Ventilen in der CDR1D-Tabelle (Parameter CZDR). Im CW VALVE des Datensatzes waren die Druckverluste richtig angegeben, es wurden aber zusätzliche Verluste in der CDR1D Tabelle für das ABB-Ventil berücksichtigt im Parameter CZDR = 0,25 vorgegeben, welche zu einem Überlagerungseffekt geführt haben. Außerdem wurden die Reibungsverluste, aufgrund der im TFO berechneten Beschleunigungsdruckverluste, zusätzlich von ATHLET angepasst mit dem Ergebnis, dass die reibungsbedingten Druckverluste überschätzt wurden. Dies führte zu einem Überschießen des Druckwertes am entsprechenden ABB-Ventil. Nach der Korrektur (Parameter CZDR = 0) steigt der Druck nicht mehr über zulässige Werte an und die Simulation kann stabil regulär beendet werden.

8.2.6 Notstromfall gleich oder kürzer als 10 Stunden (D2-28)

Bei der Verifikation wurde festgestellt, dass die Druckhalter-Grundheizung auch im Notstromfall durch ein Begrenzungssignal in Betrieb ist. In der Datensatzimplementierung der DH-Heizung wurde die Grundheizung im Druckhalter auch bei Ausfall der Off-Site-Stromversorgung durch die Logik im Block YP_SYS zugeschaltet. Laut Schaltplan sollte die DH-Grundheizung durch den Schutz-Ein Befehl erst aktiv werden, wenn die Befehle "Sperre DH-Heizen" und "DH-Heizung AUS" nicht mehr anstehen. Eine entsprechende Änderung wurde im Leittechnikmodell umgesetzt.

Weiter sind zur Deckung der Wärmeverluste im Notstromfall die Heizstäbe nur per Hand in Betrieb zu nehmen und nicht von der KMD-Regelung angesteuert. Im Analysesimulator wurde im Zuge der Analyse der Verifikationsrechnung festgestellt, dass in diesem Ereignis mit Notstromfall die Heizstäbe vom Notstromnetz automatisch zugeschaltet werden. Um den Fehler zu beseitigen, wurde eine entsprechende Änderung der Logik umgesetzt.

Es wurde außerdem ein Fehler in den L-Bank Regelbefehl (RL-REG Block) identifiziert. Nach dem Ende einer Einschwingzeit zur Stabilisierung der Anlagenparameter wird standardmäßig die Bewegung von Steuerstäben aus der KMT-Regelung erlaubt. Im Zuge der Analyse der Verifikationsergebnisse wurde beobachtet, dass trotz sehr kleiner KMT-Regelabweichung die L-Bank automatisch bewegt wurde. Eine solche Bewegung entspricht nicht dem erwarteten Anlagenverhalten. Es wurden in Folge ein Fehler im GCSM-Block RL-REGEL im Bereich der Koordination der L- und D-Bank-Regelbefehle identifiziert.

Die im Baustein CRKMTTL1 angegebene Hysterese war falsch implementiert. Für den SWITCH ist der Bereich -1,3 < x < -1.265 gültig wobei der obere Grenzwert +1.335 betrug. Dieser Fehler wurde behoben. Die Wiederholung der Verifikationsrechnung zeigte gute Ergebnisse und keine Bewegung der L-Bank bei Regelabweichung von =< -0,13 K.

8.2.7 Abfahren der Anlage in den Zustand „Unterkritisch Kalt“

Bei der Durchführung der Verifikation des Ereignisses #2 „Abfahren der Anlage in den Zustand ‘Unterkritisch Kalt‘“ wurde festgestellt, dass die Steuerung der Heizung (Stufe 1 und 2) nicht aktiviert wurde. Laut Leittechnik-Schema werden die ersten zwei Stufen der Heizung im Druckhalter in Betrieb genommen, wenn der Sollhub-Wert der Generator-

leistung PG-sHub > +x%/min oder < -x %/min ist. Die Implementierung setzte aber PG-sHub auf >= 0,0. Die implementierte Logik führte dazu, dass nur bei Lastanstiegen keine Zuschaltung der Heizstufe möglich war. Für eine Behebung kommen zwei Vorgehensweisen in Frage, um das Signalbild zu verbessern:

1. Lösung: Anwendung des Absolutwertes des Signals GENLGRX3
2. Lösung: Anpassung der Signalkette bis CRFPGGR

Zunächst wurde die erste Lösung (Anwendung des Absolutwertes des Signals GENLGRX3) erprobt. Die Anwendung des Signals als Differentialglied scheint aber für den gewählten Modellierungsansatz nicht für die Logik geeignet. Deshalb wurde die zweite Lösung umgesetzt und getestet und führte zu zufriedenstellenden Ergebnissen mit erwartungsgemäßer Aktivierung der ersten beiden Heizstufen der DH-Heizung. Der Datensatz wurde entsprechend angepasst und eine neue Signalkette implementiert. Die alte Signalkette wurde von der Steuerung abgekoppelt.

8.3 Übertragung auf einen generischen Konvoi-Datensatz

8.3.1 Voraussetzungen zur Übertragung der Verifikationsprozedur

Zur Demonstration der Übertragbarkeit und allgemeinen Anwendbarkeit der entwickelten automatischen Verifizierungsprozedur für die in der GRS erstellten Analysesimulatoren wurde eine exemplarische Anwendung an einer Auswahl von Ereignissen mit einem generischen Konvoi-Datensatz umgesetzt. Die ausgewählten Ereignisse sollten dabei einen ausreichend hohen Grad an Komplexität in der Automatisierung aufweisen, um eine generalisierte Anwendbarkeit des automatisierten Verifikationsverfahrens nachzuweisen. Folgende Ereignisse wurden dabei bespielhaft ausgewählt:

- Volllastsimulation als Basisrechnung und zur Stabilitätsprüfung des Analysesimulators (entsprechend Ereignis #1 in Tab. 5.1),
- Fehlöffnen und Offenbleiben eines Druckhaltersicherheitsventils (D3_30) (entsprechend Ereignis #17 in Tab. 5.1),
- Versagen eines Dampferzeuger-Heizrohres (D3_31) (entsprechend Ereignis #15 in Tab. 5.1).

Für die Einbeziehung eines Datensatzes in die automatische Verifikationsprozedur ist der gesamte Prozess zur Erstellung von Spezifikationsdateien (Ablaufprotokollen) zur automatisierten Steuerung der Ereignisse und der Erstellung einer Bewertungsbasis mit Referenz- und Grenzkurven zu durchlaufen. Außerdem ist eine Speicherstruktur entsprechend der Beschreibung in Kapitel 4.2.1 für die zu verifizierenden Anlage zu erstellen.

Für die Beispielanwendung mit einem generischen Konvoi-Datensatz wurden als Bewertungsbasis Informationen aus der vorliegenden Anlagendokumentationen sowie Informationen aus verfügbaren Schulungsunterlagen und aus dem abgelaufenen Störfallanalysenhandbuch-Projekt (Vorhaben 3612R01335) herangezogen. Auf dieser Grundlage wurden Referenzrechnungen erzeugt und qualifiziert. Zur Erstellung der Grenzkurven wurden diese Referenzrechnungen mit den ATHLET Version 3.2, 3.2 Patch 1 und der aktuellen ATHLET Release 3.3 wiederholt und ein Ensemble an Referenzkurven erzeugt.

Die für die Grenzkurvenerzeugung der ausgewählten Ereignisse relevanten Parameter wurden entsprechend dem Vorgehen zum Vorkonvoi-Datensatz (siehe Kapitel 6) zusammengestellt und maschinenlesbar aufbereitet. Die entsprechende Signalliste der simulationsspezifischen Anlagenparameter wurde in Form einer Excel-Datei in dem Repository des Verifikationsprojektes (Verify[5]) auf dem GRS Gitlab-Server hinterlegt.

Für die Umsetzung der Übertragung wurde die Verifikationsprozedur generalisiert. Zur zukünftigen Einbeziehung zusätzlicher Analysesimulatoren sind die Ordnerstrukturen des Repositorys sowie im Datenarchiv (fs-gar08) entsprechend der Darstellungen in Abb. 8.5 respektive in Abb. 4.3 zu ergänzen. Grenzkurven für die Datensätze werden über die Python-Skripte im Verzeichnis „prepare_tests“ aus den Referenzrechnungen im Datenarchiv erzeugt und automatisch in Form von .hdf5-Dateien im Bereich „Ablaufprotokolle“ abgelegt. Des Weiteren sind die Dateien `case_specs.json` und `simulation_times.json` datensatzspezifisch anzupassen.

[5] https://gitlab.grs.de/grs/AnalysisSimulators/dsa_public/verify

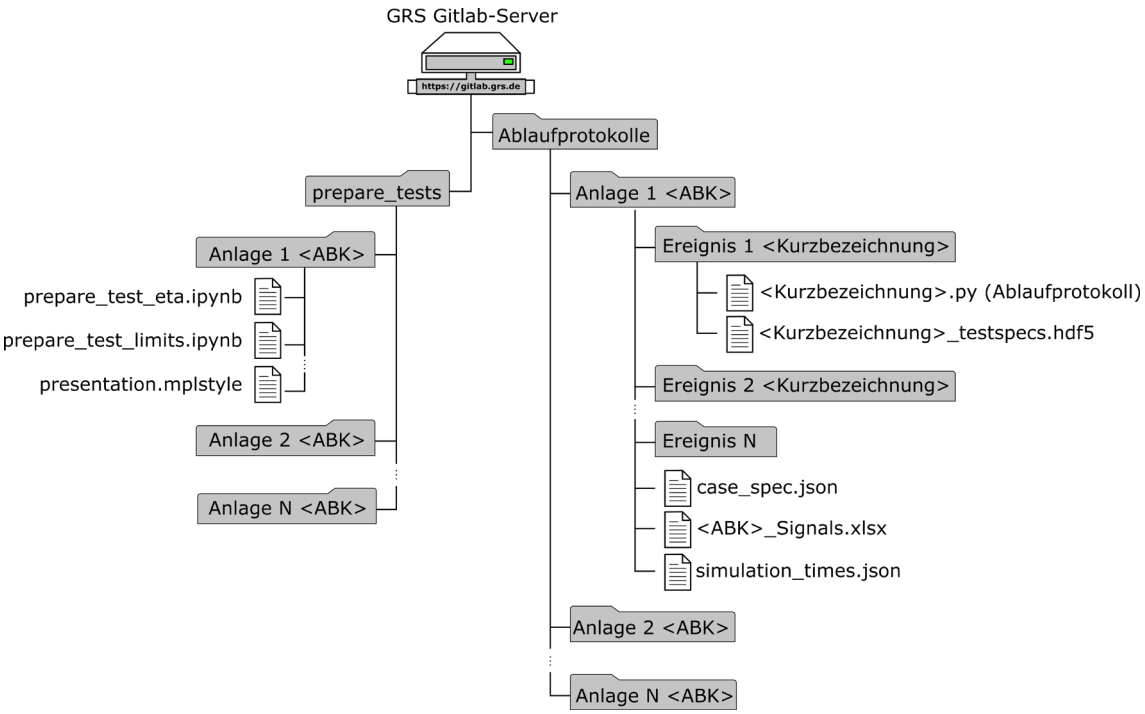

Abb. 8.5 Ordnerstruktur im Verify-Repository zur Einbeziehung zusätzlicher Analysesimulatoren in die automatische Verifikationsprozedur

