IAS-Forschungsberichte

Forschungsbericht
Institut für Automatisierungs- und Softwaretechnik

Hrsg.: Prof. Dr.-Ing. Dr. h. c. P. Göhner

Friedemann Bitsch

Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme in Temporallogik

Band 1/2007

Universität Stuttgart

Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme in Temporallogik

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Friedemann Bitsch
aus Konstanz

Hauptberichter:	Prof. Dr.-Ing. Dr. h. c. Peter Göhner
Mitberichter:	Prof. Dr.-Ing. Dr. h. c. Eckehard Schnieder

Tag der Einreichung:	13.04.2006
Tag der mündlichen Prüfung:	17.11.2006

Institut für Automatisierungs- und Softwaretechnik
der Universität Stuttgart

2006

IAS-Forschungsberichte

Band 1/2007

Friedemann Bitsch

Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme in Temporallogik

D 93 (Diss. Universität Stuttgart)

Shaker Verlag
Aachen 2007

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: Stuttgart, Univ., Diss., 2006

Copyright Shaker Verlag 2007
Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8322-6035-4
ISSN 1610-4781

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen
Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Automatisierungs- und Softwaretechnik (IAS) der Universität Stuttgart.

Mein ausgesprochener Dank gilt dem Leiter des Instituts, Herrn Prof. Dr.-Ing. Dr. h. c. Peter Göhner, der die Bearbeitung dieses Promotionsthemas ermöglicht und gefördert hat. Insbesondere danke ich ihm herzlich für die fortwährende Begleitung meiner Arbeit durch unzählige intensive, lehrreiche und wertvolle Diskussionen und Anregungen.

Herrn Prof. Dr.-Ing. Dr. h. c. Eckehard Schnieder danke ich für das Interesse an meiner Arbeit, die freundliche Bereitschaft zur Übernahme des Mitberichts und die hilfreichen Hinweise bereits während der Bearbeitung des Forschungsthemas.

Der Deutschen Forschungsgemeinschaft bin ich dankbar für die Förderung des Forschungsprojekts *SafeRail* mit dem Thema *Integration von Methoden zur Spezifikation und Verifikation von Sicherungseinrichtungen im spurgeführten Verkehr*, aus dem heraus die vorliegende Arbeit entstanden ist. Dr.-Ing. Saeid Arabestani als Projektmitarbeiter sage ich herzlich Dank für die intensiven und fruchtbaren Diskussionen. Ebenso danke ich Paul Linder, Michael Wedel und Dr.-Ing. Jan-Tecker Gayen für die kritischen und ergiebigen Diskussionen sowie für die sorgfältige Durchsicht des Manuskripts.

Allen Kolleginnen und Kollegen des IAS sei an dieser Stelle für die angenehme Zusammenarbeit gedankt. Jeder von ihnen hat auf seine eigene Weise zum Gelingen dieser Arbeit beigetragen. Danken möchte ich außerdem allen betreuten Studierenden. Sie waren mir bei der praktischen Umsetzung meiner Ideen eine große Hilfe.

Sehr dankbar bin ich darüber hinaus meinem bereits verstorbenem Vater und meiner Mutter, Gerhard und Lore Bitsch, die mich immer in jeder Hinsicht vertrauensvoll förderten, sowie meinen Schwiegereltern, Eugen und Ingerose Ankele, für die Unterstützung in den Jahren der Promotion.

Besonders bedanke ich mich bei meiner Frau Birgit für ihr Einfühlungsvermögen, ihre Nachsicht, allen Ansporn, das Korrekturlesen und ihre Liebe, die entscheidend zum Gelingen der Arbeiten beigetragen hat. Ihr widme ich diese Arbeit.

Stuttgart, im Februar 2007

Friedemann Bitsch

Inhaltsverzeichnis

Abbildungsverzeichnis ..... **v**

Tabellenverzeichnis ..... **vii**

Abkürzungsverzeichnis ..... **viii**

Begriffsverzeichnis ..... **x**

Zusammenfassung ..... **xiii**

Abstract ..... **xiv**

1 Einleitung und Motivation **1**

1.1 Bedeutung von Sicherheitsanforderungen für die Entwicklung sicherer Automatisierungssysteme 1

1.2 Zielsetzung 2

1.3 Gliederung der Arbeit 3

2 Grundlagen **4**

2.1 Charakteristische Merkmale von Automatisierungssystemen 4

2.2 Sicherheit von Automatisierungssystemen 5

2.2.1 Sicherheit 5

2.2.2 Sicherheitsnachweis 6

2.2.3 Sicherheitsanforderungen 7

2.2.4 Aufstellung und Formulierung funktionaler Sicherheitsanforderungen 9

2.2.5 Vorgaben und Empfehlungen der Normen zur Formulierung funktionaler Sicherheitsanforderungen 10

2.3 Präzise Spezifikation von Anforderungen 11

2.3.1 Möglichkeiten zur Spezifikation 11

2.3.2 Spezifikation in Normsprachen 13

2.3.3 Formale Spezifikation 14

2.4 Einführung in die formale Spezifikation und Verifikation von Anforderungen 14

2.4.1 Mathematische Logik 14

2.4.2 Formale Verifikation 18

3 Aufstellung der Konzept-Anforderungen für ein Verfahren zur anwendbaren Spezifikation von Sicherheitsanforderungen **21**

3.1 Formale Spezifikation und Verifikation von Sicherheitsanforderungen für Automatisierungssysteme 21

3.2 Anwendbare formale Spezifikation und Verifikation 23

4 Stand der Forschung für die anwendbare formale Spezifikation von Anforderungen in Temporallogik **26**

4.1 Schwierigkeiten bei der formalen Spezifikation mit Temporallogik 26

4.2 Ansätze zur anwendbaren formal verifizierbaren Spezifikation von Anforderungen ... 27
4.2.1 Verwendung von Textbausteinen in einer Terminologie der natürlichen Sprache ... 28
4.2.2 Verwendung von Spezifikationssprachen in textueller Notation ... 31
4.2.3 Verwendung von Spezifikationssprachen in grafischer Notation ... 33
4.2.4 Werkzeugführung zur direkten Spezifikation in Temporallogik ... 37
4.2.5 Verwendung von temporallogischen Formeln ... 38
4.3 Zusammenfassende Bewertung der vorgestellten Ansätze und Schlussfolgerungen für ein geeignetes Verfahren ... 40
4.3.1 Kriterien und Bewertung ... 40
4.3.2 Zusammenfassender Vergleich der Ansatzarten ... 43
4.3.3 Schlussfolgerungen für ein Verfahren zur anwendbaren Spezifikation von Sicherheitsanforderungen ... 44

5 Anwendbare formale Spezifikation von Sicherheitsanforderungen durch Wiederverwendung ... 46
5.1 Das adäquate Beschreibungsmittel zur formal verifizierbaren Spezifikation von Sicherheitsanforderungen für Automatisierungssysteme ... 46
5.2 Vermittlung von Expertenwissen für die Spezifikation ... 48
5.2.1 Analyse von Unterstützungssystemen ... 49
5.2.2 Analyse von Techniken zur Wiederverwendung bewährter Lösungen ... 49
5.3 Formale Spezifikation von Sicherheitsanforderungen mit Safety-Pattern ... 54
5.3.1 Verwendung von Safety-Pattern zur Spezifikation von Sicherheitsanforderungen ... 54
5.3.2 Bedeutung der eindeutigen Charakterisierung der Safety-Pattern für die Anwendbarkeit ... 56
5.3.3 Bedeutung der Vollständigkeit des Safety-Pattern-Katalogs für die Anwendbarkeit ... 57
5.3.4 Vermittlung von Expertenwissen durch generische Formulierungen ... 58
5.3.5 Instanziierung von Sicherheitsanforderungen ... 62
5.3.6 Interpretation von Sicherheitsanforderungen ... 63
5.3.7 Erläuterung der Safety-Pattern zur Unterstützung der klaren Verständlichkeit von Safety-Pattern ... 64
5.3.8 Anleitung zur Anwendung durch Musterbeispiele ... 66
5.3.9 Bedeutung der Werkzeugunterstützung für die Anwendbarkeit ... 66
5.4 Zusammenfassung des Kapitels ... 68

6 Entwicklung des Safety-Pattern-Konzepts ... 69
6.1 Bestimmung der vollständigen Safety-Pattern-Menge ... 69
6.1.1 Vorgehen zur Ermittlung der Safety-Pattern-Menge ... 69
6.1.2 Bestimmung der Menge der notwendigen Formulierungen für das Safety-Pattern-Konzept ... 70
6.1.3 Bestimmung der Menge der nützlichen generischen Formulierungen für die Anwendbarkeit des Safety-Pattern-Konzepts ... 81
6.1.4 Klassifikation der Safety-Pattern und die resultierende Safety-Pattern-Menge ... 84
6.2 Entwicklung der Safety-Pattern-Normsprache ... 86

6.3 Entwicklung einer geeigneten grafischen Notation zur Spezifikation mit Safety-Pattern ... 90
6.4 Entwicklung von grafischen Darstellungen zur Erläuterung der Safety-Pattern ... 92
6.4.1 Verbreitete Darstellungsweisen ... 92
6.4.2 Visualisierung im Ausschnitt eines beliebigen Zustandsraums ... 93
6.4.3 Erläuterung mithilfe von Zeitdiagrammen ... 95
6.5 Erläuterung der Safety-Pattern in natürlicher Sprache ... 98
6.6 Anleitung zur Anwendung durch Musterbeispiele ... 99

7 Realisierung eines Web-basierten Unterstützungssystems zur Spezifikation von Sicherheitsanforderungen ... 100
7.1 Grundlegende Entwicklungsentscheidungen ... 100
7.2 Speicherung der Safety-Pattern in einer Datenbank ... 101
7.3 Beratungssystem für die Selektion und Darstellung von Safety-Pattern ... 101
7.3.1 Funktionalität für die Safety-Pattern-Selektion und -Darstellung ... 101
7.3.2 Unterstützung der Interpretation von Safety-Pattern durch grafische Simulation ... 106
7.4 Beratungssystem für die Instanziierung ... 107
7.5 Verwaltung von instanziierten Sicherheitsanforderungen ... 109
7.6 Integration der Teilsysteme von SAPIS ... 109

8 Anwendung des Safety-Pattern-Konzepts für einen eingleisigen Bahnübergang im Funk-Fahr-Betrieb ... 111
8.1 Anforderung an eine geeignete Anwendung zur Erprobung und Demonstration ... 111
8.2 Modellprozess „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb“ ... 112
8.2.1 Beschreibung des betrieblichen Konzepts ... 113
8.2.2 Aufbau des Modellprozesses ... 114
8.3 Steuerungsmodell ... 116
8.4 Spezifikation der Sicherheitsanforderungen ... 116
8.5 Erprobung des Systemverhaltens in sicherheitskritischen Situationen ... 119
8.6 Erfahrungen und Erkenntnisse ... 121

9 Schlussbetrachtungen und Ausblick ... 125
9.1 Zusammenfassung der Ergebnisse und Bewertung ... 125
9.2 Ausblick ... 127

Literaturverzeichnis ... 128

Anhang ... 142

A Syntax von CTL und Umformungsregeln ... 142

B Ausschnitt aus der Safety-Pattern-Klassifikation ... 143

C Gegenbeispiel für die automatische Generierung von Formulierungen in der Safety-Pattern-Normsprache ... 153

D Grafische Notation zur Spezifikation mit Safety-Pattern ... 155

E Die Symbole des Simulations-Konzepts und zusätzliche Informationen zur Konzept-Umsetzung 159

E.1 Übersicht über die Symbole des Simulations-Konzepts 159

E.2 Umsetzung des Simulations-Konzepts zur Unterstützung der Interpretation von Safety-Pattern 160

E.3 Umsetzung des Simulations-Konzepts für Safety-Pattern-Kompositionen 162

F Beispiel eines Safety-Pattern 165

G Erörterung der Generierung der nützlichen generischen Formulierungen in SAPIS 167