8.3.2 Beschreibung erstellter Protokolle für die exemplarische Übertragung der Verifikationsprozedur auf einen Konvoi-Datensatz

Im Folgenden werden Verifikationsziel und Ablaufprotokolle der für die exemplarische Übertragung ausgewählten Ereignisse vorgestellt. Die Nummerierung der Ereignisse bezieht sich auf die Angaben in Tab. 5.1.

8.3.2.1 Ereignis #1 – Volllast (für Konvoi)

Die Berechnung eines stationären Zustandes (Vollastbetrieb) erfolgt entsprechend dem Vorgehen bei der Verifikation des Vorkonvoi-Datensatzes (siehe Kapitel 5.2) für 7.000 s simulierte Laufzeit und dient der Überprüfung des Gleichgewichts der Energie- und Massenbilanzen zwischen Primär und Sekundärseite. Im Ablaufprotokoll wird diese Laufzeit und somit der Abbruchzeitpunkt festgelegt. Die dazu angesteuerten Signale sind in Vorkonvoi- und Konvoi-Datensatz identisch implementiert, sodass die notwendigen und im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) den Angaben in Tab. 5.2 und Tab. 5.3 entsprechen. Der stationäre Endzustand der Simulation stellt den Startzeitpunkt der nachfolgenden transienten Simulationen der beiden ausgewählten Ereignisse dar.

8.3.2.2 Ereignis #15 – D3-31_DE_Heizrohr (für Konvoi)

Das doppelendige Versagen eines Dampferzeuger-Heizrohres (2F-Bruch) zeichnet sich in seinem Störfallablauf durch besondere Komplexität und hohe Anforderungen an das Zusammenwirken von betrieblichen und Sicherheitssystemen aus und wurde deshalb als beispielhaftes Ereignis zur Verifikation des Konvoi-Datensatzes ausgewählt. Der grundsätzliche Ablauf des Störfalls stimmt mit der Beschreibung in Kapitel 5.2.14 überein. Die Funktionsweise der Implementierung folgender Systeme im Datensatz wird dabei geprüft:

- das Zusatzboriersystem;
- die betriebliche Borsäure- und Deinateinspeisung sowie Chemikalieneinspeisung;
- das Volumenregelsystem;
- MADTEB, RELEB und Reaktorschutzsystem;
- Hauptkühlmittedruckhaltesystem.

Die zur Steuerung der Transiente im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 8.1 und Tab. 8.2 aufgelistet.

Tab. 8.1 Datensatzeingriffe (Actions) im Fall D3-31_DE_Heizrohr

Maßnahme / Ereignis	Bedingung	Action	Parameter
Bruchöffnung an einem DE-Heizrohr	T0	up	-
Ende der Simulation	TEND	terminate	-

Tab. 8.2 Kontrollbedingungen (Trigger) im Fall D3-31_DE_Heizrohr

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>	7.001	-
TEND	simulierte Zeit [s]	>	8.000	-

8.3.2.3 Ereignis #17 – D3-30_DH_SiVentil (für Konvoi)

Zur exemplarischen Übertragung des automatisierten Verifikationsverfahrens auf den Konvoi-Datensatz wurde ein zweites Ereignis mit besonderer Komplexität im Umfang der notwendigen Handmaßnahmen und Schalthandlungen ausgewählt. Ziel es ist dabei zum einen, das Auslösen mehrerer Child-Jobs (vgl. Kapitel 7) und die parallele

Verarbeitung von Simulationen in der Gitlab-CI zu prüfen und zum anderen komplexe Steuerungsprozesse mit dem erstellten ATHLET-Controller (vgl. Kapitel 3) in einem abweichenden Datensatz umzusetzen. Das Ereignis „Fehlöffnen und Offenbleiben eines Druckhalter-Sicherheitsventils“ (D3-30, Ereignis #17 in Tab. 5.1) ist dafür besonders geeignet, da hier mehrere Eingangssignale zur Überwachung, Auslösekriterien (Trigger) und Aktionen zur Steuerung des Simulationsablaufs (Actions) verwendet werden. In der Verifikation dient es der Prüfung einer richtigen Implementierung folgender Funktionen und Systemen:

- Notkühlkriterien in Reaktorschutzsystem;
- Bildung der Störfallerkennungssignale „Abf. 100 K/h“, „KMVÜ“ und „KMV-HD“
- Einsatzlogik „Kühlmittelverluststörfall“ für die Durchführung der Maßnahmen anhand der KMV-Kennlinie aus MADTEB.

Weiter wird die korrekte thermohydraulische Nachbildung der DH-Sicherheitsventile sowie die Modellierung der Leitungen zur Verbindung zwischen Druckhalter und Abblasebehälter geprüft. Es erfolgt zusätzliche die Verifizierung des implementierten Modells des Abblasebehälterkühlers. Die dazu notwendigen und im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 8.3 und Tab. 8.4 aufgeführt.

Tab. 8.3 Datensatzeingriffe (Actions) im Fall D3-30_DH_SiVentil

Maßnahme / Ereignis	Bedingung	Action	Parameter
Generatorleistungssollwert erhöhen	T0	jump	1,4E+09
RESA-Anregekriterium 1 rücksetzen	T0	down	-
RESA-Anregekriterium 2 setzen	T0	up	-
Diesel unverfügbar	T1	down	-
420 kV Netz unverfügbar	T2&T3	down	-
Generator unverfügbar	T2&T3	down	-
Fehlöffnen des DH-SiV	T3	jump	1
Ende der Simulation	TEND	terminate	-

Tab. 8.4 Kontrollbedingungen (Trigger) im Fall D3-30_DH_SiVentil

Bedingung	Beschreibung	Logik	Wert	Parameter
T0	simulierte Zeit [s]	>=	7.001	-
T1	simulierte Zeit [s]	>=	7.200	-
T2	RS-Signal TUSA	>	0,9	-
T3	simulierte Zeit [s]	>=	7.400	-
TEND	simulierte Zeit [s]	>=	9.000	-

8.3.3 Anpassungen und Optimierungen am Konvoi-Analysesimulator nach exemplarischer Übertragung der Verifikationsprozedur

Die Anwendung des automatisierten Verifizierungsverfahrens am Konvoi-Analysesimulator hat trotz der geringeren Anzahl an berechneten Ereignissen zur Identifizierung eines Optimierungsbedarfs von existierenden Modellen und deren Optimierung beigetragen.

Die im Folgenden aufgelisteten Punkte stellen die Beiträge des automatisierten Verifizierungsverfahrens zur Optimierung des Konvoi-Datensatzes dar:

- Verbesserung der Energiebilanz im Primärkreislauf durch die Korrektur des betrieblichen Sprühmassenstroms im Druckhalter
- Optimierung der Steuerstabfahrbegrenzung-Funktion EIKO (Überwachung und Begrenzung der Einspeise-Borkonzentration)

8.3.3.1 Verbesserung der Energiebilanz im Primärkreislauf durch die Korrektur des Sprühmassenstroms im Druckhalter

Während eines Verifikationslaufs in der Entwicklungsphase wurde im Fall #1 – Volllast bei langzeitiger Simulation (t > 1.000 s) beobachtet, dass der Druck im Primärsystem nicht stabil auf ca. 158 bar bleibt, sondern kontinuierlich steigt.

Bei der vertieften Untersuchung der Ursache dieses Verhaltens wurde als Problem die Wirksamkeit der DH-Sprühung identifiziert. Durch eine Energiebilanzierung wurde dabei festgestellt, dass die Wärmeverluste der Dauersprühung aufgrund eines zu geringen Dauersprüh-Massenstroms zu niedrig berechnet wurden. Um einen Teil der Heizleistung der Grundheizung im Druckhalter (ca. 315 kW) zu kompensieren, sollen die Verluste durch die Dauersprühung ca. 150 kW betragen. Die restlichen Wärmeverluste durch die

Strukturen im DH berechnet ATHLET auf ca. 160 kW bei einem Wärmeübergangskoeffizient von 4,7 W/(m²K).

Die Verluste der Dauersprühung ergeben sich aus der Anzahl der Sprühventile (4) multipliziert mit deren Durchsatz (ca. 0,1 kg/s) und der Enthalpieänderung, die dieser Dauersprühdurchsatz erfährt.

$Q_{Verlust\ Dauerspr.} = 4 * m_{Dauerspr.} * (h_{DH} – h_{RKL}) = 4 * 0,1 * (1.638\ kJ/kg – 1.265\ kJ/kg) \approx 150\ kW$

Der Testlauf nach der Anpassung des Dauersprüh-Massenstroms im Druckhalter weist einen stabilen Druckverlauf im Primärkreislauf auf (siehe Abb. 8.6).

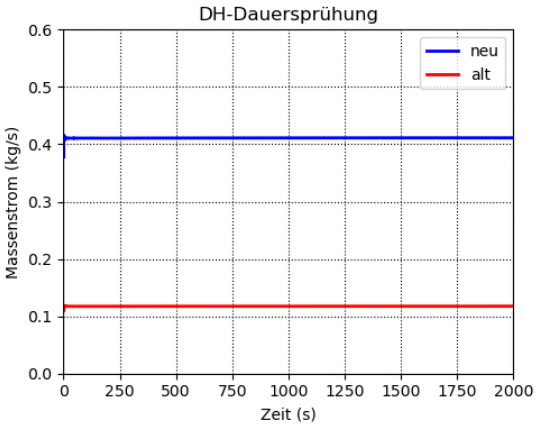

Abb. 8.6 Vergleich des Druckhalter-Sprühmassenstroms (links) und des Primärdruckes (rechts) vor und nach der Anpassung im Konvoi-Datensatz

8.3.3.2 Optimierung der Steuerstabfahrbegrenzung-Funktion EIKO (Überwachung und Begrenzung der Einspeise-Borkonzentration)

Während eines Verifikationslaufs wurde im Fall #15 – D3-31_DE_Heizrohr eine Einspeisung von Bor- und Deionat im Primärkreislauf durch das Volumenregelsystem beobachtet, welche zu einer langfristigen Entborierung des Primärsystems geführt hat.

Die eingespeiste Mischung von Borsäure und Deionat soll durch die Funktion EIKO (Überwachung und Begrenzung der Einspeise-Borkonzentration) in der Steuerstabfahrbegrenzung ständig überwacht werden. Die o. g. Funktion für die Überwachung der Einspeisekonzentration im Primärkreis war im Konvoi-Datensatz sehr vereinfacht modelliert, sodass eine Optimierung bzw. Weiterentwicklung der Logik erforderlich wurde.

Für die Weiterentwicklungsarbeit wurde die Erfahrung aus dem vorangegangenen BMU-Eigenforschungsprojekt 4715R01345 /PAL 18/ herangezogen. Die Logik für die Nachbildung der EIKO-Funktion wurde aufgrund vergleichbarer Funktionalität aus dem Vorkonvoi-Analysesimulator abgeleitet. Eine ausführliche Beschreibung der Funktion EIKO ist im Abschlussbericht des BMU-Eigenforschungsprojekts 4715R01345 /PAL 18/ zu finden.

9 Ausdehnung des automatisierten Verifizierungsverfahrens auf einen gekoppelten ATHLET/QUABOX-CUBBOX Vorkonvoi-Datensatz

Das Hauptziel des vorgestellten Eigenforschungsprojekts 4719R01375 war die Entwicklung eines automatisierten Verifizierungsverfahrens, durch dessen Anwendung an einem Analysesimulator-Datensatz mögliche Fehler im Datensatz gefunden und behoben werden können bzw. Optimierungsbedarf identifiziert werden kann, um eine umfassend verifizierte Datenbasis für Störfallanalysen zu erreichen.

Im Zuge der Arbeiten konnte der Nutzen und die Funktionalität des entwickelten automatischen Verifizierungsverfahrens an dem verwendeten Referenzdatensatz sowie eine Flexibilität bei der Übertragung auf weitere Analysesimulatordatensätze demonstriert werden. Die Ergebnisse dieser Demonstration sind in Kapitel 8 zusammengefasst. Dort wurde die automatische Verifikationsmethodik ausschließlich mit dem thermohydraulischen Code ATHLET angewendet. Um Aussagen zu einigen speziellen Störfallszenarien treffen zu können, kann es aber notwendig sein, zusätzliche Kopplungen mit externen Codes anzuwenden, beispielsweise um Prozesse abzubilden, welche stark von 3D-neutronenpysikalische Phänomenen beeinflusst sind. Aus diesem Grund wurde das Verfahren auf eine komplexere gekoppelte Version (thermohydraulisch/3D-neutronenphysikalisch) des Vorkonvoi-Analysesimulators mit den Codes ATHLET/QUABOX-CUBBOX ausgedehnt. Die Durchführung einer gekoppelten thermohydraulischen und 3D-Neutronenphysikalischen Analyse mit den Codes ATHLET/QUABOX-CUBBOX bietet die Möglichkeit, den für die Kopplung entwickelten Datensatz des Vorkonvoi-Analysesimulators abdeckend zu verifizieren. Die Integration der gekoppelten Version des Analysesimulators in das automatische Verifizierungsverfahren ermöglicht bei der Anwendung außerdem die Identifizierung von möglichem Weiterentwicklungsbedarf der gekoppelten Version des Analysesimulators im Bereich der Begrenzungen, der Neutronenflussmessung und der Leistungsverteilung im Kern.

Das grundsätzliche Vorgehen bei der Übertragung der automatischen Verifikationsprozedur stimmt mit den Beschreibungen in Kapitel 4 sowie Kapitel 8.3 überein. Auch für die Durchführung einer gekoppelten Simulation mit dem aktuellen ATHLET/QUABOX-CUBBOX Datensatz waren zunächst die relevanten Störfallszenarien zu identifizieren. Anschließend wurde der Datensatz für den Ablauf der gewählten Szenarien vorbereitet. Dazu wurden Anpassung am Datensatz selbst vorgenommen sowie entsprechende Simulationsrandbedingung vorgegeben. Während des Simulationsablaufs werden

abhängig vom zu untersuchenden Szenario Steuerungseingriffe in Form von Störeinflüssen und Handmaßnahmen vorgenommen. Zur Erstellung von Referenzkurven und daraus abgeleiteten Grenzkurven wurden die Simulationsergebnisse tiefgehend ausgewertet und mit einer geeigneten Bewertungsbasis abgeglichen, um das Systemverhalten zu verifizieren. Bei der Durchführung der ATHLET/QUABOX-CUBBOX-Rechnung konnte hierfür auf Erfahrung aus den Vorhaben 4717R01334 /PAL 17/ und 3614R01306 /POI 17/ zurückgegriffen werden.

Die zur Simulationsdurchführung notwendigen transientenspezifischen Maßnahmen (Steuerungseingriffe) wurden zur Automatisierung in der in Kapitel 3.2.2 beschriebenen Syntax in Form von Python-basierten Spezifikationsdateien protokolliert. Für eine Integration in die Verifikationsmatrix und die aktuelle CI-Umgebung wurden die YAML-Dateien zur Steuerung der Pipelines in der Gitlab-CI derart erweitert, dass eine gekoppelte Simulation mit ATHLET/QUABOX-CUBBOX durchgeführt werden kann. Kapitel 9.1 liefert hierzu einige weitere Details. Weiter wurden die verwendete Störfallszenario in den Ablauf der Simulationsfolge eingebracht. Die gekoppelten Rechnungen werden parallel zum Start der Initialrechnung (#1 Volllast in Tab. 5.1) gestartet und nicht als Restarts auf eine eigene Initialrechnung aufgesetzt. Dieses Vorgehen erwies sich aufgrund der geringen Anzahl der in der Verifikation berücksichtigten gekoppelten Simulationen als effizienter.

Zur Verifikation des gekoppelten ATHLET/QUABOX-CUBBOX Datensatzes des Vorkonvoi-Reaktors wurden die Ereignisse „Volllast mit BOC- und EOC-Kernbedingungen“ sowie die Transienten „Fehlöffnen eines FD-Sicherheitsventils (EOC) (D2-01)“, „Auswurf des wirksamsten Steuerelements (BOC)“ (D3-16) und „Lastabwurf auf Eigenbedarf (EOC)“ (D2-07) ausgewählt und manuell verifiziert. Die Ergebnisse des manuellen Verifikationsprozesses sind in Kapitel 9.2 dokumentiert. Unter Anwendung variierender ATHLET-Programmversionen sind Grenzkurven für ausgewählte Anlagenparameter erstellt worden. Diese bilden den Akzeptanzkorridor für die Verifizierung der simulierten Ereignisse und werden fortlaufend überprüft und ggf. angepasst, wenn sich der Kenntnisstand zu den berücksichtigten Transientenverläufen ändern sollte.

9.1 Notwendige Anpassungen der Verifizierungsprozedur auf Gitlab-CI

Zur Integration gekoppelter Analysen mit ATHLET/QUABOX-CUBBOX in die automatische Verifikationsprozedur war eine Anpassung der in Kapitel 4 vorgestellten Skripte

(YAML-Dateien; .yml sowie PowerShell; .ps1) zur Steuerung und Ausführung der Verifizierung in der Gitlab-CI-Umgebung erforderlich. Für die Erstellung der Setups für die Simulationen wird dazu im Prepare-Job der Prozedur (vgl. Abb. 4.2) direkt auf das Repository der QUABOX-CUBBOX-Software zugegriffen und jeweils bei Start der Verifizierung die aktuelle Software-Version kompiliert und verwendet.

Zur Optimierung der Laufzeit des gesamten Verifizierungsverfahrens mit Einbeziehung der gekoppelten Simulationen werden die ATHLET/QUABOX-CUBBOX-Läufe parallel zur Volllast-Initialrechnung gestartet auf welcher anschließend die transienten Standalone-ATHLET-Simulationen aufsetzen. Entsprechende Anpassungen in den Gitlab-CI-Job-Skripten wurden umgesetzt. Entsprechend des Vorgehens zur Erstellung der Grenzkurven und automatisierten Steuerung der Simulationsabläufe in Kapitel 5 und Kapitel 0 ist auch für die gekoppelten Rechnungen eine entsprechende Bewertungsbasis beziehungsweise sind Ablaufprotokolle zu erzeugen. Diese wurden im Zuge der Umsetzung für die ausgewählten Ereignisse in dem Repository des Verifizierungsprojekts hinterlegt. Die Ergebnisse der Verifikation der gekoppelten Analysen mit ATHLET/QUABOX-CUBBOX werden im Folgenden vorgestellt.