H Beispiel der Anzeige eines Safety-Pattern durch SAPIS 171

I Beispiel für die Selektion eines Safety-Pattern 173

Abbildungsverzeichnis

Abbildung 2.1: Struktur eines Prozessautomatisierungssystems ... 5
Abbildung 2.2: Berechnungsbaum eines Zustandsmodells ... 17
Abbildung 4.1: Spezifikation der Referenzanforderung mit strukturierten Textbausteinen ... 28
Abbildung 4.2: Spezifikation der Referenzanforderung mit STD ... 34
Abbildung 4.3: Spezifikation der Referenzanforderung mit LSC ... 35
Abbildung 4.4: Spezifikation der Referenzanforderung mit MSC_{CTL} ... 36
Abbildung 5.1: Komposition von Softwaresystemen aus Softwarekomponenten [Göhn98] ... 50
Abbildung 5.2: Grundschema der Spezifikation mithilfe von Safety-Pattern ... 55
Abbildung 5.3: Klassifikationsbaum zur Safety-Pattern-Selektion ... 57
Abbildung 5.4: Grundprinzip des Safety-Pattern-Konzepts ... 68
Abbildung 6.1: Veranschaulichung von (6.1) „p muss in allen Systemzuständen gelten“ ... 76
Abbildung 6.2: Veranschaulichung von (6.2) „von jedem Systemzustand aus muss p in allen Systemzuständen gültig sein“ ... 76
Abbildung 6.3: Veranschaulichung von (6.2) „von jedem Systemzustand aus muss p in allen Systemzuständen gültig sein“ mit Auflösung von „G p“ ... 76
Abbildung 6.4: Erste mögliche Erfüllung von (6.3) ... 78
Abbildung 6.5: Zweite mögliche Erfüllung von (6.3) ... 78
Abbildung 6.6: Dritte mögliche Erfüllung von (6.3) ... 78
Abbildung 6.7: Darstellung von Safety-Pattern-Klassifikationskriterien ... 85
Abbildung 6.8: Kombination der Klassifikationskriterien ... 85
Abbildung 6.9: Ausschluss nicht sinnvoller Klassenkombinationen ... 85
Abbildung 6.10: Schematische Darstellung eines Klassifikationsbaums zur Selektion des geeigneten Safety-Pattern ... 86
Abbildung 6.11: Veranschaulichung der Relation „q darf ab p gültig sein“ ... 87
Abbildung 6.12: Veranschaulichung der Relation „q darf strikt nach p gültig sein“ ... 87
Abbildung 6.13: Veranschaulichung der Relation „q darf nur exakt nach p gültig sein“ ... 87
Abbildung 6.14: Beispiel eines Zustandsdiagramms für die Erfüllung des Ausdrucks (6.4) ... 91
Abbildung 6.15: Darstellung einer Relation zwischen p und q mit Reduktion von Informationen hinsichtlich irrelevanter Zustandsstrukturen ... 91
Abbildung 6.16: Darstellung einer Relation zwischen p und q mit Kennzeichnung der Bedingung p und der Modalitäten ... 92
Abbildung 6.17: Grafische Darstellung für den Ausdruck (6.4) ... 92
Abbildung 6.18: Veranschaulichung von EG p in [McMi97] ... 94
Abbildung 6.19: Erweiterte Veranschaulichungstechnik für das Saftey-Pattern-Konzept ... 94
Abbildung 6.20: Veranschaulichung von Erreichbarkeitsrelationen ... 95
Abbildung 6.21: Veranschaulichung der Bedeutung eines Safety-Pattern durch Simulation ... 96
Abbildung 6.22: Veranschaulichung für ein Safety-Pattern mit Erreichbarkeitseigenschaft ... 97
Abbildung 7.1: Direkter Zugriff auf die Erläuterung der Safety-Pattern-Klassifikation ... 104
Abbildung 7.2: Klassifikationsdialoge von SAPIS ... 105
Abbildung 7.3: Funktionen zur Safety-Pattern-Suche ... 106
Abbildung 7.4: Anleitung zur Spezifikation aussagenlogischer Ausdrücke in SAPIS ... 108
Abbildung 7.5: Systemarchitektur von SAPIS ... 110
Abbildung 8.1: Systemkomponenten der Fallstudie ... 113
Abbildung 8.2: Systemübersicht des Modellprozesses ... 115
Abbildung 8.3: Schnittstellen des Modellprozesses ... 116
Abbildung 8.4: Prinzip für die Erprobung ... 119

Abbildung 8.5: Transformation des UML-Modells der Systemfunktionen in C++-Code und in die Eingangssprache des Model-Checker SMV 120
Abbildung B.1: Gültigkeitsdauer einer Eigenschaft q bis zum Zeitpunkt t = T 145
Abbildung B.2: Gültigkeitsbeginn einer Eigenschaft q nach dem Zeitpunkt t = T 146
Abbildung B.3: Gültigkeit von q nach dem Zeitpunkt t = T_1 und bis zum Zeitpunkt t = T_2 146
Abbildung B.4: Gültigkeit von q ab dem Zeitpunkt, zu dem p eintritt 149
Abbildung B.5: Gültigkeit von q ab einem Zeitpunkt strikt nach p 149
Abbildung B.6: Gültigkeit von q im genau nächsten Zustand nach p 150
Abbildung B.7: Gültigkeit von q im selben Systemzustand wie p 150
Abbildung B.8: Gültigkeit von q bis einschließlich zu dem Zeitpunkt, zu dem p eintritt 151
Abbildung B.9: Gültigkeit von q vor dem Zeitpunkt, zu dem p eintritt 151
Abbildung C.1: Veranschaulichung der Bedeutung von SP 5 153
Abbildung C.2: Veranschaulichung der Bedeutung von SP 11 153
Abbildung D.1: Beispiel mit der permanenten Gültigkeit einer Aussage 157
Abbildung D.2: Beispiel für die Gültigkeit einer Aussage ab einem bestimmten Zeitpunkt .. 157
Abbildung D.3: Beispiel für die Erreichbarkeit der Gültigkeit einer Aussage 157
Abbildung E.1: Beispiel für Rückwärtssimulation 160
Abbildung E.2: Beispiel mit Werteverlaufvorgabe ohne Rückwärtssimulierbarkeit 161
Abbildung E.3: Beispiel für die Simulation der Komposition mehrerer Safety-Pattern 162
Abbildung E.4: Kompositions-Beispiel mit widersprüchlichem Werteverlauf 163
Abbildung F.1: Safety-Pattern-Beispiel – Teil 1 165
Abbildung F.2: Safety-Pattern-Beispiel – Teil 2 166
Abbildung H.1: Erster Teil der Anzeige eines Safety-Pattern durch SAPIS 171
Abbildung H.2: Zweiter Teil der Anzeige eines Safety-Pattern durch SAPIS 172

Tabellenverzeichnis

Tabelle 2.1: Symbole, Konnektoren und Begriffe der Aussagenlogik 15
Tabelle 2.2: Wahrheitstafel für die aussagenlogischen Konnektoren..... 15
Tabelle 4.1: Ansatzarten zur anwendbaren formalen Spezifikation im Vergleich 44
Tabelle D.1: Grafische Notation von GNSS..... 155
Tabelle D.2: Erläuterung der Bedeutung besonderer Symbole und Symbolkombinationen 156
Tabelle E.1: Verwendete Symbole in der Zeitdiagramm-Simulation des Safety-Pattern-Konzepts..... 159
Tabelle G.1: Bedeutung der besonderen Zeichen der Ableitungssprache 170

Abkürzungsverzeichnis

ACD	**A**rchitecture **C**omponent **D**evelopment
API	**A**pplication **P**rogram **I**nterface
ASP	**A**ctive **S**erver **P**ages
BCNF	**B**oyce-**C**odd **N**ormal **F**orm
BDD	**B**inary **D**ecision **D**iagrams
CTL	**C**omputation **T**ree **L**ogic
CSL	**C**ontinuous-time **S**tochastic **L**ogic
DF	**Def**inition
DFG	**D**eutsche **F**orschungs-**G**emeinschaft
DIN	**D**eutsche **I**ndustrie**n**orm
EBO	**E**isenbahn**b**etriebs**o**rdnung
EKrG	**E**isenbahn-**Kr**euzungs-**G**esetz
EN	**E**uropäische **N**orm
FFB	**F**unk-**F**ahr-**B**etrieb
GNSS	**G**rafische **N**otation zur **S**pezifikation mit **S**afety-Pattern / **G**raphical **N**otation for **S**pecification with **S**afety Patterns
GUI	**G**raphical **U**ser **I**nterface
HTML	**H**yper**t**ext **M**arkup **L**anguage
ID	**Id**entifikationsnummer
IT	**I**nformations**t**echnik
ITL	**I**nterval **T**emporal **L**ogic
INKA	**I**nduction **P**rover **K**arlsruhe
JSP	**J**ava **S**erver **P**ages
KA	**K**onzept-**A**nforderung
KIV	**K**arlsruhe **I**nteractive **V**erifier
LTL	**L**inear-Time **T**emporal **L**ogic
LSC	**L**ife **S**equence **C**harts

LGB	**L**ehmann **G**roß **B**ahn
LzA	**L**icht**z**eichen-**A**nlage
MSC	**M**essage **S**equence **C**harts
MuWiz	**Mu**-Calculus Formula Constructor **Wiz**ard
OCL	**O**bject **C**onstraint **L**anguage
PCTL	**P**robabilistic Real-Time **C**omputation **T**ree **L**ogic
PTL	**P**ast **T**emporal **L**ogic
PVS	**P**rototype **V**erification **S**ystem
RT-OCL	**R**eal-**T**ime-**OCL**
r.u.c	**r**estricted **u**nnested **c**losed
SAPIS	**Sa**fety**-P**attern **I**nstanziierungs-**S**ystem **/ Sa**fety **P**attern **I**nstantiation **S**ystem
SFC	**S**equential **F**unction **C**harts
SIL	**S**afety **I**ntegrity **L**evel **/ S**icherheits-**I**ntegritäts-**L**evel
SMV	**S**ymbolic **M**odel **V**erifier
StP	**S**oftware **t**hrough **P**ictures
SP	**S**afety-**P**attern
SchA	**Sch**ranken-**A**nlage
StVO	**St**raßen-**V**erkehrs-**O**rdnung
STD	**S**ymbolic **T**iming **D**iagrams
TDL	**T**ransformation **D**escription **L**anguage
TLA	**T**emporal **L**ogic for **A**ctions
UML	**U**nified **M**odeling **L**anguage
VIS	**V**erification **I**nteracting with **S**ynthesis
VSE	**V**erification **S**upport **E**nvironment
WWW	**W**orld **W**ide **W**eb
XML	**E**xtensible **M**arkup **L**anguage
XSLT	E**X**tensible **S**tylesheet **L**anguage **T**ransformation

Begriffsverzeichnis

Aussagenlogik: Die *Aussagenlogik* befasst sich mit der logischen Bewertung von Aussagen.

Automatisierungssystem: Ein *System* mit dem Ziel der Automatisierung technischer Prozesse.

Bedingungsvariable: Boolesche Variable, deren Erfüllung die Geltung oder Erreichbarkeit einer Eigenschaft durch eine logische Relation bedingt.

Eigenschaftsvariable: Boolesche Variable, die bei der Geltung einer Eigenschaft erfüllt sein muss.

Erreichbarkeitseigenschaft: Eine Eigenschaft, bei der entweder generell oder ausgehend von einer Bedingung, die Geltung einer Eigenschaft erreicht wird.

Fachwörter: In *Normsprachen* wird durch die Definition von *Fachwörtern* (auch *Themenwörter* genannt) in einem Lexikon eine genormte Terminologie festgelegt. Es wird klar definiert und abgegrenzt, was mit den *Fachwörtern* im Kontext des Anwendungs- bzw. Problembereichs gemeint ist.

Formale Korrektheit: Anforderungen werden vollständig mit mathematischer Exaktheit erfüllt.

Formale Methoden: *Formale Methoden* sind mathematisch fundierte Techniken zur Spezifikation und Analyse von *Systemen*.

Formale Spezifikation: Für die Spezifikation werden Beschreibungsmittel verwendet, die aufgrund einer eindeutigen *Syntax* und *Semantik* eine mathematisch exakte Beschreibung ermöglichen, sodass die Spezifikation formal verifizierbar ist.

Formale Verifikation: Überprüfung der Einhaltung von Anforderungen mit mathematischer Exaktheit.

Funktionale Sicherheitsanforderungen: Sie umfassen die Anforderungen an die sicherheitsrelevanten Funktionen eines *Automatisierungssystems*. Durch die Einhaltung dieser Anforderungen wird die sichere Funktionalität gewährleistet.

Gefahr: Nicht akzeptierbare erreichbare Situation eines *Automatisierungssystems*, in der ein Schaden droht (komplementär zu *Sicherheit*).

Inhärente Fehler: *Inhärente Fehler* sind bereits vor der Inbetriebnahme vorhanden. Sie haben ihre Ursache in menschlichen logischen Fehlern in der Systementwicklung.

Instanziierung: Überführung eines generischen Elements in ein konkretes.

Logik: Allgemeine Theorie der folgerichtigen Aussagen und ihrer gültigen Beziehungen.

Model-Checking: Verfahren zur *formalen Verifikation*. Die Durchführung der *formalen Verifikation* geschieht bei diesem Verfahren algorithmisch und deshalb automatisch.

Modell der Systemfunktionen: Allgemeine Bezeichnung für das System-, Entwurfs- oder Implementierungsmodell eines Automatisierungssystems bzw. der Software eines Automatisierungssystems.

Muster: Vorgefertigte generische Lösungsvorschrift zur Mehrfachverwendung.

Nichtinhärente Fehler: Unter *nichtinhärenten Fehlern* wird menschliches Fehlverhalten verstanden, das erst nach der Inbetriebnahme begangen wird (Bedienfehler oder Wartungsfehler).

Normsprache: Eine *Normsprache* ist eine konstruierte Sprache, die aus einer in einem Anwendungsbereich eingesetzten natürlichen Sprache (Gemeinsprache oder Fachsprache) abgeleitet wird. Die Wörter der *Normsprache* nehmen nicht durch einen Anwendungskontext eine bestimmte Bedeutung an, sondern sind als Terminologie eines Fachbereichs für stets dieselbe Verwendung vorgesehen.

Pfadquantor: In temporallogischen Sprachen können durch *Pfadquantoren* Aussagen über die Verzweigungen eines Zustandsraums eines *Systems* getroffen werden. Mit Pfadquantoren kann eine Aussage für alle möglichen Systemverhaltenspfade einer Verzweigung oder für mindestens einen Pfad formuliert werden. Dabei spiegelt ein Systemverhaltenspfad ein mögliches Systemverhalten wider.

Physikalische Fehler: Physikalische Fehler werden durch physikalische oder chemische Ausfallmechanismen oder Effekte verursacht.

Prädikatenlogik: Sie entsteht aus der *Aussagenlogik* durch Hinzunahme der Quantoren $\forall$ („für alle“) und $\exists$ („es gibt mindestens ein“) sowie der Einführung von Funktionen, sog. „Prädikate“. Aufgrund der Quantoren sind Aussagen über Mengen möglich.

Safety-Pattern: Spezifikationsmuster zur Formulierung von *Sicherheitsanforderungen*

Safety-Pattern-Katalog: Auflistung und Ordnung aller *Safety-Pattern*.

Safety-Pattern-Variablen: Die im *Safety-Pattern* vorkommenden *Bedingungs-* und *Eigenschaftsvariablen*.

Sicherheit: Das Nichtvorhandensein eines unzulässigen Schadensrisikos.

Sicherheitsanforderung: Anforderung, die zur Gewährleistung der *Sicherheit* erfüllt werden muss.