9.2 Ereignisse zur Verifizierung der gekoppelten ATHLET/QUABOX-CUBBOX Datensatzes

9.2.1 Volllast mit BOC- und EOC-Kernbedingungen

Die Berechnung eines stationären Zustandes (Vollastbetrieb) erfolgt für die gekoppelte ATHLET-QUABOX/CUBBOX Version für 600 s simulierte Laufzeit und dient der Überprüfung sowohl des Gleichgewichts der Energie- und Massenbilanzen zwischen Primär und Sekundärseite als auch der Übertragung der wesentlichen thermohydraulischen und neutronenkinetischen Größen zwischen den beiden verwendenden Codes.

Zwei Datensätze wurden für die Zykluszustände BOC (6 Volllasttage unter Xenon-Gleichgewicht-Bedingungen, 1.274,8 ppm Bor) und EOC (373 Volllasttage, 0 ppm Bor) vorbereitet. Die zwei Datensätze unterscheiden sich in folgenden Angaben:

- initiale Steuerstabposition im aktiven Kernbereich (D1 bis D4-Bänke, E0-Bank und L-Bank, wobei die D5- und D6-Bänke in der L-Bank simuliert sind),
- Anfangskonzentration von Borsäuren im Primärkreis,

- Doppler-Kennlinie in D-BARE,
- Handsollwert L-Bank für die L-STABS Regelung und
- Referenzwerte für die Abschalt- und Abkühlborkonzentration (C_H, C_{HK}) in der Überwachung und Begrenzung der Einspeise-Borkonzentration.

Zur Steuerung der Simulation mit beiden Kernzuständen (BOC und EOC) wird dasselbe Ablaufprotokoll verwendet. In Tab. 9.1 und Tab. 9.2 sind die notwendigen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) aufgelistet.

Tab. 9.1 Datensatzeingriffe (Actions) im Fall Volllast für die zwei Kernzustände (BOC und EOC)

Maßnahme / Ereignis	Bedingung	Action	Parameter
Ende der Simulation	TEND	terminate	-

Tab. 9.2 Kontrollbedingungen (Trigger) im Fall Volllast für die zwei Kernzustände (BOC und EOC)

Bedingung	Beschreibung	Logik	Wert	Parameter
TEND	simulierte Zeit [s]	>=	600	-

9.2.2 Lastabwurf auf Eigenbedarf (EOC) (D2-07)

Diese Simulation wurde bereits für die Qualifikation der gekoppelten ATHLET/QUABOX-CUBBOX Version im BMU-Eigenforschungsvorhaben 4717R01334 durchgeführt. Eine ausführliche Beschreibung dieser Transiente ist im Kapitel 2.3.3 des Abschlussberichts /PAL 20/ zu finden. Ziel der Analyse ist die Verifizierung des Einflusses der Kernbedingungen im EOC.

Im Rahmen dieses Vorhabens wurde eine Spezifikationsdatei für die automatische Simulationsdurchführung erstellt, um sowohl die wesentlichen simulationsspezifischen Anfangs- und Randbedingungen zu implementieren als auch die Störung im Stromnetz zu initiieren.

In Tab. 9.3 und Tab. 9.4 sind dazu notwendigen und im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) aufgelistet.

Tab. 9.3 Datensatzeingriffe (Actions) im Fall Lastabwurf auf Eigenbedarf (Kernzustand: EOC)

Maßnahme / Ereignis	Bedingung	Action	Parameter
Verfügbarkeit 1. RESA Anregekrite-rium	T1	Down	-
Verfügbarkeit 2. RESA Anregekrite-rium	T1	Up	-
Störung des 400 kV Schalters	T2	Down	-
Auslösung des LAW	T2	Up	-
Verfügbarkeit Fremdnetz	T3	Up	-
Verfügbarkeit 10 kV Schiene BB	T3	Down	-
Verfügbarkeit Hauptkondensatpumpe	T3	Down	-
Verfügbarkeit Hauptspeisewasser-pumpe	T4	Down	-
Ende der Simulation	TEND	terminate	-

Tab. 9.4 Kontrollbedingungen (Trigger) im Fall Lastabwurf auf Eigenbedarf (Kernzustand: EOC)

Bedingung	Beschreibung	Logik	Wert	Parameter
T1	simulierte Zeit [s]	>=	599,0	-
T2	simulierte Zeit [s]	>=	600,5	-
T3	simulierte Zeit [s]	>=	601,06	-
T4	simulierte Zeit [s]	>=	602,15	-
TEND	simulierte Zeit [s]	>=	900,0	-

9.2.3 Fehlöffnen eines FD-Sicherheitsventils (EOC) (D2-01)

Bei dieser Transiente wird ein fehlerhaftes Auffahren des FD-Sicherheitsventils in der FD-Station im Dampferzeuger angenommen. Abb. 9.1 (links) zeigt den Massenstrom, der im Störfallverlauf durch das Ventil austritt. Die Auswertung der Ergebnisse der Transiente mit der gekoppelten ATHLET/QUABOX-CUBBOX Datensatzversion zeigt eine (erwartungsgemäße) Abweichung bei dem Verlauf der thermohydraulischen Parameter im Vergleich zu der ATHLET-Standalone-Rechnung mit dem Punktkinetik-Modell.

Der Parameter „Erlaubte Reaktorleistung“ wird ca. 19 s nach Transientenbeginn um 7 % abgesenkt und führt zu einer Absenkung des Generatorleistungssollwertes. Abb. 9.1 (rechts) sowie Abb. 9.2 (links) zeigen diese Absenkungen im Störfallverlauf. Diese Absenkungen werden bei der Durchführung der ATHLET-Standalone-Rechnung mit Punktkinetikmodell nicht errechnet. Dort verleibt die erlaubte Reaktorleistung auf 100 %.

Der Grund für dieses abweichende Verhalten findet sich bei der Auslösung des „STAFE-90“ Signals, welches in der STAFE-RELEB abgebildet wird und nach dem Steuerstabentwurf-Befehl (STEW) getriggert und als „Stab entklinkt“-Meldung angezeigt wird. Der Grenzwert 90 ist nur in der gekoppelten Version der Analysesimulators ATHLET/QUABOX-CUBBOX vorhanden, denn für die Abbildung des Signals ist die Messung der Leistungsverteilungswerte an den acht im Kern radial angeordneten Detektoren erforderlich, welche nur mit dem 3D-Kinetikmodell berechnet werden können.

Die Grenzwerte für die Auslösung der gestaffelten Maßnahmen der L-RELEB werden bei der Auslösung der Grenzwert 90 von STAFE-RELEB entsprechend abgesenkt. Dadurch kommt es bei der Rechnung nach dem Einwurf der D-10-Bank zum Ansprechen der L-RELEB und somit zum Einfahren der Steuerstäbe im Kern (siehe Abb. 9.2; rechts).

Des Weiteren erfolgt durch das Signal STAFE-RELEB 90 der Schnellangleich auf den Istwert der D-Bank-Stellung. Hierdurch werden nach dem Stabeinwurf Borsäureeinspeisungen durch die „D-BARE bei konstanter Leistung“ vermieden.

Durch den einsetzenden Xe-Aufbau beginnen die D-Bänke in Richtung des stationären Sollwertes auszufahren. Das STAFE 90-Signal aus der RELEB wird erst wieder zurückgesetzt, wenn die eingeworfenen Stabpaare wieder in ihrer D-Bank bzw. der zentrale Steuerstab E0 wieder in der L-Bank steht.

Mit dem Rücksetzen des STAFE-90-Signals ist der Schnellangleich wieder aufgehoben.

Die in Tab. 5.14 und Tab. 5.15 aufgelistete Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) für das transientenspezifischen Ablaufprotokoll gelten auch für die automatische Durchführung dieser gekoppelten ATHLET-QUABOX/CUBBOX Rechnung.

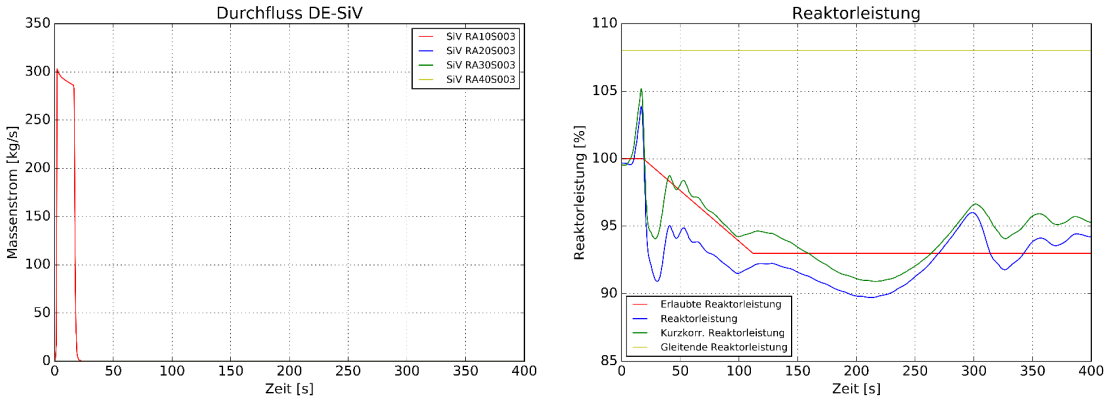

Abb. 9.1 Massenstrom im DE-SiV (links) und Reaktorleistung (rechts)

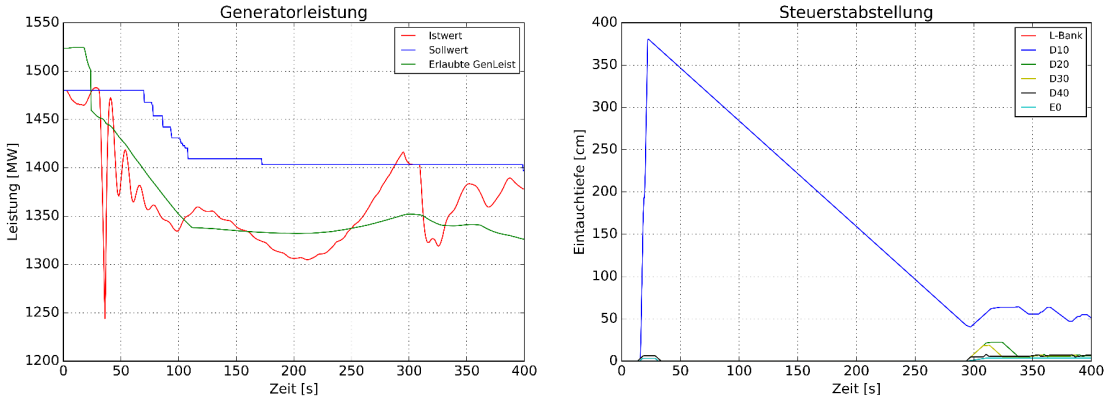

Abb. 9.2 Generatorleistung (links) und Steuerstabstellungen (rechts)

9.2.4 Auswurf des wirksamsten Steuerelements (BOC) (D3-16)

Dieses Ereignis wurde über den Auswurf von Steuerelementen aus der D1-Bank approximiert. Die Simulation dieses Ereignisses eignet sich aus folgenden Gründen besonders für die Verifizierung des gekoppelten ATHLET/QUABOX-CUBBOX:

- Prüfung der Übertragung von berechneten Kerndaten von Q/C an ATHLET während einer Transiente,
- Prüfung der implementierten Zuordnung von Steuerelementen mit den jeweiligen simulierten Kernkanälen (Mapping-Schema),
- Prüfung von RELEB Peak-Oben (PO) und -Unten (PU) Signalen sowie
- Prüfung der Querschnittbibliotheken in Q/C für diverse Zykluszustände.

Der Auswurf eines Steuerelementes ist nur denkbar, wenn man einen Bruch eines Steuerelementantriebsstutzens annimmt.

Durch den hohen Innendruck würde das Steuerelement aus dem Reaktorkern herausgeschleudert was zu einem plötzlichen Anstieg der Reaktivität führt. Der Betrag des Reaktivitätsanstieges ist abhängig von der Eintauchtiefe der Steuerelementbank, in der das ausgeworfene Steuerelement stand, und von der Anzahl der Steuerelemente, aus der sich diese Bank zusammensetzt.

Für diese Simulation wurde der Auswurf von zwei Stabpaaren aus der D1-Bank unterstellt. Der Steuerstabauswurf (auf Englisch REA – Rod Ejection Accident) wird ausgehend vom Teillastzustand wegen der simulierten Abfahrvorgänge und den damit verbundenen Aktionen für die Reduktion der thermischen Leistung durchgeführt, die eine umfassende Verifizierung von betrieblichen Systemen ermöglicht. Weiter ist in diesem Zustand wegen einer größeren Eintauchtiefen zu Transientenbeginn von einer höheren Reaktivitätszufuhr beim Auswurf der Steuerelemente auszugehen.

Die Anlage wird ausgehend vom Leistungsbetrieb und mit dem Kernzustand „Zyklusbeginn“ (BOC) auf ca. 80 % der Nennleistung mit einem Gradienten von 10 MW/min abgefahren. Durch die Leistungsreduktion am Generator sinkt der DH-Füllstand entsprechend der Sollwertführung in Abhängigkeit von dem Abfahrdiagramm sowie die KMT entsprechend der KMT-Regelung.

Die während des Abfahrvorgangs am tiefsten eingetauchten Steuerelemente (zwei Stabpaare aus D-Bank D1) werden mit einer angenommenen Geschwindigkeit von 39 m/s aus dem Kern ausgeworfen, wenn die Reaktorleistung den Schwellwert von ca. 80 % erreicht hat (Zeitpunkt t = 1.700 s nach Beginn des Abfahrens). Die Ergebnisse der Rechnung sind in Abb. 9.3 bis Abb. 9.5 dargestellt.

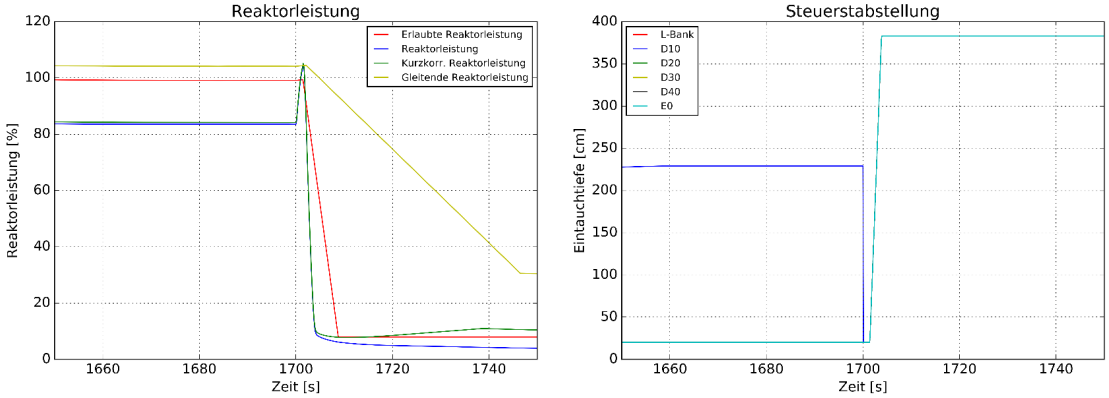

Abb. 9.3 Reaktorleistung (links) und Steuerstabstellung (rechts)

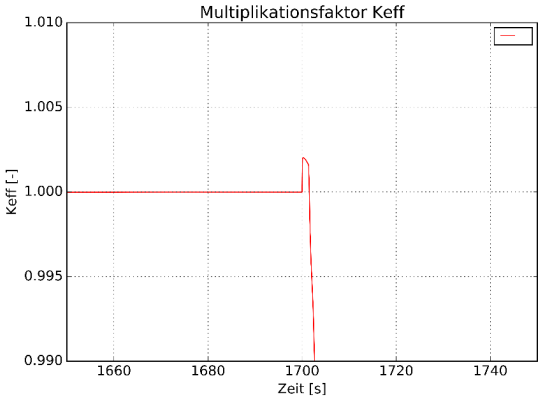

Abb. 9.4 Neutronenmultiplikationsfaktor K_{eff}

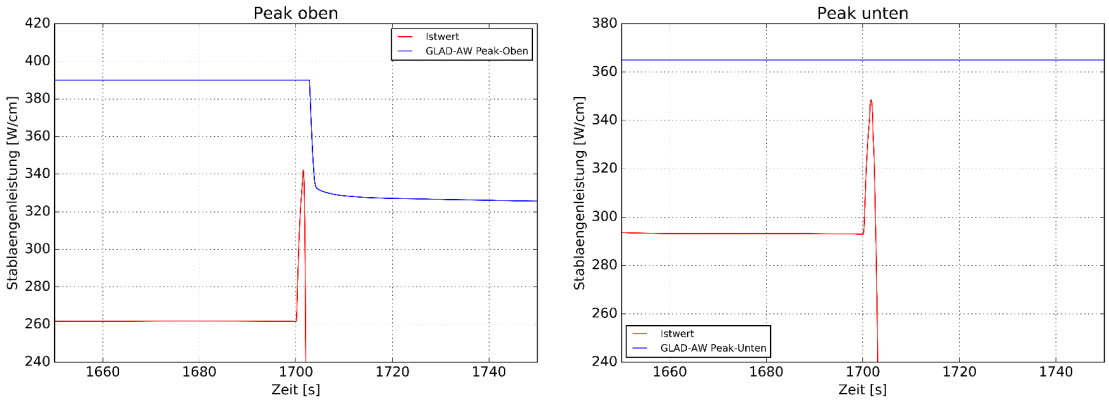

Abb. 9.5 Stablängenleistung mit „Peak oben“ (links) und „Peak unten“ (rechts)

Das D-Bank-Fahrprinzip sieht für Leistungsänderungen (wie z. B. bei Abfahren) vor, dass aus dem nahezu steuerstabfreien Volllastbetrieb die D-Bänke einzeln und nacheinander, nämlich in der von den Regelungseinrichtungen festgelegten Fahrfolge in den Reaktorkern eingefahren werden.

In der Simulation erreichen nach 1.700 s die Steuerelemente von D1-Bank einen Eintauchtiefe von 229 cm während sich die anderen Steuerelemente (D2, D3 und D4 bzw. L-Bank) im oberen Kernbereich bei einer Eintauchtiefe von 20 cm befinden. Der Auswurf von zwei Stabpaaren der D1-Bank bewirkt eine Änderung des Multiplikationsfaktor K_{eff} von ca. 0,2 %, was eine Reaktivitätszufuhr von ca. 0,34 % entspricht. Der darauffolgende Leistungsanstieg löst die RESA aus, sobald die kurzkorrigierte Reaktorleistung den gleitenden Grenzwert übersteigt.

Der berechnete maximale Wert der Stablängenleistung aus dem LVD beträgt nach dem Auswurf der Steuerstäbe ca. 350 W/cm, was unterhalb des maximalen erlaubten leistungsabhängigen Wertes (GLAD-Ansprechwert) von 365 W/cm liegt. Der GLAD-AW ist der zeitlich gleitende Ansprechwert der Leistungsdichte zur Begrenzung der betrieblichen BE-Belastung bei schneller Leistungsdichteerhöhung.

In Abb. 9.6 und Abb. 9.7 ist die Variation des von QUABOX/CUBBOX berechneten Neutronenflusses (schnell und thermisch) in Prozent ca. 1 s nach Eintritt der Transiente gezeigt. Der schnelle Neutronenfluss verdoppelt sich innerhalb von einer Sekunde in denjenigen BE, welche im Bereich der Position der D1-Bank liegen (siehe dazu die Steuerelemente-Position in Abb. 9.8).

Die für den automatisierten Ablauf der Transiente im Verifikationsverfahren notwendigen und im Ablaufprotokoll festgehaltenen Datensatzeingriffe (Actions) und Kontrollbedingungen (Trigger) sind in Tab. 9.5 und Tab. 9.6 aufgelistet.