Sicherheitsbezogene Korrektheit: Das *Modell der Systemfunktionen* erfüllt vollständig die *funktionalen Sicherheitsanforderungen*.

Syntax: Die *Syntax* behandelt die Regeln für den Aufbau und die Anordnung von Ausdrücken in einem Beschreibungsmittel.

System: Eine Menge kooperierender Elemente, die zusammen eine bestimmte Aufgabe erfüllen.

Semantik: Sinn und Bedeutung von Beschreibungsmitteln beziehungsweise sprachlichen Zeichen, Wörtern und Ausdrücken.

Temporallogik: *Temporallogik* ermöglicht Aussagen, die unter Berücksichtigung der Zeit (eine Menge mit linear angeordneten Elementen, den Zeitpunkten) notwendig oder möglicherweise gelten.

Theorem-Proving: Verfahren zur *formalen Verifikation*, bei dem die Beweisschritte durch interaktive Kommunikation mit einem Benutzer werkzeuggestützt erfolgen.

Themenwörter: s. *Fachwörter*.

Zeitquantor: In temporallogischen Sprachen können durch *Zeitquantoren* zeitliche Relationen in Systemeigenschaften beschrieben werden.

Zusammenfassung

Durch formale Verifikation kann die Einhaltung funktionaler Sicherheitsanforderungen im Modell der Systemfunktionen eines Automatisierungssystems mit Sicherheitsverantwortung mit mathematischer Exaktheit überprüft werden. Eine Voraussetzung hierfür ist, dass die Sicherheitsanforderungen in einer formalen Spezifikationssprache, d. h. mit einer eindeutigen Syntax und Semantik, formuliert werden. Eine entscheidende Ursache für die wenig verbreitete Anwendung formaler Verifikation liegt in der Schwierigkeit der formalen Spezifikation temporaler Relationen, die bei der Formulierung funktionaler Sicherheitsanforderungen für Automatisierungssysteme ausgedrückt werden müssen. Wird die formale Spezifikationssprache nicht vollständig beherrscht, werden Sicherheitsanforderungen leicht fehlerhaft spezifiziert, woraus die Entwicklung eines Automatisierungssystems resultieren kann, von dem Gefahren ausgehen. Dasselbe ist der Fall, wenn eine Sicherheitsanforderung falsch interpretiert wird.

Diesen Schwierigkeiten kann durch ein Verfahren begegnet werden, bei dem Expertenwissen für die formale Spezifikation von funktionalen Sicherheitsanforderungen vermittelt wird. Dies wird durch die Adaption und Nutzung von aus der Softwaretechnik bekannten Wiederverwendungskonzepten erreicht. Mithilfe des daraus resultierenden Safety-Pattern-Konzepts wird die Formalisierung funktionaler Sicherheitsanforderungen vereinfacht, indem Safety-Pattern mit generischen formalen Spezifikationen verwendet werden. Die Safety-Pattern, die für das jeweilige Spezifikationsproblem geeignet sind, müssen aus einem Katalog selektiert werden. Die korrekte Interpretation von Sicherheitsanforderungen, die mithilfe von Safety-Pattern spezifiziert worden sind, wird unterstützt, indem die Bedeutung im Safety-Pattern-Katalog nachgeschlagen werden kann. Um alle Arten funktionaler Sicherheitsanforderungen spezifizieren zu können, wurde der Katalog so entwickelt, dass er durch die Verwendung einzelner generischer Formulierungen sowie durch deren Komposition eine vollständige Grundlage zur Spezifikation aller Arten funktionaler Sicherheitsanforderungen für Automatisierungssysteme bietet. Darüber hinaus enthält der Katalog generische Formulierungen, durch welche die Handhabbarkeit des Verfahrens gewährleistet wird.

Durch die Verwendung einer Safety-Pattern-Normsprache wird neben der formalen Spezifikation eine präzise und einfach interpretierbare Spezifikation funktionaler Sicherheitsanforderungen in einer eingeschränkten Terminologie der natürlichen Sprache ermöglicht. Zudem wurde eine spezielle grafische Notation für die Veranschaulichung komplizierter logischer Zusammenhänge entwickelt. Durch ein Werkzeugkonzept und den Einsatz multimedialer Techniken (Interaktionen, textuelle sowie grafische Beschreibungen und Simulationen) wird die Selektion, Interpretation und Instanziierung der Safety-Pattern unterstützt. Durch das Fallbeispiel „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb“ wird die Anwendung des Verfahrens für die Entwicklung sicherer Automatisierungssysteme demonstriert.

Abstract

For industrial automation systems with safety responsibility it is of vital importance to check the compliance of functional safety requirements. By means of formal verification it is possible to check with mathematical exactness if functional safety requirements are preserved in a model of the system functions. As a precondition, the safety requirements have to be specified in a formal specification language, i.e. that they are formulated with unambiguous syntax and semantics. It is highly challenging to specify formally temporal relations, which is compulsory for functional safety requirements of industrial automation systems. This is a decisive reason for the very rare use of formal verification. If the formal specification language is not mastered completely, then safety requirements will easily be falsely specified or will easily be misinterpreted. As a consequence unsafe industrial automation systems might be developed.

Theses difficulties can be tackled by a technique, in which expertise on formal specification of functional safety requirements is imparted. This is achieved by adapting and utilizing reuse concepts, known from software engineering. The resulting safety patterns concept allows to simplify the formalisation of functional safety requirements as safety patterns with generic formal specifications are used. The safety patterns which are suitable for the respective specification problem have to be selected from a catalogue. The correct interpretation of safety requirements which have been specified on the basis of safety patterns is supported by the possibility to check their meaning in the safety patterns catalogue. The catalogue represents a complete basis for specifying all kinds of functional safety requirements for industrial automation systems by using single generic formulations as well as compositions of these formulations. Moreover, the catalogue contains generic formulations, which guarantee that the technique can be handled.

Beside the formal specification, a safety patterns norm language allows to specify functional safety requirements in a precise and simply interpretable way by using a restricted terminology of natural language. In addition, a particular graphical notation has been developed in order to visualize complicated logical connections. The safety patterns selection, interpretation and instantiation are supported by a tool concept and by the use of multimedia technologies (interactions, textual as well as graphical descriptions and simulations). The application of the safety patterns concept is demonstrated by means of the case study „single-track level crossing in radio-based operation“.

Der Unterschied zwischen dem richtigen Wort und dem beinahe richtigen ist derselbe wie zwischen dem Blitz und dem Glühwürmchen.

M. Twain

1 Einleitung und Motivation

1.1 Bedeutung von Sicherheitsanforderungen für die Entwicklung sicherer Automatisierungssysteme

Viele Automatisierungssysteme besitzen Sicherheitsverantwortung. Alle Verkehrssysteme (Systeme im spurgeführten Verkehr, Flugzeugsysteme und Automobilsysteme) sind sicherheitskritische Systeme. Andere klassische sicherheitskritische Anwendungsbereiche sind Energiekraftwerke, verfahrenstechnische Anlagen sowie Produktionsanlagen. Zudem gewinnt Sicherheitstechnik auch durch die zunehmende Automatisierung von Haushaltsgeräten und die wachsende Nutzung von technischen Geräten in der Medizintechnik immer mehr an Bedeutung. Dabei ist Sicherheit (engl.: *safety*) im Sinne der Verhinderung von Gefahren zu verstehen, die von Automatisierungssystemen im Betrieb ausgehen können [LaGö99a, S.321].

Entscheidende Grundlagen für die Entwicklung sicherer Automatisierungssysteme ist die fachtechnische Lösungskonzeption (Systemmodell). Die fachtechnische Lösungskonzeption beschreibt das grundsätzliche Konzept zur Lösung der gestellten Automatisierungsaufgabe [LaGö99a, S. 332]. Wenn nun die fachtechnische Lösungskonzeption fehlerbehaftet ist, so kann sich solch ein Fehler durch den gesamten weiteren Systementwicklungsprozess hindurchziehen. Ist der Fehler sicherheitskritisch, sodass im fachtechnischen Lösungskonzept ein System modelliert wird, das gefährliche Zustände einnehmen kann, so können die Auswirkungen fatal sein. Die Folge kann die Entwicklung eines Automatisierungssystems sein, das unsicher ist.

In der Entwicklung sicherer Automatisierungssysteme ist deshalb ein Nachweis der sicheren Funktion zur Aufdeckung inhärenter Fehler in der fachtechnischen Lösungskonzeption von großer Bedeutung. Heutzutage werden auch Sicherheitsfunktionen meist durch Software realisiert [Moik02] [Gunz03]. In der Entwicklung sicherer Automatisierungssysteme ist es deshalb wichtig, die Korrektheit der Softwaremodelle zu überprüfen. Dieser Nachweis der sicherheitsbezogenen Korrektheit wird von der generischen Norm für funktionale Sicherheit [IEC61508] [DINEN61508] gefordert. Zu diesem Zweck müssen Sicherheitsanforderungen formuliert werden, durch deren Einhaltung die sichere Funktion gewährleistet werden kann.

Wenn solche Sicherheitsanforderungen in natürlicher Sprache erstellt werden, ergeben sich zwei Probleme. Zum einen ist natürliche Sprache mehrdeutig, sodass die Sicherheitsanforderungen falsch interpretiert werden können. Zum anderen ist ein Nachweis der sicheren Funktion auf der

Grundlage von Sicherheitsanforderungen in natürlicher Sprache vage, weil er auf Argumentation in natürlicher Sprache basiert. Im Gegensatz hierzu steht die formale Spezifikation von Sicherheitsanforderungen. Formale Spezifikation bedeutet, dass für die Spezifikation Beschreibungsmittel verwendet werden, die aufgrund einer eindeutigen Syntax und Semantik eine mathematisch exakte Beschreibung ermöglichen. Voraussetzung dabei ist eine eindeutig definierte Menge von Symbolen bzw. Zeichen (Sigmatik) [AFSC00]. Durch die formale Beschreibung werden Missverständnisse bei der Interpretation, die aus Mehrdeutigkeiten resultieren, vermieden. Sie erlaubt zudem einen Nachweis der sicheren Funktion mit mathematischer Exaktheit (formale Verifikation). Ein gravierender Nachteil liegt jedoch darin, dass bei der Spezifikation und Interpretation leicht Fehler gemacht werden können. Wird das formale Beschreibungsmittel nicht vollständig beherrscht, können Sicherheitsanforderungen entgegen der eigentlichen Intention spezifiziert werden. Außerdem kann sich die Interpretation formal spezifizierter Sicherheitsanforderungen von der Bedeutung der formalen Spezifikation unterscheiden. Die formale Spezifikation ermöglicht zwar präzise Beschreibungen, allerdings besteht ein erheblicher Mangel bezüglich der einfachen Anwendung der formalen Beschreibungsmittel.

1.2 Zielsetzung

In dieser Arbeit wird die Fragestellung verfolgt, wie einerseits eine präzise Spezifikation funktionaler Sicherheitsanforderungen erreicht werden kann und wie andererseits zugleich ein einfach anwendbares Verfahren zur Spezifikation bereitgestellt werden kann, um Fehler bei der Spezifikation und Interpretation zu vermeiden.

Präzise Spezifikation der Sicherheitsanforderungen

Die Zielsetzung dieser Arbeit ist die Bereitstellung eines Verfahrens zur präzisen Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme. Sicherheitsanforderungen müssen korrekt formalisiert werden, sodass ein Nachweis der sicheren Funktion mit mathematischer Exaktheit durchgeführt werden kann. Ausgangspunkt dafür ist, dass die Sicherheitsanforderungen hergeleitet und in informeller Form im Kontext der fachtechnischen Lösungskonzeption bzw. des Softwaremodells aufgestellt wurden.

Anwendbarkeit in der industriellen Praxis

Das in dieser Arbeit vorgestellte Verfahren soll die industrielle Praxis verbessern. Deshalb müssen Randbedingungen der industriellen Praxis berücksichtigen werden. Das Verfahren muss so entwickelt werden, dass es für die ingenieurmäßige Lösung von Automatisierungsaufgaben einsetzbar ist. Die Bedürfnisse aller Personengruppen,

- die an der Spezifikation der Sicherheitsanforderungen beteiligt sind,
- die die Sicherheitsanforderungen verwenden und dabei interpretieren und
- die die Spezifikation der Sicherheitsanforderungen prüfen und nachvollziehen,

müssen berücksichtigt werden.

1.3 Gliederung der Arbeit

In **Kapitel 2** werden die für diese Arbeit notwendigen Grundlagen beschrieben. Dazu werden die für das Thema wichtigen Begriffe und Verfahren aus den Bereichen Sicherheitstechnik und formale Methoden eingeführt.

In **Kapitel 3** werden die Konzept-Anforderungen an ein Verfahren zur Spezifikation von Sicherheitsanforderungen aufgestellt. Die Grundlagen werden hinsichtlich der Zielsetzung analysiert. Auf dieser Basis werden die Konzept-Anforderungen hergeleitet und definiert.

In **Kapitel 4** wird der Stand der Forschung hinsichtlich der definierten Konzept-Anforderungen analysiert. Dabei wird das adäquate Beschreibungsmittel zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme bestimmt. Bestehende Ansätze zur anwendbaren formalen Anforderungsspezifikation werden gruppiert und beschrieben. Die untersuchten Ansätze werden bewertet und miteinander verglichen. Aus den Ergebnissen werden Schlussfolgerungen für die Entwicklung eines Verfahrens zur Spezifikation von Sicherheitsanforderungen gezogen. Die Konzept-Anforderungen werden verfeinert.

In **Kapitel 5** wird der Safety-Pattern-Ansatz als Verfahrens-Ansatz für die anwendbare formale Spezifikation funktionaler Sicherheitsanforderungen hergeleitet. Dabei wird der Grundgedanke des Einsatzes von Mustern und Komponenten für Wiederverwendung in der Softwareentwicklung auf die Spezifikation von Sicherheitsanforderungen übertragen. Um die zuvor festgelegten Anforderungen zu erfüllen, werden entsprechende Maßnahmen abgeleitet und erläutert.

In **Kapitel 6** erfolgt die Konzeption für den in Kapitel 5 hergeleiteten Ansatz. Die universelle Verwendbarkeit des Verfahrens für die Spezifikation von Sicherheitsanforderungen in der Automatisierungstechnik wird hergeleitet. Die Teilkonzepte des Safety-Pattern-Konzepts werden entwickelt.

Kapitel 7 behandelt ausführlich das Werkzeugkonzept für die Spezifikation, Interpretation und Verwaltung von Sicherheitsanforderungen. Es wird sowohl die Konzeption als auch die Anwendung eines Web-basierten Unterstützungssystems für das Safety-Pattern-Konzept beschrieben.

Zur Veranschaulichung der entwickelten Ideen und Teilkonzepte wird das vorgeschlagene Verfahren in **Kapitel 8** auf ein konkretes Anwendungsbeispiel aus der Verkehrsleittechnik angewendet. Als Fallstudie dient das betriebliche Konzept „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb“. Dabei wird ein Modellprozess der Anwendung für die Demonstration und Erprobung verwendet.

Die wesentlichen Ergebnisse der Arbeit werden in **Kapitel 9** zusammengefasst und bewertet. Außerdem werden Ansatzpunkte für weiterführende Arbeiten genannt.

2 Grundlagen

In diesem Kapitel werden die Grundlagen für diese Arbeit beschrieben. Unter Berücksichtigung des thematischen Rahmens der Arbeit wird eine Einführung in die Themen Automatisierungssysteme, Sicherheit von Automatisierungssystemen sowie formale Methoden gegeben. Es wird auf den Stand der Technik, wie er von den relevanten Normen definiert wird, eingegangen. Generelle Möglichkeiten der Spezifikation von Sicherheitsanforderungen werden beschrieben. Dabei werden die wesentlichen Begriffe für diese Arbeit definiert.

2.1 Charakteristische Merkmale von Automatisierungssystemen

Ein Prozessautomatisierungssystem setzt sich aus folgenden Bestandteilen zusammen: Mensch (Benutzer), Rechner- und Kommunikationssystem und technisches System mit dem technischen Prozess. Abbildung 2.1 zeigt den Aufbau eines solchen Prozessautomatisierungssystems. *Prozess* ist „eine Gesamtheit von aufeinander einwirkenden Vorgängen in einem System, durch die Materie, Energie oder Information umgeformt, transportiert oder gespeichert wird. Ein *technischer Prozess* ist ein Prozess, dessen physikalische Größen mit technischen Mitteln erfasst und beeinflusst werden“ [LaGö99a, S. 1] [DIN66201]. Der technische Prozess läuft innerhalb eines technischen Systems ab (z. B. „Backprozess in einer Brotbackmaschine“ oder „Sicherungsvorgang eines Bahnübergangs“). Kern des Rechner- und Kommunikationssystems ist ein durch Software realisiertes Steuerungsprogramm [Moik 02, S. 6].

Mit *Automatisierungssystem* werden die Einrichtungen bezeichnet, die für die Automatisierung technischer Prozesse erforderlich sind [Göhn05b]. Die Besonderheit von Automatisierungssystemen ist die Interaktion mit dem technischen Prozess. Aufgrund dessen sind Automatisierungssysteme aus Sicht der Informatik reaktive Systeme (s. [LaGö99a, S.181], [Moik 02, S. 6]) im Unterschied zu Transformationssystemen. Während bei Transformationssystemen lediglich Eingangs- in Ausgangsdaten transformiert werden, unterliegt ein Automatisierungssystem zudem den zeitlichen Abläufen im technischen Prozess. Das bedeutet, dass das Rechnersystem mit den Vorgängen im technischen Prozess Schritt halten muss. Deshalb spielt die Dimension der Zeit für Automatisierungssysteme eine besondere Rolle [LaGö99a, S. 11]. Hierdurch unterscheiden sich solche Systeme von der allgemeinen Datenverarbeitung. Auf Vorgänge im technischen Prozess muss rechtzeitig reagiert werden. Die Korrektheit der Informationsverarbeitung in Automatisierungssystemen hängt deshalb nicht nur von dem Resultat der Berechnung ab, sondern auch davon, wann dieses Resultat vorliegt. Zudem müssen unterschiedliche Vorgänge des technischen Prozesses (Teilprozesse) gleichzeitig bearbeitet werden.

Deshalb sind die ersten beiden Hauptmerkmale von Automatisierungssystemen: **Rechtzeitigkeit und Gleichzeitigkeit** [LaGö99a, S. 11]

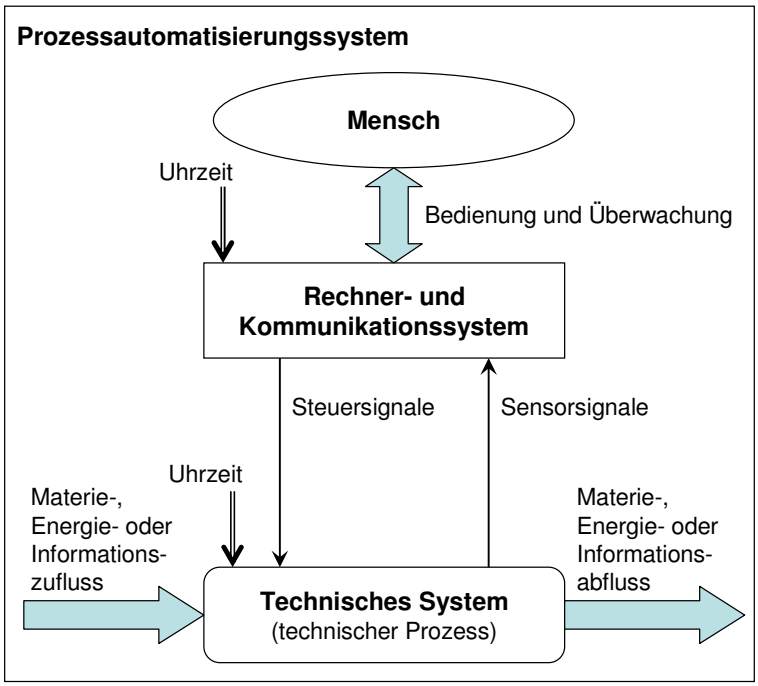


Abbildung 2.1: Struktur eines Prozessautomatisierungssystems

Da das zeitliche Verhalten bei Automatisierungssystemen von besonderer Bedeutung ist, spielt bei der Beschreibung von Eigenschaften von Automatisierungssystemen das Ausdrücken zeitlicher Relationen eine besondere Rolle. Deshalb müssen in Anforderungen für Automatisierungssysteme zeitliche Aussagen formuliert werden können.

Der Ausfall eines Automatisierungssystems kann hohe betriebswirtschaftliche Kosten mit sich bringen, kann aber auch zu Sachschäden führen, Verletzungen von Menschen zur Folge haben, oder gar Menschenleben gefährden. Um dies zu verhindern, müssen Automatisierungssysteme verlässlich sein. Bei verlässlichen Automatisierungssystemen kann volles Vertrauen in die Funktionalität gesetzt werden. Für sichere Automatisierungssysteme ist dies von entscheidender Bedeutung. Automatisierungssysteme müssen sich zudem vorhersehbar verhalten. Die Reaktionen des Systems auf die möglichen Vorgänge und Fehlersituationen des technischen Prozesses müssen genau geplant (vorhersehbar) sein. Somit sind die weiteren Hauptmerkmale von Automatisierungssystemen: **Verlässlichkeit und Vorhersehbarkeit** [LaGö99a, S. 11].

2.2 Sicherheit von Automatisierungssystemen

2.2.1 Sicherheit

Sicherheit bedeutet die Verhinderung von Gefahren, die vom Betrieb eines Automatisierungssystem ausgehen können, und die der Umwelt oder Menschen drohen (engl.: *safety*) [LaGö99a,

S. 321]. Im Gegensatz hierzu wird mit IT-Sicherheit[1] (engl. *security*) die Verhinderung von Bedrohungen bezeichnet, die während des Betriebs des Automatisierungssystems von der Umwelt oder dem Menschen ausgehen. IT-Sicherheit steht nicht im Fokus dieser Arbeit. Mit „Sicherheit“ wird in dieser Arbeit immer technische Sicherheit (engl.: *safety*) bezeichnet. Die Sicherheit eines Automatisierungssystems bezeichnet das Nichtvorhandensein eines unzulässigen Schadensrisikos [Kona03] [EN50126]. Ein sicheres System „beinhaltet die Fähigkeit innerhalb vorgegebener Grenzen für eine gegebene Zeitdauer keine Gefahr zu bewirken oder eintreten zu lassen“ [LaGö99a, S. 318]. Dabei ist eine Gefahr eine „potenzielle Schadensquelle“ [IEC61508] oder in anderen Worten eine „Bedingung, die zu einem Unfall führen kann“ [EN50129]. Bei einem sicheren Automatisierungssystem sind also keine nichtakzeptablen Systemsituationen erreichbar, in denen Schaden droht.

Insbesondere für solche sicherheitskritischen technischen Prozesse muss das Verhalten des Automatisierungssystems deterministisch und damit vorhersehbar sein [LaGö99a, S. 12]. Um dies zu erreichen, müssen Maßnahmen ergriffen werden, die gegen gefährliche Auswirkungen von Fehlern und Ausfällen gerichtet sind.

2.2.2 Sicherheitsnachweis

Im Sicherheitsnachweis muss der Nachweis erbracht werden, dass das Automatisierungssystem sicher ist. Die generische Norm für funktionale Sicherheit [IEC61508] fordert die Prüfung dieses Sicherheitsnachweises durch unabhängige Gutachter (Genehmigung durch Zulassungsbehörde). Für einen Sicherheitsnachweis ist eine Risiko- und Gefährdungsanalyse erforderlich, vgl. [Bits02b], [Brab02], [Kona03], [PoHa02], [Renp02]. In der Risikoanalyse werden zunächst die System-Gefährdungen ermittelt, welche aus dem Versagen der in der fachtechnischen Lösungskonzeption beschriebenen Funktionen entstehen können. Für das Eintreten der Gefährdungen werden tolerierbare Gefährdungsraten festgelegt, für die das Automatisierungssystem als sicher gilt. In der Gefährdungsanalyse wird untersucht, ob durch das System, so wie es realisiert werden soll, die festgelegten Gefährdungsraten eingehalten werden. Bei der Anwendung der Risiko- und Gefährdungsanalyse geht es um Maßnahmen zur Überprüfung von Absicherungen gegen Funktionsversagen, das durch physikalische Fehler sowie nichtinhärente Fehler [LaGö99a, S. 325] verursacht wird. Physikalische Fehler werden durch physikalische oder chemische Ausfallmechanismen (Ausfälle von Hardwarekomponenten) oder Effekte verursacht. Unter nichtinhärenten Fehlern wird menschliches Fehlverhalten verstanden, das erst nach der Inbetriebnahme begangen wird (Bedienfehler oder Fehler bei Wartung).

Darüber hinaus muss im Sicherheitsnachweis die korrekte sichere Funktion nachgewiesen werden. Es muss gezeigt werden, dass zunächst durch die fachtechnische Lösungskonzeption und dann auch durch die Art und Weise der Realisierung des Automatisierungssystems die

[1] Informationstechnik-Sicherheit

Anforderungen an die sichere Funktion eingehalten werden. Dabei handelt es sich um die Absicherung gegen inhärente Fehler [LaGö99a, S. 325]. Inhärente Fehler sind systematische Fehler. Sie entstehen durch menschliche Fehler in der Systementwicklung (logische Fehler). Es sind Fehler, die schon vor Beginn des Betriebs in der fachtechnischen Lösungskonzeption, im Systementwurf oder in der Realisierung unterlaufen sind.

Diese Arbeit behandelt die Spezifikation von Sicherheitsanforderungen zur Aufdeckung und Vermeidung von inhärenten sicherheitskritischen Fehlern. Um solche Fehler aufdecken zu können, muss die sicherheitsbezogene Korrektheit des jeweiligen Modells der Systemfunktionen überprüft werden. Das bedeutet, dass überprüft werden muss, ob das Modell der Systemfunktionen die erforderlichen Sicherheitsanforderungen erfüllt.

Wie bereits in Abschnitt 2.1 erläutert wurde, ist der Kern des Rechner- und Kommunikationssystems, das die Systemfunktionen realisiert, ein durch Software realisiertes Steuerungsprogramm. Somit liegt die Sicherheitsverantwortung heutzutage in zunehmendem Maße bei der Software, vgl. [Moik02], [Gunz03, S. 7]. Nach [Leve95] kommt der Software dabei in zweierlei Hinsicht Bedeutung zu. Zum einen darf sie selbst keine Gefahrensituationen verursachen, d. h., sie darf keine sicherheitskritischen systematischen Fehler enthalten. Zum anderen muss die Software Sicherheitsfunktionen bereitstellen, durch die das Automatisierungssystem sicher auf Ausfälle von Systemkomponenten reagiert. In der Entwicklung solcher Automatisierungssysteme ist es deshalb sinnvoll, die Korrektheit der System- und Softwaremodelle bezüglich der Sicherheitsanforderungen vollständig zu prüfen (sicherheitsbezogene funktionale Korrektheit).