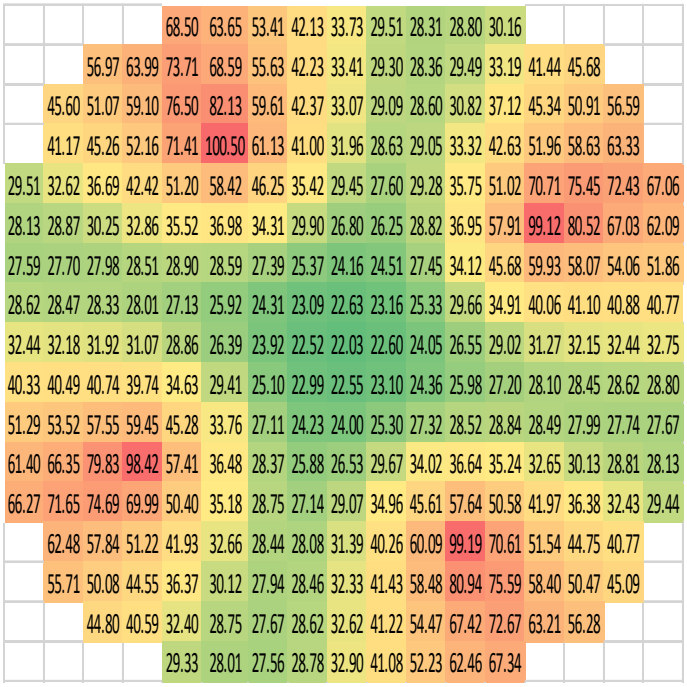

Abb. 9.6 Variation des schnellen Neutronenflusses in % für alle BE im Kern 1 s nach Eintritt der Transiente

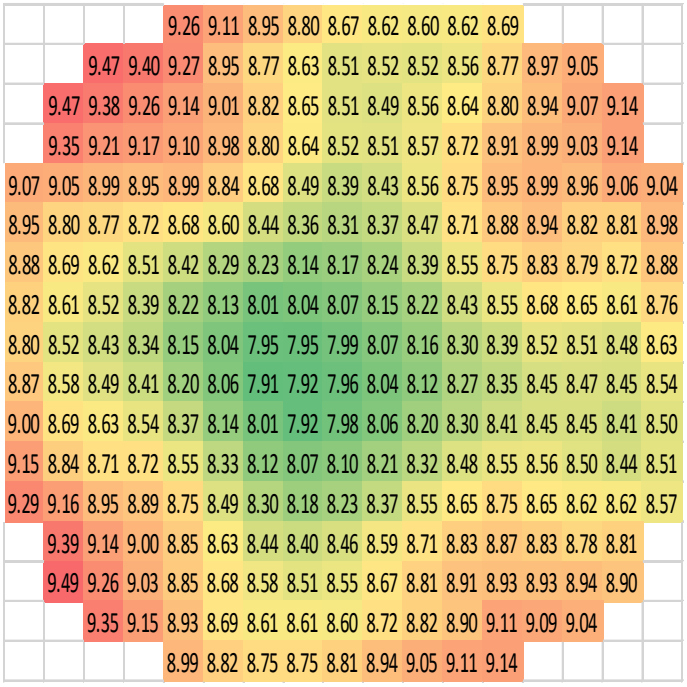

Abb. 9.7 Variation des thermischen Neutronenflusses in % für alle BE im Kern 1 s nach Eintritt der Transiente

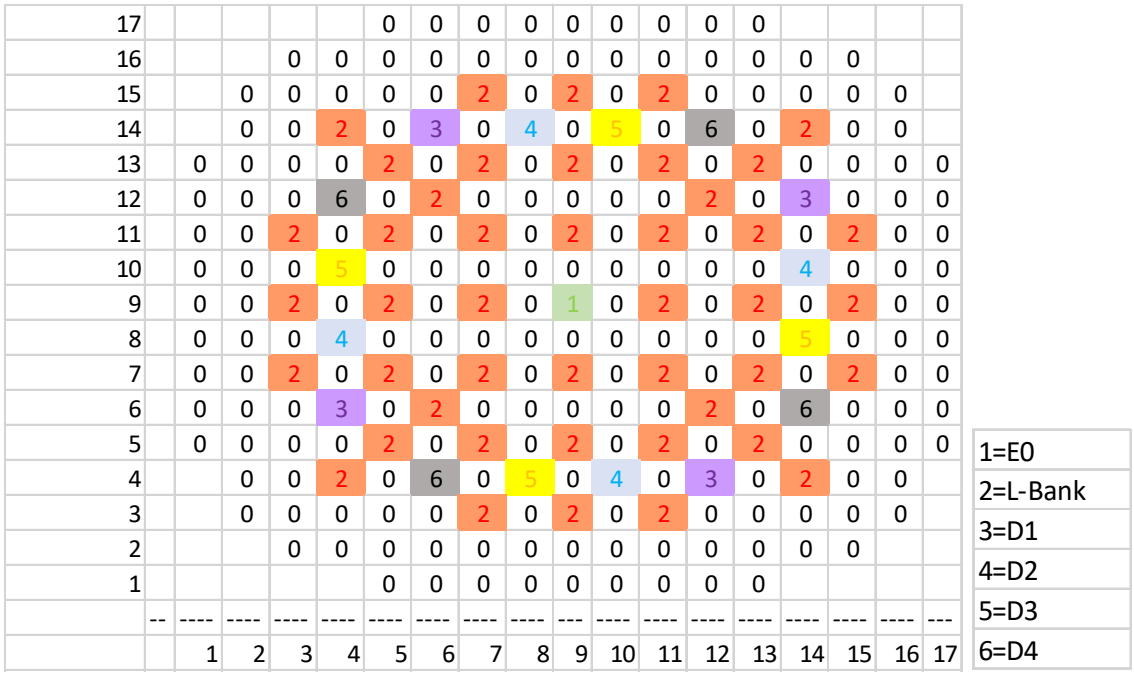

Abb. 9.8 Position der Steuerelemente im Kern

Tab. 9.5 Datensatzeingriffe (Actions) im Fall Auswurf der D1-Bank (Kernzustand: BOC)

Maßnahme / Ereignis	Bedingung	Action	Parameter
	T1	jump	0,01
Generatorleistung absenken: ▪ Generatorleistung absenken ▪ Leistungssollwert anpassen	T2	jump	10
	T2	jump	470
	T2	Up	-
	T2	Up	-
Auswurfgeschwindigkeit D11	T3	jump	-3.900
Auswurfgeschwindigkeit D12	T3	jump	-3.900
Handbefehl D11-Bank Hoch	T3	Up	-
Handbefehl D12-Bank Hoch	T3	Up	-
Handsteuerung D1-Bank	T3	Up	-
Ende der Simulation	TEND	terminate	-

Tab. 9.6 Kontrollbedingungen (Trigger) im Fall Auswurf der D1-Bank (Kernzustand: BOC)

Bedingung	Beschreibung	Logik	Wert	Parameter
T1	simulierte Zeit [s]	>=	2.298,0	-
T2	simulierte Zeit [s]	>=	600,0	-
T3	simulierte Zeit [s]	>=	2.300,0	-
TEND	simulierte Zeit [s]	>=	2.400,0	-

10 Zusammenfassung und Ausblick

Zur Sicherstellung der Aussagesicherheit des simulierten Anlagenverhaltens von in der GRS erstellten und verwendeten anlagenspezifische Analysesimulatoren, wurde ein automatisiertes Verifizierungsverfahren entwickelt. Hierbei werden Simulation im Rahmen eines kontinuierlichen Integrationsprozesses ausgeführt und ihre Ergebnisse anhand einer geeigneten Datenbasis automatisiert überprüft. Der Umfang der zur Verifikation notwendigen Analysen ist vom Detailgrad des zu verifizierenden Simulators abhängig und ist im Vorfeld der Verifizierung von den Datensatzentwicklern festzulegen. Das entwickelte Verifizierungsverfahren ist dabei derart flexibel gestaltet, dass bei Datensatzerweiterungen oder einem veränderten Kenntnisstand über die Referenzanlage der Umfang der in die Verifizierung einbezogenen Ereignissimulationen mit geringem Aufwand erweitert werden kann. Weiter ist das Verfahren derart generisch gestaltet, dass zusätzliche Analysesimulatoren mit überschaubarem Aufwand in die Verifizierungsprozedur einbezogen werden können.

Um eine automatisierte Durchführung von ATHLET-Simulationen unter Anwendung detaillierter anlagenspezifischer Analysesimulatoren zu ermöglichen, wurde im Rahmen des Vorhabens ein Steuerungsprogramm entwickelt. Dieses nutzt eine in ATHLET verfügbare Schnittstelle, um mit Hilfe der Programmiersprache Python in den Simulationsablauf einzugreifen. Dadurch können fallspezifische Abläufe und Tests für den zu überprüfenden Datensatz festgelegt werden. Mit Anwendung dieses ATHLET-Controllers werden Zustandsänderungen in der Anlage innerhalb des Simulationsablaufs umgesetzt, wie sie sich beispielsweise aus festgelegten Prozeduren der Betriebsdokumentation ergeben. Das Steuerprogramm greift zur Ablaufsteuerung auf eine einfache ereignisspezifische Spezifikationsdatei zu, in welcher Kontrollmaßnahmen und Randbedingungen festgelegt sind. Für die Erstellung dieser Spezifikationsdateien wurde eine eigene einfache und nutzerfreundliche Syntax auf Basis der Programmiersprache Python entwickelt.

Der erstellte ATHLET-Controller hat sich als effizientes und leicht zugängliches Werkzeug zur Durchführung komplexer Analysen erwiesen, dass über den Kontext des Vorhabens hinaus bereits breite Anwendung im Bereich DSA innerhalb der GRS findet.

Für die Durchführung automatisierter Simulationen und Ergebnisüberprüfungen im Rahmen eines automatisierten Verifizierungsverfahrens wurde auf das Prinzip der kontinuierlichen Integration zurückgegriffen. Das Verfahren wurde auf der GRS-internen

Software-Verwaltungsplattform Gitlab /CHO 20/ umgesetzt und verwendet hierbei entsprechende Speicherressourcen für Repositorys und Server zur kontinuierlichen Integration (CI). Die CI-Umgebung stellt Software zur Verfügung, welche die Kommunikation mit den Repositorys und einer bereitgestellten Speicherinfrastruktur für große Datenmengen aus den Simulationsergebnissen unterstützt. Sie ist in der Lage, komplex verschachtelte Aufgaben abzuarbeiten und verwendete Codes in der Vorbereitung eines Verifizierungslaufes zu kompilieren.