2.2.3 Sicherheitsanforderungen

Für die Gewährleistung der Sicherheit eines Automatisierungssystems müssen Sicherheitsanforderungen eingehalten werden. In solchen Anforderungen werden Aussagen über die Sicherheitseigenschaften des Automatisierungssystems getroffen. Dies sind Eigenschaften, deren Fehlen die Sicherheit des Systems gefährden kann [Gunz03]. Des Weiteren wird mit Sicherheitsanforderungen spezifiziert, wie die erforderlichen Sicherheitseigenschaften gewährleistet werden sollen. Insgesamt können *funktionale Sicherheitsanforderungen, Sicherheitsintegritätsanforderungen* und *nichtfunktionale Sicherheitsanforderungen* unterschieden werden, vgl. [EN50129], [Kona03], [FNT04], [Warl04]:

Funktionale Sicherheitsanforderungen beschreiben die sicherheitsrelevanten Funktionen, die durch das Automatisierungssystem ausführbar sein sollen, vgl. [LaGö99b, S.337]. Es wird beschrieben, was das Automatisierungssystem leisten können muss und welches Verhalten es nicht aufweisen darf, um Sicherheit zu gewährleisten. Werden alle funktionalen Sicherheitsanforderungen vom Automatisierungssystem eingehalten, so ist das sichere Verhalten gewährleistet. Durch die Überprüfung der Einhaltung von funktionalen Sicherheitsanforderungen im Modell der Systemfunktionen wird die sicherheitsbezogene funktionale Korrektheit kontrolliert.

Sicherheitsintegritätsanforderungen bestimmen das geforderte Maß an Sicherheit, mit dem die funktionalen Sicherheitsanforderungen erfüllt werden sollen. Durch eine Sicherheitsanforderungsstufe (SIL0 bis SIL4 – engl.: *Safety Integrity Level*) wird dabei der erforderliche Grad des Vertrauens festgelegt, mit dem das Versagen der Sicherheitsfunktionen durch systematische Fehler ausgeschlossen werden kann. Die Anforderungen an die Sicherheit für den Ausschluss physikalischer Fehler werden hingegen durch Ausfallraten quantitativ vorgegeben.

In *nichtfunktionalen Sicherheitsanforderungen* wird beschrieben, auf welche Weise funktionale Sicherheitsanforderungen erfüllt werden sollen. Sie geben an, wie die Entwicklung, die Inbetriebnahme und der Betrieb erfolgen sollen, um Sicherheit zu gewährleisten. Dies schließt auch Anforderungen an Schnittstellen, die Systemtechnik und die Qualität mit ein [LaGö99a, S. 338]. Typische nichtfunktionale Sicherheitsanforderungen betreffen z. B. die Einhaltung von sicherheitsrelevanten Richtlinien, Vorschriften sowie Normen. Eine häufige nichtfunktionale Sicherheitsanforderung in Ausschreibungen für Eisenbahnsicherungssysteme ist die Forderung der Entwicklung gemäß den CENELEC-Normen ([EN50126], [EN50128], [EN50129]), vgl. [Bits02b]. Ein weiteres typisches Beispiel dafür, wie die Softwareentwicklung erfolgen soll, um Sicherheit gewährleisten zu können, ist die folgende aus den Normen abgeleitete Sicherheitsanforderung: „*In der Softwareimplementierung der Sicherheitsfunktionen, die die SIL 3 erfüllen sollen, dürfen keine dynamischen Objekte verwendet werden.*“

In dieser Arbeit wird die Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme behandelt. Für solche Anforderungen ist die Formalisierung von wesentlicher Bedeutung, da dies die Voraussetzung ist, um sicherheitsbezogene funktionale Korrektheitsprüfungen mit mathematischer Exaktheit durchführen zu können. Nichtfunktionale Sicherheitsanforderungen werden nicht für Korrektheitsprüfungen verwendet. Deshalb ist deren Formalisierung von geringerer Bedeutung (vgl. [Moik02, S. 93]). Sie werden ausschließlich informell in natürlicher Sprache beschrieben [LaGö99a, S. 338]. Zudem enthalten sie keine temporalen Relationen, deren Formalisierung jedoch insbesondere die Schwierigkeiten ausmachen, wie noch gezeigt wird. Deshalb werden in dieser Arbeit nur funktionale Sicherheitsanforderungen betrachtet.

Funktionale Sicherheitsanforderungen haben die Verhinderung von Systemsituationen zum Ziel, von denen Gefahren ausgehen oder die Verhinderung von sicherheitskritischen Folgen, die durch das Erreichen bestimmter Systemsituationen ausgelöst werden, vgl. [LaGö99a, S. 362], [LaGö99b, S. 398], [Schn99, S. 240]. Im ersten Fall geht es um die Vermeidung von Systemfehlern und Ausfällen. Im zweiten Fall treten zwar Systemfehler und Ausfälle auf, aber es werden Sicherheitsfunktionen gefordert, die sicherheitskritische Folgen verhindern. Werden Sicherheitsanforderungen verletzt, kann es zu sicherheitskritischen Fehlfunktionen kommen. Sicherheitsfunktionen dürfen somit zum einen selbst keine sicherheitskritischen Fehler verursachen. Zum anderen müssen Sicherheitsfunktionen Fehler und Ausfälle im Automatisierungssystem erkennen und entsprechend darauf reagieren.

In beiden Fällen dürfen zur Spezifikation der Sicherheitsanforderung niemals Formulierungen verwendet werden, mit denen eine nicht-deterministische Relation von Systemgrößen ausgedrückt wird oder in denen offen gelassen wird, unter welchen Voraussetzungen die getroffene Aussage gilt (nicht-deterministische Forderung). Werden solche Formulierungen verwendet, kann die Verhinderung gefährlicher Systemsituationen bzw. die Verhinderung von sicherheitskritischen Folgen und damit die Sicherheit nicht gewährleistet werden. Für sicherheitskritische Automatisierungssysteme spielt die Vorhersehbarkeit, d. h. die Determiniertheit eine entscheidende Rolle [LaGö99a, S. 186]. Ist der genaue Ablauf nicht determiniert, kann das zeitliche Verhalten und damit die Sicherheit des Automatisierungssystems nicht garantiert werden.

Im Produktlebenszyklus sind funktionale Sicherheitsanforderungen Vorgaben für die Entwicklung eines sicheren Automatisierungssystems. Der Nachweis sicherheitsbezogener Korrektheit muss gegenüber den Sicherheitsanforderungen geführt werden. Somit sind funktionale Sicherheitsanforderungen zudem die Grundlage für die Erbringung des Sicherheitsnachweises. Des weiteren stellen sie Vorgaben dar, die bei der Wartung und Pflege eines sicheren Automatisierungssystems beachtet und damit verstanden werden müssen.

2.2.4 Aufstellung und Formulierung funktionaler Sicherheitsanforderungen

In [BCM00], [Bits02a] und [ABG04] sind Verfahren für die Aufstellung von Sicherheitsanforderungen beschrieben. Zunächst werden von Experten der jeweiligen Anwendungsdomäne Sicherheitsanforderungen auf einer globalen Ebene aufgestellt. Dabei werden die Automatisierungsaufgaben, die erforderlich sind, um Sicherheit zu gewährleisten, generell festgelegt. Diese werden dabei so allgemein formuliert, dass sie unabhängig von einem bestimmten Lösungskonzept sind. Diese globalen Sicherheitsanforderungen werden aufgrund von Expertenwissen, Normen (im Eisenbahnwesen sind dies z. B. insbesondere [EN50126], [EN50128] und [EN50129]) und Vorschriften (im Eisenbahnwesen sind dies z. B. die Eisenbahn-Bau- und Betriebsordnung (EBO), das Eisenbahnkreuzungsgesetz (EKrG) sowie die Straßenverkehrsordnung (StVO)) formuliert. Ein Beispiel einer globalen Sicherheitsanforderung für einen Bahnübergang im Funk-Fahr-Betrieb ist [GaPa99]: „*Ein Bahnübergang darf nur befahren werden, wenn der Zug die Meldung über die ordnungsgemäße Sicherung von dem Bahnübergang erhalten hat.*“

Die Sicherheitsanforderung muss nun für die jeweils gewählte fachtechnische Lösungskonzeption konkretisiert werden. Dabei muss in dem angeführten Beispiel insbesondere präzisiert werden, was mit „ordnungsgemäße Sicherung“ gemeint ist. Hierfür werden lösungsspezifische Sicherheitsanforderungen formuliert. Diese gelten für ein bestimmtes fachtechnisches Lösungskonzept eines Automatisierungssystems. Deshalb sind diese „systemspezifische Sicherheitsanforderungen“ [ABG02]. Solche systemspezifischen Sicherheitsanforderungen können z. B. im

Kontext der Beschreibung des Lösungskonzepts in natürlicher Sprache oder im Kontext einer Use-Case-Analyse [Bits02a] erstellt werden. Dabei werden sicherheitsrelevante Einzelheiten des Lösungskonzepts berücksichtigt. Für den angeführten Fall ist das z. B.: „*Wird das Rotlicht der Lichtzeichenanlage nicht eingeschaltet, so darf die Schrankenanlage des Bahnübergangs nicht geschlossen werden und der Status „gesichert“ nicht gemeldet werden.*“

Globale Sicherheitsanforderungen werden in der Regel schon sehr früh im Entwicklungsprozess bestimmt, bevor die fachtechnische Lösungskonzeption erstellt wird. Die Einhaltung von globalen Sicherheitsanforderungen lässt sich in Bezug auf ein konkretes Modell in der Regel nicht direkt prüfen. In einem Systemmodell herrscht eine detaillierte Systembetrachtung vor, während globale Sicherheitsanforderungen weitgehend lösungsunabhängig sind. Systemspezifische Sicherheitsanforderungen sind oft auch noch zu abstrakt, um sie für die formale Verifikation des Systemmodells anwenden zu können. Bezeichner und Systemgrößen sind allgemeiner gehalten als im Systemmodell. Deshalb ist es schwierig auf dieser Grundlage eine Sicherheitsanforderung so zu formalisieren, dass deren Erfüllung im Systemmodell formal überprüft werden kann [Bits00]. Aus diesem Grund werden, bevor mit der formalen Spezifikation der Sicherheitsanforderungen begonnen wird, die Sicherheitsanforderungen modellspezifisch formuliert. Das bedeutet, sie werden im Kontext des Modells der Systemfunktionen formuliert. Dies kann mithilfe der Betrachtungen in Sicherheitsanalysen (z. B. Fehler-Möglichkeits- und Einfluss-Analyse – FMEA [VDA96]) geschehen, die auf der Grundlage des Systemmodells durchgeführt werden (vgl. [ABG04], [Bits02b]). Dabei wird unter Berücksichtigung der globalen oder systemspezifischen Sicherheitsanforderungen bestimmt, welche möglichen Fehler im Systemmodell zu System-Gefährdungen führen. Beruhen deren Ursachen auf möglichen systematischen Fehlern im Systemmodell, so muss jeweils eine Sicherheitsanforderung aufgestellt werden, durch die das Vorhandensein eines solchen systematischen Fehlers durch funktionale Korrektheitsprüfung überprüft werden kann. In der Formulierung werden Modellvariablen (Aktions-, Ereignis- und Bedingungsvariablen) bei der Formulierung mitverwendet, da sonst die Gefahr von Fehlinterpretationen besteht, welche Variablen des Systemmodells durch die Formulierungen jeweils gemeint sind. Für das gewählte Beispiel ist eine entsprechende modellspezifisch formulierte Sicherheitsanforderung:

`Bahnübergang.Schrankenanlage_einschalten` *darf nur gelten, wenn die Vorleuchtzeit* `Bahnübergang.tl` *nach Auftreten von* `Lichtzeichenanlage.Gelblicht_einschalten` *abgelaufen ist.*

2.2.5 Vorgaben und Empfehlungen der Normen zur Formulierung funktionaler Sicherheitsanforderungen

Für sicherheitskritische Systeme hat sich mittlerweile die [IEC61508] auf internationaler, europäischer und nationaler Ebene [DINEN61508] etabliert (Klassifikation VDE 0803 im

deutschen Normenwerk). Sie hat zur Einführung einheitlicher Richtlinien auf internationaler Ebene und zur Ablösung nationaler Einzellösungen geführt. Es ist eine generische Norm für „Funktionale Sicherheit – Sicherheitssysteme“, aus der domänenspezifische Normen abgeleitet wurden und werden. Für die Prozessleittechnik ist dies mit [IEC61511] und für das Eisenbahnwesen mit [EN50126], [EN50128] sowie [EN50129] bereits geschehen. Aber auch in Domänen, in denen noch keine spezifische Norm erstellt wurde, hat sich die generische Norm bereits etabliert (z. B. Automobilbereich: seit dem 01.08.04 Ablösung aller einschlägigen nationalen Normen in Deutschland durch die IEC 61508). Technische Normen sind generell rechtlich nicht bindend, außer wenn sie in Gesetzen und Verordnungen zitiert werden. Diese Normen beschreiben jedoch den Stand der Technik [DKE03] in Bezug auf funktionale Sicherheit.

Für die Spezifikation von Sicherheitsanforderungen empfiehlt IEC 61508 im Fall von SIL 2-4 die formale Spezifikation. Dabei wird Temporallogik als Sprache zur formalen Spezifikation der Sicherheitsanforderungen herausgestellt. Sie wird als „Basis für die formale Demonstration“ der Einhaltung von Sicherheitsanforderungen bewertet, das bedeutet, dass auf der Grundlage einer formalen Spezifikation der Anforderungen in Temporallogik mit mathematischer Exaktheit überprüft werden kann, ob die aufgestellten Anforderungen erfüllt werden.

Allerdings berücksichtigen die Normen, dass die Beherrschung und damit der korrekte Gebrauch formaler Sprachen schwierig ist. Deshalb wird gefordert, dass im Falle der formalen Spezifikation, eine entsprechende Beschreibung in natürlicher Sprache angegeben werden muss. Diese Forderung ist jedoch in zweierlei Hinsicht problematisch:

1. Es ist nicht gewährleistet, dass die Spezifikation in natürlicher Sprache äquivalent zur formalen Spezifikation ist. Dadurch können bei der Interpretation der Sicherheitsanforderungen in der Entwicklung leicht Fehler entstehen.

2. Natürliche Sprache erlaubt Mehrdeutigkeiten, sodass die Sicherheitsanforderung entgegen der ursprünglichen Intention der formalen Spezifikation interpretiert werden kann.

2.3 Präzise Spezifikation von Anforderungen

2.3.1 Möglichkeiten zur Spezifikation

Grundsätzlich ist es sinnvoll, für die Spezifikation der Anforderungen eine andere Sprache als die für die Modellierung der Systemfunktionen zu gebrauchen. Dieses Vorgehen wird als *Dual-Language Approach* bezeichnet [Ostr97]. Für die Beschreibung der Systemfunktionen werden operationelle Sprachen verwendet, während für Anforderungen entsprechende deklarative Sprachen gebraucht werden [HKL+98]. Während für die Beschreibung der Systemfunktionen eine ausführbare vollständige Verhaltensbeschreibung wichtig ist, ist für die Beschreibung der Anforderungen eine kurze prägnante Beschreibung einzelner Relationen wesentlich.

Ein weiterer Vorteil des *Dual-Language Approach* ist die Ermittlung von Inkonsistenzen und damit von Spezifikationsfehlern. In [HKL+98] wird von der Verifikationsfallstudie [DDH+92] berichtet, in der durch die Korrektheitsprüfung sowohl im Modell der Systemfunktionen als auch in den Anforderungen Fehler ermittelt werden konnten. Dabei wurden die Systemfunktionen operationell durch ein Transitionssystem modelliert, während die Anforderungen in Temporallogik spezifiziert wurden.

Generell gibt es folgende Möglichkeiten für die Spezifikation von Anforderungen:

Spezifikation …

- in natürlicher Sprache
- mit semi-formalen textuellen oder grafischen Beschreibungsmitteln
- in Normsprache
- mit formalen textuellen oder grafischen Beschreibungsmitteln

Die Eignung natürlicher Sprache wurde bereits in Abschnitt 2.2.5 als unzureichend bewertet. Geschriebener natürlicher Sprache fehlt der Tonfall. Beim Sprechen können durch Betonungen und Abschwächungen Mehrdeutigkeiten vermieden werden. Das Kernproblem ist jedoch, dass natürliche Sprache an sich bereits mehrdeutig ist. Natürliche Sprache erlaubt unpräzise Formulierungen, und Formulierungen in natürlicher Sprache lassen verschiedene Interpretationen zu. Wenn Anforderungen für Automatisierungssysteme verschieden zur Intention des Spezifizierers interpretiert werden, können die Auswirkungen fatal sein. Das Automatisierungssystem kann aufgrund dessen entgegen der ursprünglichen Intention entwickelt werden. Dies kann hohe Kosten nach sich ziehen und im Fall von sicherheitskritischen Automatisierungssystemen Gefahrensituationen im Betrieb des Systems entstehen lassen. Im Folgenden ist eine unpräzise und deshalb mehrdeutige Sicherheitsanforderung an einen Bahnübergang formuliert: „*Nachdem das Eisenbahnfahrzeug den Bremswegabstand vor dem Bahnübergang erreicht hat und keine Statusmeldung über die ordnungsgemäße Sicherung des Bahnübergangs erhalten hat, muss der Bremsvorgang begonnen werden.*“

Mehrdeutig ist hier bereits das Wort “nachdem“ (vgl. [Bits03a]). Bedeutet es „unmittelbar nachdem“ oder liegt die Betonung darauf, dass es „danach“ geschieht? Wann wird im letzteren Fall jedoch nicht ausgedrückt, d. h. „nachdem“ im Sinne von „irgendwann nachdem“?

Semi-formale Beschreibungsmittel besitzen eine definierte vollständige Syntax, aber keine vollständig mathematische Basis [CJS98]. Ein semi-formales textuelles Beschreibungsmittel ist z. B. die Object Constraint Language (OCL) der Unified Modeling Language (UML). Die Semantik von Constraints, die in OCL formuliert werden können, ist nicht präzise definiert [RiGo98]. Ein semi-formales grafisches Beschreibungsmittel sind z. B. Sequenzdiagramme der UML. In der Semantik von Sequenzdiagrammen ist z. B. unklar, ob die spezifizierte Sequenz gelten muss oder ob sie nur ein Beispiel eines möglichen Verlaufs darstellt [Canv99]. Eine

präzise Spezifikation wird nur durch Normsprachen und formale Sprachen ermöglicht. In den folgenden beiden Abschnitten werden diese Spezifikationsarten beschrieben.

2.3.2 Spezifikation in Normsprachen

Ursachen für Missverständnisse sind häufig einzelne Begriffe. Deshalb wurde von [Ortn97] und [Schi97] das Konzept der Normsprachen für die Softwareentwicklung eingeführt, vgl. [Schn99, S. 132 ff.]. In Normsprachen wird zunächst in einem Lexikon festgelegt, welche Begriffe (sog. Fachwörter), die aus der natürlichen Sprache bekannt sind, in den Beschreibungen der Entwicklungsschritte der zu realisierenden Anwendung verwendet werden sollen. Diese Begriffe entstammen dem jeweiligen Anwendungs- bzw. Problembereich. Es wird klar definiert und abgegrenzt, was mit den Fachwörtern im Kontext der zu realisierenden Anwendung gemeint ist (Entwicklung einer genormten Terminologie, d. h. Begriffskonvention). Für zeitlich-logische Zusammenhänge ist z. B. die Klärung wichtig, ob der Begriff „wenn“ als Implikations-Relation („falls“) oder als temporale Relation („sobald“) gebraucht werden soll [Bits03a].

Eine Normsprache ist so konzipiert, dass sie für die Kommunikation mit „Alltagssprachen-Benutzern“ einsetzbar ist. Eine zentrale Forderung beim Aufbau des Lexikons der Normsprache ist eine „größtmögliche Kontextunabhängigkeit“ [Ortn97, S. 96]. Zudem müssen bei der Definition der genormten Terminologie defekte, ungeschickte Sprachregelungen nach folgenden Gesichtspunkten aufgedeckt und beseitigt werden [Ortn97, S. 32]:

1. ***Synonyme* müssen unter Kontrolle sein:** Wörter, welche dieselbe Bedeutung haben, müssen identifiziert werden. Beispiel: „auftreten“ und „sich ereignen“.

2. ***Homonyme* müssen beseitigt werden:** Wörter, die unterschiedliche Bedeutungen besitzen können, müssen identifiziert und eindeutig definiert werden. Beispiel: „Sicherheit“ kann *safety* bedeuten oder IT-Sicherheit, d. h. im Sinne von *security* verstanden werden.

3. ***Äquipollenzen* müssen aufgedeckt werden:** Äquipollenzen bestehen zwischen Wörtern, wenn die Bedeutungen von Worten so miteinander zusammenhängen, dass dieselbe Sache unter verschiedenen Blickwinkeln betrachtet, unterschiedlich bezeichnet wird. Beispiel: Eine „Aktion“ einer Systemkomponente kann in einer anderen Systemkomponente als „Ereignis“ wahrgenommen werden, das durch die Aktion ausgelöst wird.

4. ***Vagheiten* müssen geklärt werden:** Wörter, deren Bedeutung in der Alltagssprache nicht klar festgelegt ist, müssen identifiziert und eindeutig definiert werden. Es muss eine klare Abgrenzung des Begriffs erfolgen. Beispiel: „Wenn“ kann die Bedeutung von „falls“ oder „sobald“ besitzen.

5. ***Falsche Bezeichner* müssen ersetzt werden:** In der Alltagssprache werden oftmals unpassende Wörter gebraucht. Solche Wörter dürfen nicht in die genormte Terminologie integriert werden. Beispiel: Ein Ereignis „passiert“ nicht, sondern es tritt ein.

Die Definition von Wörtern geschieht durch Benennung der Wortart (z. B. Substantiv, Adjektiv), explizite Definition, eine Erläuterung, durch die Beschreibung der Beziehungen zu anderen Wörtern (z. B. Synonyme, Antonyme) und ein Musterbeispiel. Explizite Definition bedeutet, dass der für die Definition verwendete Ausdruck als Ersatz für den Begriff verwendet werden kann.

2.3.3 Formale Spezifikation

Die Vorteile formaler Spezifikation werden in [HeHe98] treffend ausgedrückt: "*Formal specification of the required system behaviour has many benefits. For example, a formal specification is unambiguous and precise, [...] and can help ensure that a specification is well-formed.*"

Die Spezifikation in formaler Sprache erlaubt die präzise Formulierung von Anforderungen. Dabei werden Beschreibungsmittel verwendet, die aufgrund einer eindeutigen Syntax und Semantik eine mathematisch exakte Beschreibung gestatten. Dadurch ist eine eindeutige Interpretation möglich. Zudem muss auf der Grundlage der Spezifikation eine mathematische Weiterverarbeitung möglich sein, sodass ein automatisierter bzw. automatischer Korrektheitsnachweis durchführbar ist (formale Methoden der Ebene 2 nach [Rush95]). Im folgenden Abschnitt wird auf die formale Spezifikation und Verifikation im Detail eingegangen.

2.4 Einführung in die formale Spezifikation und Verifikation von Anforderungen

2.4.1 Mathematische Logik

Mathematische Logik ist die allgemeine Theorie der folgerichtigen Aussagen und ihrer gültigen Beziehungen [ECS93]. Sie ist die Lehre bzw. Wissenschaft von der Struktur, den Formen, den Gesetzen des folgerichtigen Denkens und des Schließens aufgrund gegebener Aussagen [Moik02, S. 11]. Auf der Basis von Ausdrücken mit festgelegter Syntax und Semantik können Schlussfolgerungen mit mathematischer Exaktheit getroffen werden. Dies ist die Grundlage für die formale Verifikation. In den folgenden drei Unterkapiteln werden die grundlegenden Logik-Arten vorgestellt.

2.4.1.1 Aussagenlogik

Die Aussagenlogik behandelt die logische Bewertung von Aussagen. Komplexe Aussagen sind aus einfachen Aussagen als Teilen aufgebaut [KBW00, S. 211]. Beispiel: „Die Schrankenbäume sind geschlossen und das Rotlicht ist eingeschaltet." Eine Teilaussage des Beispiels ist: „Die Schrankenbäume sind geschlossen".

Aussagen und ihre Verknüpfungen besitzen einen Wahrheitswert. Eine Aussage ist wahr („1“) oder falsch („0“) [Schn99, S. 138]. Eine Wahrheitsfunktion ordnet einer Aussage den Wahrheitswert zu. Tabelle 2.1 zeigt die möglichen aussagenlogischen Verknüpfungen (Konnektoren), vgl. [HuRy00]. Dabei sind p und q selbst ebenso Aussagen (engl. *propositions*).

Tabelle 2.1: Symbole, Konnektoren und Begriffe der Aussagenlogik

Symbol	**Bezeichnung**	**Beispiel**	**Interpretation**
&, ∧, and	Konjunktion	$p \wedge q$	p und q
|, +, ∨, or	Disjunktion	$p \vee q$	p oder q
→	Implikation	$p \rightarrow q$	p impliziert q, wenn p dann auch q
¬, ~, !, not	Negation	$\neg p$	nicht p

Für die Kombination von Wahrheitsfunktionen entwickelte *Wittgenstein* die Wahrheitstafeln [KBW00, S. 211]. Tabelle 2.2 zeigt die Wahrheitstafel für die grundlegenden aussagenlogischen Verknüpfungen.

Tabelle 2.2: Wahrheitstafel für die aussagenlogischen Konnektoren

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$\neg p$
0	0	0	0	1	1
0	1	0	1	1	1
1	0	0	1	0	0
1	1	1	1	1	0

Zur äquivalenten Umformung von Ausdrücken gelten die folgenden Regeln:

- $p \vee q \equiv \neg(\neg p \wedge \neg q)$
- $p \rightarrow q \equiv \neg p \vee q$
- $1 \equiv \neg p \vee p$
- $1 \equiv \neg 0$

In der Elektrotechnik findet die Aussagenlogik in der Schaltalgebra Anwendung und gehört deshalb zu den Grundkenntnissen von Ingenieuren im Bereich der Automatisierungstechnik. Dabei wird in der digitalen Schaltungstechnik Pegel 0 zu „falsch“ und Pegel 1 zu „wahr“

zugeordnet. Die Aussagenlogik wird für den Schaltungsentwurf verwendet. Wahrheitsfunktionen werden mithilfe logischer Gatter realisiert.

Die Aussagenlogik allein eignet sich nicht zur Spezifikation von Anforderungen, da nur Aussagen formuliert werden können. Es kann nicht angegeben werden, in welchem Kontext eine Aussage gelten soll. Die ist mit der Prädikatenlogik und der Temporallogik möglich. Diese sind Erweiterungen der Aussagenlogik.