Grundsätzlich folgt die Verifizierung eines Datensatzes nach dem entwickelten Verfahren in seiner Umsetzung folgendem Schema: Entwickler erstellen einen Datensatz für einen Analysesimulator oder führen Wartungen oder Erweiterungen daran aus. Diese Arbeiten werden zentral in einem Git-Repository auf einem Gitlab-Server gespeichert. In einem Git-Repository auf demselben Server liegen datensatzspezifische Spezifikationsdateien (Ablaufprotokolle), welche der Steuerung des Datensatzes dienen und die Durchführung eines vordefinierten Satzes an Ereignissen ermöglichen. Ein erstelltes Softwarepaket zur Durchführung der automatisierten Simulationen und Ergebnistests in der CI-Umgebung des Gitlab-Servers sammelt notwenige Daten ein, breitet die Rechenläufe durch Softwarekompilierung und -zugriff vor und führt diese aus. Die Ergebnisse dieser Ereignissimulationen werden gegen zugehörige Grenzkurven abgeglichen, welche als Akzeptanzkorridore für Abweichungen im Systemverhalten dienen. Die Informationen zu diesen Grenzkurven sind ebenfalls in dem Git-Repository hinterlegt. Nach Ablauf der Tests werden alle zur Ergebnisauswertung und -reproduktion notwendigen Informationen sowie die Rechenergebnisse selbst in einer dafür bereitgestellten Speicherinfrastruktur abgelegt. Ein ebenfalls automatisch erstellter Testreport, welcher den Entwicklern zur Verfügung gestellt wird, erleichtert die Auswertung der und den Zugriff auf die Simulationsergebnisse.

Zur Ableitung von Grenzkurven für einen Datensatz ist zunächst eine geeignete Bewertungsbasis zu identifizieren. Diese kann beispielsweise auf Anlagendaten bzw. -dokumentationen, Betreibersimulation oder eigenen Erkenntnissen basieren. Für die Verifizierung relevante Ereignisse werden anschließend bestimmt und simuliert. Der Umfang der festzulegenden Ereignismatrix ist dabei insbesondere abhängig vom Detailgrad der Modellierung des Analysesimulators aber auch vom Umfang der zur Verfügung stehenden Bewertungsbasis. Die Ergebnisse der Ereignissimulationen werden in Form simulationsspezifischer Anlagenparameter mit der Bewertungsbasis abgeglichen. Etwaige Abweichungen werden bewertet und ggf. durch Datensatzanpassungen behoben. Die

finalen verifizierten Simulationsergebnisse bilden die erste Grundlage zur Erzeugung von Referenzkurven. Die Referenzkurven werden danach für die automatische Verifizierung gespeichert und für die Bestimmung der Grenzkurven verwendet. Die Grenzkurven selbst bilden konvexe Einhüllende mit einem Abstand von 5 % um die Referenzwertgrenzen. Um die Verifikationsbasis zu verbreitern und Tests robust zu gestalten, können Ergebnisse weiterer Verifikationssimulationen in ein Ensemble von Referenzkurven, individuell für jedes zu untersuchende Ereignis, einfließen.

Ziel sollte es auch zukünftig sein, die Ereignismatrix derart zu optimieren, dass die Anzahl der einbezogenen Ereignisse so klein wie möglich aber so groß wie nötig ist, um das Verhalten aller relevanten Systeme im Analysesimulator in allen Betriebsphasen zu überprüfen. Dazu ist es notwendig, die angeforderten und beeinflussten Systeme im jeweiligen Ereignis zu identifizieren und ihr Verhalten zu bewerten.

Die entwickelte Vorgehensweise zur automatischen Verifizierung von Datensätzen wurde exemplarisch angewendet, um die entwickelten Prozeduren und den gesamten automatisierten Verifikationsprozess zu prüfen. Dazu wurde zunächst der Datensatz eines Vorkonvoi-Analysesimulators einbezogen und eine Ereignismatrix entsprechend den Ergebnissen aus dem vorangegangenen Vorhabens 4715R01345 /PAL 18/ umgesetzt. Im Verlauf der Umsetzung des automatisierten Verifizierungsverfahrens und seiner Anwendung an dem Beispieldatensatz, wurden eine Reihe von Optimierungen, Anpassungen und Fehlerkorrekturen im Analysesimulator vorgenommen. Um die Übertragbarkeit und allgemeine Anwendbarkeit der entwickelten automatischen Verifizierungsprozedur für in der GRS erstellte Analysesimulatoren zu demonstrieren, wurde außerdem eine exemplarische Anwendung an einer Auswahl von Ereignissen mit einem Konvoi-Datensatz erfolgreich umgesetzt.

Im Zuge der Arbeiten konnte der Nutzen und die Funktionalität des entwickelten automatischen Verifizierungsverfahrens an den verwendeten Referenzdatensätzen (Vorkonvoi/Konvoi) demonstriert werden. Um Aussagen zu einigen speziellen Störfallszenarien treffen zu können, kann es aber notwendig sein, zusätzliche Kopplungen mit externen Codes anzuwenden, beispielsweise um Prozesse abzubilden, welche stark von 3D-neutronenpysikalische Phänomenen beeinflusst sind. Aus diesem Grund wurde das Verfahren auf eine komplexere gekoppelte Version (thermohydraulisch/3D-neutronenphysikalisch) des Vorkonvoi-Analysesimulators mit den Codes ATHLET/QUABOX-CUBBOX ausgedehnt. Die Durchführung einer gekoppelten thermohydraulischen und 3D-Neutronenphysikalischen Analyse mit den Codes ATHLET/QUABOX-CUBBOX bietet die

Möglichkeit, den für die Kopplung entwickelten Datensatz des Vorkonvoi-Analysesimulators abdeckend zu verifizieren. Die Integration der gekoppelten Version des Analysesimulators in das automatische Verifizierungsverfahren ermöglicht bei der Anwendung außerdem die Identifizierung von möglichem Weiterentwicklungsbedarf der gekoppelten Version des Analysesimulators im Bereich der Begrenzungen, der Neutronenflussmessung und der Leistungsverteilung im Kern.

Im Verlauf des Vorhabens zeigte sich, dass es häufiger zur Verletzung vordefinierter Grenzkurven kam, wobei abrupt ein Übergang von einem Anlagenzustand zu einem anderen infolge kleiner Änderungen in bestimmten Anlagenparametern erfolgte. Weiter zeigte sich, dass Anlagenparameter, welche im Störfallverlauf sehr steile Gradienten mit kurzer Dauer (z. B. Druckspitzen) aufweisen, aufgrund numerischer Ungenauigkeiten oder begrenzter Ausgabefrequenz ebenfalls häufig Verletzungen der Grenzkurven verursachen. Dieses Problem führt zwar zu einem erhöhten Prüfaufwand durch die Entwickler, da diese durch den Meldegenerator über derartige Verletzungen informiert werden, kann aber durch regelmäßige Anwendung der automatischen Verifizierung und Integration entsprechender Ergebnisverläufe in das Referenzkurvenensemble graduell reduziert werden.

Das entwickelte automatisierte Verifizierungsverfahren hat sich als robust und flexibel erwiesen und führt zu einer starken Steigerung der Effizienz und Effektivität bei der Durchführung von Datensatzverifizierungen. Es ist aus Sicht der GRS anzustreben, alle in Anwendung befindlichen Datensätze zukünftig in das Verfahren zu integrieren, um die Aussagesicherheit von simuliertem Anlagenverhaltens beizubehalten und ihre Belastbarkeit weiter zu erhöhen.

Literaturverzeichnis

/ARE 19/ Arefeen, M. S., Schiller, M., Continuous Integration Using Gitlab. Undergraduate Research in Natural and Clinical Science and Technology Journal, 3, 1-6, 2019.

/AUS 21/ Austregesilo, H., et al.: ATHLET 3.3 User's Manual, GRS, GRS-P-1 / Vol. 1, Rev. 9, 2021.

/CHO 20/ Choudhury, P., et al., GitLab: work where you want, when you want. Journal of Organization Design, 9(1), 1-17, 2020.

/DAU 04/ D'Auria, F. et. al., CRISSUE-S – WP2 Neutronics/Thermal-hydraulics Coupling in LWR Technology: State-of-the-art Report, OECD/NEA Report No. 5436, ISBM 92-64-02084-5, Paris, France, 2004.

/GRA 72/ Graham, R. L., An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set; Information Processing Letters Vol. 1: pp 132–133. DOI 10.1016/0020-0190(72)90045-2, 1972.

/GRS 06/ Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, Unterstützung der Bundesaufsicht bei thermohydraulischen Fragestellungen, Auftrags-Nr. 857001, GRS-A-3367, Dezember 2006.

/HER 18/ Herb, J., A continuous integration platform for the deterministic safety analyses code system AC², Proceedings of the 26th ICONE, ICONE26-81123, July 22-26, 2018, London, England.

/IAE 09/ Deterministic safety analysis for nuclear power plants, IAEA Safety Standards Series No. SSG-2, IAEA Wien 2009.

/JAC 21/ Jacht, V., Scheuer, J., Schöffel, P., Wielenberg, A.: ATHLET 3.3 Programmer's Manual, GRS, GRS-P-1 / Vol. 2, Rev. 9, 2021.

/KSG 07/ KSG/GfS Schulungsunterlagen – Fachtheorieschulung – Niveaumessung, Rev.0, 15.09.2007.

/OKK 17/ Okken, B., Python Testing with pytest; Pragmatic Bookshelf, 2017.

/PAL 17/ Palazzo, S., Fortschreibung des Standes von Wissenschaft und Technik bei der Durchführung und Bewertung von Störfallanalysen und der Verwendung von Analysesimulatoren, BMU-Vorhaben 4717R01334.

/PAL 18/ Palazzo, S. et. al., Verifizierung von Analysesimulatoren, AP1 des Eigenforschungsvorhaben „Vertiefte und generische Auswertung von Betriebserfahrungen – neue Bewertungsmethoden und Störfallanalyseverfahren“, BMU-Vorhaben 4715R01345, GRS-A-3920, März 2018.