2.4.1.2 Prädikatenlogik

In der natürlichen Sprache werden Subjekte und Prädikate gebraucht. Prädikate ordnen Subjekten eine Eigenschaft zu [KBW00, S. 211]. Die Prädikatenlogik lässt die Analyse solcher Relationen zu. Im Gegensatz zur Aussagenlogik können mithilfe von Quantoren Aussagen über Mengen von Betrachtungseinheiten formuliert werden. Die Prädikatenlogik bedient sich der Aussagenlogik und erweitert diese durch Hinzunahme der Quantoren ∀ (Allquantor – „für alle“) und ∃ (Existenzquantor – „es gibt mindestens ein“) [ECS93, S. 383] sowie der Einführung von Funktionen, sog. „Prädikate“. Beispiel: `∃ e1, e2, e3, e4: (f(e1, e2, e3, e4) = defekt)`. Das bedeutet: „Es gibt mindestens eine Kombination der Sensorwerte (`e1, e2, e3, e4`) die dazu führt, dass das Automatisierungssystem `defekt` ist.“

Bei Automatisierungssystemen spielt, wie in Kapitel 2.1 beschrieben, die Zeit eine besondere Rolle. Automatisierungssysteme sind reaktive Systeme. Deshalb muss eine Zeitvariable `t` eingeführt werden, um Anforderungen ausdrücken zu können. Ein Beispiel einer Sicherheitsanforderung an ein pneumatisches Bremssystem ist folgendes [BiGu00]:

Formulierung in natürlicher Sprache: „*Ventil 1 (Einlassventil) und Ventil 2 (Auslassventil) dürfen niemals gleichzeitig geöffnet sein.*“

Formulierung in Prädikatenlogik:
```
∀ t ≥ 0: (¬(Ventil1(t) = offen &
          Ventil2(t) = offen))
```

2.4.1.3 Temporallogik

Temporallogik wurde Anfang der 60er Jahre von dem Philosophen *Prior* zur Formalisierung natürlicher Sprache entwickelt [Prio62]. In der Temporallogik wird die Aussagenlogik um eine Menge mit linear angeordneten Elementen – den Zeitpunkten – ergänzt. Aussagen der Temporallogik gelten unter Berücksichtigung der Zeit notwendiger- oder möglicherweise (vgl. [ECS93, S. 385]). *Pnueli* zeigte, dass die Anforderungen an reaktive Systeme formal in Temporallogik ausgedrückt werden können [AlHe92] [Pnue77]. *Gorski* stellte die Bedeutung von Temporallogik für die formale Spezifikation und Verifikation von Sicherheitsanforderungen heraus [Gors86].

[EsSc03] beschreibt die Computation Tree Logic (CTL) und die Linear-time Temporal Logic (LTL) als die temporallogischen Sprachen, die sich in der Praxis der formalen Verifikation durch Model-Checking durchgesetzt haben. Dies wird durch [Béra01, S.35] – einem aktuellen Standardwerk zum Thema Model-Checking – bestätigt: *„[…]LTL […] and CTL […] are the two most commonly used temporal logics in model checking tools.“* Diese beiden temporallogischen Sprachen werden im Folgenden erläutert:

Kurzeinführung in CTL

CTL erlaubt es, logische Ausdrücke zu bilden, die zeitliche Aussagen und Aussagen über Berechnungspfade (Ausführungspfade) in Berechnungsbäumen (*computation trees*) enthalten. Dadurch ist es möglich, Eigenschaften von Zustandsmodellen [LaGö99a, S. 120 ff.] auszudrücken, bei denen Variablen verschiedener Zustände miteinander in Relation stehen. Ein Berechnungsbaum (Zustandsraum des Zustandsmodells) stellt ausgehend von einem Startzustand alle möglichen Zustandsfolgen (Berechnungspfade) in einer Baumstruktur dar. In Abbildung 2.2 ist ein einfaches Zustandsmodell mit den Zuständen S1, S2 und S3 und dem zugehörigen Berechnungsbaum dargestellt. S1 ist der Startzustand.

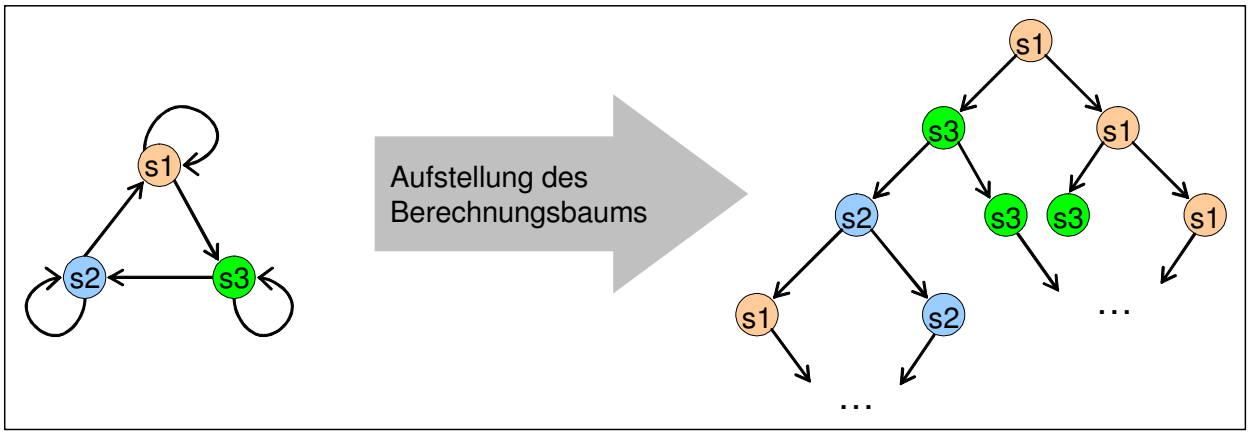


Abbildung 2.2: Berechnungsbaum eines Zustandsmodells

CTL-Audrücke sind aus einzelnen aussagenlogischen Ausdrücken zusammengesetzt. Die aussagenlogischen Ausdrücke stehen in Beziehung zu Bedingungen, Ereignissen, Aktionen, Zuständen oder Konfigurationen im Zustandsmodell. Diese aussagenlogischen Ausdrücke können durch Konnektoren der Aussagenlogik und durch CTL-Konnektoren miteinander in Beziehung stehen. Jeder CTL-Konnektor besteht aus einem Symbolpaar. Das erste Symbol ist ein Pfadquantor. Mit Pfadquantoren werden Aussagen über der Baumstruktur getroffen. Die möglichen Zeichen sind `A` und `E`. `A` bedeutet “along **A**ll paths” (auf allen Berechnungspfaden) und `E` bedeutet “along at least (there **E**xists) one path” (auf mindestens einem Berechnungspfad). Das zweite Symbol des CTL-Konnektor-Paars beschreibt die temporale Modalität, d. h. die zeitliche Aussagenfolge entlang eines Ausführungspfads (sog. Zeitquantor). Es gibt die Zeichen `X`, `F`, `G`, `U` und `W`, die “ne**X**t state”, “some **F**uture states”, “all future states (**G**lobally)”,

"**U**ntil" und "**W**eak Until" bedeuten, vgl. [HuRy00], [VSS]. "**U**ntil" und "**W**eak Until" treffen immer eine Aussage über zwei logische Ausdrücke: p `U` q. Im Unterschied zu `W` muss bei `U` die zweite Aussage, im Beispiel ist das q, irgendwann gelten.

Syntax (mit p für aussagenlogischen Ausdruck und ϕ `sowie` ψ für CTL-Ausdruck):

$$
\begin{aligned}
\phi \ ::= \ & p \ \mid \\
& (\neg\phi) \ \mid \ (\phi \vee \psi) \ \mid \ (\phi \wedge \psi) \ \mid \ (\phi \rightarrow \psi) \ \mid \\
& \mathbf{AF}\ \phi \ \mid \ \mathbf{EF}\ \phi \ \mid \ \mathbf{AG}\ \phi \ \mid \ \mathbf{EG}\ \phi \ \mid \ \mathbf{AX}\ \phi \ \mid \ \mathbf{EX}\ \phi \ \mid \\
& \mathbf{A}[\phi\ \mathbf{U}\ \psi] \ \mid \ \mathbf{E}[\phi\ \mathbf{U}\ \psi] \ \mid \ \mathbf{A}[\phi\ \mathbf{W}\ \psi] \ \mid \ \mathbf{E}[\phi\ \mathbf{W}\ \psi]
\end{aligned}
$$

Neben allen Äquivalenzen der Aussagenlogik gelten zur äquivalenten Umformung von Ausdrücken die im Anhang A beschriebenen Regeln.

Das Beispiel der Sicherheitsanforderung an ein pneumatisches Bremssystem wird in CTL wie folgt ausgedrückt. Dabei ist keine Variable für die Zeit erforderlich.

Formulierung in natürlicher Sprache: „*Ventil 1 und Ventil 2 dürfen niemals gleichzeitig geöffnet sein.*"

Formulierung in CTL: `AG ¬(Ventil1 = offen & Ventil2 = offen)`

Kurzeinführung in LTL

Der Unterschied von LTL zu CTL ist der, dass es keine Pfadquantoren gibt. Das bedeutet, dass in einem LTL-Ausdruck keine Aussagen über alternative Ausführungspfade getroffen werden können. Somit sind in LTL nur zeitliche Aussagen möglich.

Syntax (mit p für aussagenlogischen Ausdruck und ϕ für LTL-Ausdruck):

$$
\begin{aligned}
\phi \ ::= \ & p \ \mid \\
& (\neg\phi) \ \mid \ (\phi \vee \phi) \ \mid \ (\phi \wedge \phi) \ \mid \ (\phi \rightarrow \phi) \ \mid \\
& \mathbf{F}\ \phi \ \mid \ \mathbf{G}\ \phi \ \mid \ \mathbf{X}\ \phi \\
& [\phi\ \mathbf{U}\ \phi] \ \mid \ [\phi\ \mathbf{W}\ \phi]
\end{aligned}
$$

Das Sicherheitsanforderungs-Beispiel wird in LTL wie folgt ausgedrückt:

`G ¬(Ventil1 = offen & Ventil2 = offen)`

Durch die Verwendung der Zeitvariablen `t`, als Träger der globalen Systemzeit oder relativer Zeitrelationen, kann explizite Zeit in temporallogische Ausdrücke integriert werden [Bare93, S. 47].

2.4.2 Formale Verifikation

Der Nachweis der Korrektheit eines System- oder Softwaremodells gegenüber Anforderungen wird als „Verifikation" bezeichnet. Geschieht dieser Nachweis mit mathematischer Exaktheit, so

ist dies eine „formale Verifikation“. Generell können drei Verfahren zur automatisierten oder automatischen Verifikation unterschieden werden: Theorem-Proving (Theorembeweis), Proof-Checking (Beweisprüfung) und Model-Checking (Modellprüfung). Die zu diesen Verfahren gehörenden Softwarewerkzeuge heißen entsprechend Theorem-Prover (Theorembeweiser), Proof-Checker (Beweisprüfer) und Model-Checker (Modellprüfer).

2.4.2.1 Theorem-Proving und Proof-Checking

In einem mathematischen Beweis wird mithilfe von logischen Ableitungsregeln gezeigt, dass eine Behauptung (Theorem) in einem gegebenen Kontext allgemein gültig ist oder sich widerlegen lässt. Beim Theorem-Proving werden Suchalgorithmen, Heuristiken und Hinweise des Bedieners verwendet, um einen vollständigen korrekten Beweis herzuleiten [Moik02, S. 17]. Mithilfe eines Proof-Checker kann nachgewiesen oder widerlegt werden, dass alle Beweisschritte gemäß den Ableitungsregeln der verwendeten Logik erfolgt sind (vgl. [Stor96, S. 286]). In der Praxis werden beide Verfahren kombiniert und die Begriffe „Proof-Checker“ und „Theorem-Prover“ häufig synonym gebraucht. Beispiele sind VSE II (Verification Support Environment) [Göhn95] [CGM97] [BDI+99], der induktive Theorem-Prover INKA (Induction Prover Karlsruhe) [AHMS99], der deduktive Theorem-Prover KIV (Karlsruhe Interactive Verifier) [BRSS99], PVS (Prototype Verification System) [ORSS98] und Isabelle/HOL [NPW02].

2.4.2.2 Model-Checking

Das Model-Checking ermöglicht eine vollständig automatische Nachweisführung. Dabei kann die Einhaltung von Anforderungen in nebenläufigen automatenbasierten Systemen überprüft werden. Wie in Abbildung 2.2 schematisch dargestellt ist, wird der Zustandsraum des Modells der Systemfunktionen vollständig exploriert. Der Model-Checker überprüft, ob es im Zustandsraum erreichbare Zustände oder Zustandsfolgen gibt, durch welche die aufgestellten Anforderungen verletzt werden. Das Modell der Systemfunktionen muss dafür als operationelles Modell durch einen Automaten spezifiziert sein. Voraussetzung für das Model-Checking ist, dass das zu überprüfende Modell zustandsendlich ist. Für die Spezifikation der Anforderungen erfordern Model-Checker die Formulierung in einer temporallogischen Sprache [DAC99].

Ein Problem des Model-Checking ist die Explosion des Zustandsraums bei der Exploration. Um diesem Problem zu begegnen, wurde beispielsweise mit der Technologie des symbolischen Model-Checking eine Alternative zur direkten Exploration des Zustandsraums entwickelt. Anstatt den Zustandsraum explizit aufzubauen, wird eine symbolische Repräsentation gebraucht, sodass wenig Arbeitsspeicher für die Berechnung und Überprüfung des Zustandsraums erforderlich ist [BCM+90] [McMi92] [Ping02]. So konnten bereits Systeme mit 10^{100} erreichbaren Zuständen verifiziert werden [BCL91].

Sehr effiziente und weit verbreitete Model-Checker (vgl. [Béra01], [Espa03], [Step00]) sind Spin [Holz03], SMV (Symbolic Model Verifier) [McMi92] und VIS (Verification Interacting with Synthesis) [Bray96]. Durch die Verwendung von Abstraktionstechniken kann der Zustandsraum beim Model-Checking zudem erheblich reduziert werden. Dadurch konnten bereits Systeme mit 10^{1300} erreichbaren Zuständen verifiziert werden [CGL94]. Ein ingenieurgerechtes effektives Verfahren zur Reduktion des Zustandsraums wurde in [Ping02] entwickelt, das vollständig automatisiert ist. Eine Zustandsraumexplosion wird durch Verlagerung globalen Verifikationsaufwands auf lokale Module vermieden. Auf der Basis der Modul-Architektur werden aus globalen Anforderungen modulspezifische algorithmisch abgeleitet (algorithmische Dekomposition von Anforderungen). Die Module werden gegenüber diesen lokalen Anforderungen durch Model-Checking verifiziert.