/PAL 20/ Palazzo, S. et al., Fortschreibung des Standes von W&T bei der Durchführung und Bewertung von Störfallanalysen und der Verwendung von Analysesimulatoren, GRS-A-3991, Abschlussbericht, März 2020.

/PAT 17/ Pathania, N., Pro Continuous delivery: with Jenkins 2.0. Apress, New York, NY, 2017.

/PET 08/ Petruzzi, A. and D’Auria, F., Thermal-Hydraulic system codes in nuclear reactor safety and qualification procedures, Science and Technology of Nuclear Installation, Article ID 460795, doi:10.1155/2008/460795, Volume 2008.

/POI 17/ Pointner, W., et al., Ermittlung des Standes von Wissenschaft und Technik bei der Durchführung und Bewertung von Störfallanalysen und der Verwendung von Analysesimulatoren, Abschlussbericht GRS-462, ISBN 978-3-946607-45-8, März 2017.

/SCH 18/ Schöffel, P. et. al., Rechenmethodenentwicklung für Reaktorsicherheitsanalysen mit dem Systemcode ATHLET, Abschlussbericht GRS-497, ISBN 978-3-946607-82-3, Juli 2018.

/SIA 15/ Sicherheitsanforderungen an Kernkraftwerke
22. November 2012, Neufassung vom 3. März 2015.

/SMA 11/ Smart, J., Jenkins: The definitive guide, O’Reilly Media, Sebastopol, CA, 2011.

/VAT 92/ Vatti, B. R.; A generic solution to polygon clipping; Communications of the ACM 35, pp 56-63; 1992.

/ZOR 21/ Zorkator, Fortran Development Extensions (libfde), https://github.com/Zorkator/libfde, 2021.

Abbildungsverzeichnis

Abb. 3.1 Komponenten und Steuerungsschema des ATHLET-Controllers 13

Abb. 4.1 Konzept der automatischen Verifizierung von Analysesimulatoren 27

Abb. 4.2 Schematische Darstellung der programmtechnischen Umsetzung der Verifizierungsprozedur in Gitlab-CI 29

Abb. 4.3 Speicherstruktur zur Ablage der Verifizierungsergebnisse 35

Abb. 6.1 Beispiel für ein Ensemble an Referenzkurven als Bewertungsbasis zur automatischen Verifikation 63

Abb. 6.2 Vergleich eines aktuellen Verifikationslaufergebnisses mit Grenzkurven und dem vorangegangenen Verifikationslaufergebnis 64

Abb. 6.3 Verletzung einer Grenzkurve infolge von Änderungen im Datensatz 65

Abb. 7.1 Teil I und II des Test-Reports mit Informationen zu den durchgeführten Pipelines und Jobs auf Giltlab 68

Abb. 7.2 Teil III des Test-Reports mit allen Informationen zu den Ergebnissen der einzelnen Ereignissimulationen 70

Abb. 7.3 Teil IV des Test-Reports mit Informationen zu allen verwendeten Software-Versionen 71

Abb. 8.1 Prinzip der Füllstandsmessung durch Differenzdruck bei siedendem Wasser 76

Abb. 8.2 Sprung in der Generatorleistung (PG-Ist) mit Verlassen des Akzeptanzkorridors in der automatischen Verifikation von D2-09_HspW_Pumpen 78

Abb. 8.3 Druckoszillationen im Primärkreis (links) und angepasste Grenzkurven mit aktualisiertem Verifikationsergebnis (rechts) 79

Abb. 8.4 Oszillationen der Kühlmitteltemperatur (links) und unerwartetes Verfahren der Steuerstäbe der D-Bank (rechts) 81

Abb. 8.5 Ordnerstruktur im Verify-Repository zur Einbeziehung zusätzlicher Analysesimulatoren in die automatische Verifikationsprozedur 85

Abb. 8.6 Vergleich des Druckhalter-Sprühmassenstroms (links) und des Primärdruckes (rechts) vor und nach der Anpassung im Konvoi-Datensatz 89

Abb. 9.1 Massenstrom im DE-SiV (links) und Reaktorleistung (rechts) 97

Abb. 9.2 Generatorleistung (links) und Steuerstabstellungen (rechts) 97

Abb. 9.3 Reaktorleistung (links) und Steuerstabstellung (rechts) 99

Abb. 9.4 Neutronenmultiplikationsfaktor K_{eff} 99

Abb. 9.5 Stablängenleistung mit „Peak oben“ (links) und „Peak unten“ (rechts) 99

Abb. 9.6 Variation des schnellen Neutronenflusses in % für alle BE im Kern 1 s nach Eintritt der Transiente 101

Abb. 9.7 Variation des thermischen Neutronenflusses in % für alle BE im Kern 1 s nach Eintritt der Transiente 101

Abb. 9.8 Position der Steuerelemente im Kern 102

Tabellenverzeichnis

Tab. 3.1 Vordefinierte Steuerungsbefehle und zugehörige Optionen 21

Tab. 3.2 Erläuterungen zu verwendbaren Optionen des Startbefehls 23

Tab. 5.1 Ereignisliste für die Verifizierung eines Vorkonvoi-Analysesimulators nach /PAL 18/ 41

Tab. 5.2 Datensatzeingriffe (Actions) im Fall Volllast 44

Tab. 5.3 Kontrollbedingungen (Trigger) im Fall Volllast 45

Tab. 5.4 Datensatzeingriffe (Actions) im Fall D3-30_DH_SiVentil 46

Tab. 5.5 Kontrollbedingungen (Trigger) im Fall D3-30_DH_SiVentil 46

Tab. 5.6 Datensatzeingriffe (Actions) im Fall D2-14_HD_Reduzier_AUF 47

Tab. 5.7 Kontrollbedingungen (Trigger) im Fall D2-14_HD_Reduzier_AUF 47

Tab. 5.8 Datensatzeingriffe (Actions) im Fall D2-16_HD_Reduzier_ZU 48

Tab. 5.9 Kontrollbedingungen (Trigger) im Fall D2-16_HD_Reduzier_ZU 48

Tab. 5.10 Datensatzeingriffe (Actions) im Fall D2-21_KMT 48

Tab. 5.11 Kontrollbedingungen (Trigger) im Fall D2-21_KMT 48

Tab. 5.12 Datensatzeingriffe (Actions) im Fall D2-24_Deionat 49

Tab. 5.13 Kontrollbedingungen (Trigger) im Fall D2-24_Deionat 50

Tab. 5.14 Datensatzeingriffe (Actions) im Fall D2-01_FD_SiVentil 50

Tab. 5.15 Kontrollbedingungen (Trigger) im Fall D2-01_FD_SiVentil 50

Tab. 5.16 Datensatzeingriffe (Actions) im Fall D2-07_LAW-MAN 51

Tab. 5.17 Kontrollbedingungen (Trigger) im Fall D2-07_LAW-MAN 51

Tab. 5.18 Datensatzeingriffe (Actions) im Fall D2-09_HspW_Pumpen 52

Tab. 5.19 Kontrollbedingungen (Trigger) im Fall D2-09_HspW_Pumpen 52

Tab. 5.20 Datensatzeingriffe (Actions) im Fall D2-02-06_HWSenke 53

Tab. 5.21 Kontrollbedingungen (Trigger) im Fall D2-02-06_HWSenke 53

Tab. 5.22 Datensatzeingriffe (Actions) im Fall D2-10_PUMA1v4 53

Tab. 5.23 Kontrollbedingungen (Trigger) im Fall D2-10_PUMA1v4 54

Tab. 5.24 Datensatzeingriffe (Actions) im Fall D2-28_Notstrom 54

Tab. 5.25 Kontrollbedingungen (Trigger) im Fall D2-28_Notstrom 55

Tab. 5.26 Datensatzeingriffe (Actions) im Fall D3-31_DE_Heizrohr 56

Tab. 5.27 Kontrollbedingungen (Trigger) im Fall D3-31_DE_Heizrohr 56

Tab. 5.28 Datensatzeingriffe (Actions) im Fall D3-23_KMV_01F 57

Tab. 5.29 Kontrollbedingungen (Trigger) im Fall D3-23_KMV_01F 57

Tab. 5.30 Datensatzeingriffe (Actions) im Fall D3-30_DH_SiVentil 58

Tab. 5.31 Kontrollbedingungen (Trigger) im Fall D3-30_DH_SiVentil 58

Tab. 5.32 Datensatzeingriffe (Actions) im Fall D3-05_FD_Leck 59

Tab. 5.33 Kontrollbedingungen (Trigger) im Fall D3-05_FD_Leck 59

Tab. 5.34 Datensatzeingriffe (Actions) im Fall D4a-04_ATWS 60

Tab. 5.35 Kontrollbedingungen (Trigger) im Fall D4a-04_ATWS 60

Tab. 8.1 Datensatzeingriffe (Actions) im Fall D3-31_DE_Heizrohr 86

Tab. 8.2 Kontrollbedingungen (Trigger) im Fall D3-31_DE_Heizrohr 86

Tab. 8.3 Datensatzeingriffe (Actions) im Fall D3-30_DH_SiVentil 87

Tab. 8.4 Kontrollbedingungen (Trigger) im Fall D3-30_DH_SiVentil 88

Tab. 9.1 Datensatzeingriffe (Actions) im Fall Volllast für die zwei Kernzustände (BOC und EOC) 94

Tab. 9.2 Kontrollbedingungen (Trigger) im Fall Volllast für die zwei Kernzustände (BOC und EOC) 94

Tab. 9.3 Datensatzeingriffe (Actions) im Fall Lastabwurf auf Eigenbedarf (Kernzustand: EOC) 95

Tab. 9.4 Kontrollbedingungen (Trigger) im Fall Lastabwurf auf Eigenbedarf (Kernzustand: EOC) 95

Tab. 9.5 Datensatzeingriffe (Actions) im Fall Auswurf der D1-Bank (Kernzustand: BOC) 102

Tab. 9.6 Kontrollbedingungen (Trigger) im Fall Auswurf der D1-Bank (Kernzustand: BOC) 102

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 978-3-949088-70-4