Im Folgenden wird nun noch dargelegt, welche temporallogischen Sprachen für die Spezifikation von Anforderungen bei den Model-Checker im akademischen Bereich aktuell Verbreitung gefunden haben: Die bereits erwähnten Model-Checker SMV, Spin und VIS erfordern die Spezifikation von Anforderungen in CTL oder LTL (CTL: SMV und VIS; LTL: SMV und Spin). Weitere Model-Checker, welche die Verifikation von Anforderungen in CTL sowie in LTL erlauben, sind [DSSZ], [PROD] und [PEP]. Dabei können Anforderungen für die Verifikation mit PEP auch in L_1 spezifiziert werden, was jedoch wiederum eine Untermenge von CTL ist, s. [Espa98], [Grav95]. Weitere Model-Checker zur Verifikation von Anforderungen in CTL sind [LoLA], [MCB] und [INA]. [Maria] und [Murphi] sind Model-Checker zur Verifikation von Anforderungen in LTL und Mona für Anforderungen in M2L-Str, was wiederum eine LTL-basierte Sprache ist [KlMø01]. Die Model-Checker UPPAAL2K [Béra01], Kronos [Béra01] und [Raven] verfügen jeweils über eine eigene Spezifikationssprache, welche die Spezifikation von expliziter Zeit konzeptionell mit einschließt. Diese Sprachen basieren wiederum jeweils auf CTL, s. [ABB+01], [Yovi02], [FMR00a], [RuKr97]. Wie aus dieser Auflistung hervorgeht, sind die Sprachen CTL und LTL die für das Model-Checking relevanten Sprachen. Eine Ausnahme bildet lediglich der Model-Checker [Mucke] (vgl. [Bier97]), der bei wissenschaftlichen Arbeiten in der theoretischen Informatik Anwendung findet. Dieser erlaubt die Verifikation von Anforderungen im μ-Kalkül (temporallogische Sprache) [Bier97]. Insgesamt wird somit bestätigt, dass sich LTL und CTL als temporallogische Sprachen für die formale Verifikation durch Model-Checking in der Praxis durchgesetzt haben, vgl. Abschnitt 2.4.1.3.

In diesem Kapitel wurden die Grundlagen zu den Themen Automatisierungssysteme, Sicherheit, Spezifikation von Anforderungen sowie formale Spezifikation und Verifikation behandelt. Die für die folgenden Ausführungen wichtigen Grundbegriffe wurden eingeführt. Zudem wurde auf den Stand der Technik zur Spezifikation von Sicherheitsanforderungen eingegangen, wie er von den Normen definiert wird.

3 Aufstellung der Konzept-Anforderungen für ein Verfahren zur anwendbaren Spezifikation von Sicherheitsanforderungen

Auf der Basis der beschriebenen Grundlagen werden in diesem Kapitel die Konzept-Anforderungen für ein Verfahren zur anwendbaren präzisen Spezifikation von Sicherheitsanforderungen abgeleitet. Die im letzten Kapitel gewonnenen Erkenntnisse in Bezug auf die Bedeutung von Sicherheitsanforderungen, in Bezug auf die Möglichkeiten zur Spezifikation sowie hinsichtlich formaler Spezifikation und Verifikation werden ausgewertet. Anhand dieser Informationen werden Schlussfolgerungen für ein Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme gezogen. Auf der Grundlage dieser Schlussfolgerungen werden die Konzept-Anforderungen aufgestellt und definiert.

3.1 Formale Spezifikation und Verifikation von Sicherheitsanforderungen für Automatisierungssysteme

In der industriellen Praxis erfolgt die Bearbeitung eines Automatisierungsprojekts in der Regel durch verschiedene Systementwickler (Ingenieure) in Teamarbeit [LaGö99b, S. 2]. Bei der Entwicklung sicherer Automatisierungssysteme ist Teamarbeit schon allein aufgrund der Forderung in den Normen notwendig. Für die Sicherheitsanforderungsstufen (SIL) 3 sowie 4 müssen die Validierung und wahlweise zusätzlich noch die Verifikation von solchen Entwicklern durchgeführt werden, die bei den anderen Entwicklungstätigkeiten nicht beteiligt waren [WWS02]. Deshalb ist es von besonderer Bedeutung, dass die Arbeitsergebnisse eines Ingenieurs auch von anderen Ingenieuren verwendet und dabei korrekt verstanden werden.

Zudem ist zu berücksichtigen, dass für Automatisierungssysteme Wartungs- und Pflegearbeiten erforderlich sind, um das System einerseits funktionsfähig zu erhalten und es andererseits an Änderungen des technischen Prozesses und geänderte Randbedingungen anzupassen [LaGö99b, S. 5-6]. Dazu sind Korrektur-, Änderungs- und Erweiterungsarbeiten erforderlich. Für diese Tätigkeiten sind das Verständnis und die Nachvollziehbarkeit der Entwicklungsarbeiten notwendig.

Im Falle von Sicherheitsanforderungen ist deshalb die eindeutige Interpretierbarkeit von besonderer Bedeutung, da es bei der Umsetzung der Anforderungen um die Vermeidung von Gefahren geht. Werden Sicherheitsanforderungen missverstanden und das eigentliche Anliegen der Anforderungen nicht erkannt, so kann dies zu der Entwicklung und Inbetriebnahme von Automatisierungssystemen führen, von denen Gefahren ausgehen. Daraus resultiert die erste

Konzept-Anforderung (KA) für ein Verfahren zur anwendbaren formalen Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme:

/KA1/ Präzise Spezifikation: *Funktionale Sicherheitsanforderungen müssen mithilfe des Verfahrens präzise spezifiziert werden können, sodass sie eine eindeutige Interpretation erlauben.*

Da es um die Vermeidung von Gefahr geht, ist der Einsatz besonderer Maßnahmen, die einen gewissen Aufwand mit sich bringen, gerechtfertigt. Für sicherheitskritische Automatisierungssysteme gibt es zudem die Besonderheit der Zulassung durch eine Aufsichtsbehörde [LaGö99a, S. 362]. Hierzu muss ein sog. Sicherheitsnachweis erbracht werden. Um dabei die Sicherheit des Automatisierungssystems zu begründen, ist der Einsatz besonderer Methoden und Techniken erforderlich. Auch bei der Überprüfung des Sicherheitsnachweises durch Gutachter oder die Genehmigungsbehörde ist die eindeutige Interpretierbarkeit der funktionalen Sicherheitsanforderungen von besonderer Bedeutung.

Durch die Formulierung der Sicherheitsanforderungen im Kontext des funktionalen Modells werden die Semantik der Sicherheitsanforderungen [Moik02, S. 93] und die Beziehung der Sicherheitsanforderungen zueinander festgelegt. Wird die Korrektheit des Modells der Systemfunktionen in Bezug auf die Sicherheitsanforderungen gezeigt, so wird dadurch die Grundlage für die Entwicklung eines sicheren Automatisierungssystems gelegt.

Die Spezifikation in formaler Sprache erlaubt die eindeutige Formulierung von Sicherheitsanforderungen. Für nicht-sicherheitsrelevante Anforderungen ist der Nutzen der formalen Spezifikation und Verifikation aufgrund des Zusatzaufwands aus ökonomischen Gesichtspunkten fraglich [Bits03a]. Für die Entwicklung sicherer Automatisierungssysteme ist der Aufwand der Durchführung formaler Verifikation für den Nachweis sicherheitsbezogener Korrektheit gerechtfertigt, weil dadurch das Vertrauen in die sichere Funktion eines Automatisierungssystems wesentlich erhöht werden kann, vgl. [Göhn95]. Zudem wird durch die formale Verifikation die konsistente Formulierung der Sicherheitsanforderungen überprüft. Sind die Formulierungen verschiedener Sicherheitsanforderungen widersprüchlich, können nicht alle Sicherheitsanforderungen durch das Modell der Systemfunktionen erfüllt sein. Insgesamt führt dies zur zweiten Konzept-Anforderung:

/KA2/ Formal verifizierbare Spezifikation: *Funktionale Sicherheitsanforderungen müssen mithilfe des Verfahrens so spezifiziert werden können, dass sie für den Nachweis sicherheitsbezogener Korrektheit formal verifizierbar sind.*

Durch die Normen werden die in diesem Abschnitt ausgeführten Konzept-Anforderungen bestätigt (vgl. Abschnitt 2.2.5).

3.2 Anwendbare formale Spezifikation und Verifikation

Ausgangspunkt dieser Arbeit ist, dass die Sicherheitsanforderungen hergeleitet und in informeller Form im Kontext eines System- oder Softwaremodells aufgestellt wurden. Dabei geht es um die korrekte Formalisierung der Sicherheitsanforderungen im Hinblick auf die Durchführung einer formalen Verifikation der Sicherheitsanforderungen.

Damit das Verfahren für Automatisierungssysteme ohne Einschränkungen anwendbar ist, müssen alle funktionalen Sicherheitsanforderungen, die für Automatisierungssysteme formulierbar sind, formalisiert werden können. Solche Formulierungen haben die Verhinderung von Systemsituationen zum Ziel, von denen Gefahren ausgehen, oder die Verhinderung von sicherheitskritischen Folgen, die durch das Erreichen bestimmter Systemsituationen ausgelöst werden (vgl. Abschnitt 2.2.3). Die dritte Konzept-Anforderung lautet:

/KA3/ Vollständige Spezifizierbarkeit: *Alle funktionalen Sicherheitsanforderungen, die für Automatisierungssysteme notwendig sind, müssen mithilfe des Verfahrens spezifizierbar sein. Die Spezifikation unterschiedlicher Arten von Sicherheitsanforderungen muss mithilfe des Verfahrens adäquat behandelt werden können.*

In der industriellen Praxis hat die Verwendung formaler Methoden zur Anforderungsspezifikation sowie zur Durchführung von Korrektheitsnachweisen bisher kaum Verbreitung gefunden (s. [Heit98a], [Rush00], [Bits02a], [Moik02]). Als ein entscheidender Grund hierfür werden Probleme mit der formalen Anforderungsspezifikation gesehen (s. [Béra01, S. 28], [Bits03a], [DaHa99], [DAC99], [FMR00a], [Heit98b]), während es für die Erstellung des operationellen Modells sowie für die Nachweisführung heutzutage anwendbare und nachvollziehbare Techniken gibt.

Mit dem Model-Checking gibt es heutzutage eine Verifikationstechnik, die einfach verwendbar ist. Fallstudien, in denen das Modell eines Softwaresystems einerseits mithilfe des Theorem-Provers VSE II und andererseits mit dem Model-Checker SMV verifiziert wurde (s. [Bits99], [Büch00]), zeigen, dass der Aufwand und der Anspruch der Modellierung für beide Verifikationsverfahren vergleichbar sind. Die Verifikation mit dem Model-Checker stellte sich jedoch wesentlich einfacher dar, da die formale Prüfung vollautomatisch durchgeführt wird. Dies wird auch durch [Moik02, S. 116 ff.] bestätigt. *Moik* bewertete Model-Checking als ingenieurgerecht und somit für Ingenieure in der industriellen Praxis anwendbar.

Die Erstellung des formalen funktionalen Modells ist nach [Moik02, S. 118] sowie den Erfahrungen in [Bits99] und [Büch00] einfach erlernbar und nach einer kurzen Einarbeitungszeit für Ingenieure durchführbar. Zudem existieren heute zahlreiche Ansätze, in denen gebräuchliche Notationen formalisiert wurden und die formale Verifikation auf der Basis von Modellen herkömmlicher Entwicklungswerkzeugen durchgeführt wird, z. B. UML-Notation [Arab05] [Arab03] [Bits02a] [KnMe02] mit verschiedenen Werkzeugen (Rhapsody (I-Logix) [DaWe03],

Software through Pictures (Aonix) [ABG04] [Faud04]), SFC-Notation (Sequential Function Charts) mit verschiedenen Werkzeugen [BHLE04] [HLB03], Petrinetz-Notationen mit verschiedenen Werkzeugen [Espa04] [Béra01] sowie Block- und Zustandsdiagramme mit den Werkzeugen ASCET-SD (ETAS) [BBB+04] [DSW+03], MATLAB®/Stateflow® (The MathWorks) [BaPi04] und Statemate (I-Logix) [BBB+04] [DaHa99]. Die Modelle werden für die formale Verifikation in die Eingangssprache des verwendeten Verifikationswerkzeugs transformiert.

Wesentlich für anwendbare formale Methoden ist zudem die klare Beschreibung der Anwendung der Methoden im Vorgehen für die Entwicklung von Automatisierungssystemen. Dazu werden in [Moik02] geeignete Ansätze aufgezeigt.

Damit das Verfahren zur Spezifikation der Sicherheitsanforderungen für die ingenieurmäßige Lösung von Automatisierungsaufgaben in der industriellen Praxis praktikabel ist, muss also auf die einfache und effiziente Anwendbarkeit besonderer Wert gelegt werden. Das Verfahren muss anwendbar sein (vgl. [Moik02, S. 18 f.]). Das bedeutet, dass es Ingenieure bei der korrekten Erfüllung ihrer Aufgabe in der Entwicklung und Wartung von Automatisierungssystemen unterstützt. Wesentlich für die Anwendbarkeit und Akzeptanz in der industriellen Praxis ist, dass Entwicklungskosten gering gehalten werden. Für den vorliegenden Kontext spielen die Kostenfaktoren *erforderliche Ingenieurleistung für die Anwendung der Methode*, *Art des Einsatzes der Ingenieure* (das bedeutet z. B., dass Ingenieure für Ingenieurtätigkeiten eingesetzt werden und automatisierbare Abläufe und Tätigkeiten automatisiert werden), *Schulungsaufwand* sowie *Erforderlichkeit von Spezial-Experten* (wie z. B. für Theorem-Proving) eine Rolle.

Insgesamt muss bei einem für die industrielle Praxis anwendbaren Verfahren auf die effiziente Anwendbarkeit (Handhabbarkeit) besonderer Wert gelegt werden. Dabei geht es um Kriterien, die menschlichen Fehlern entgegenwirken und ein wirtschaftliches Arbeiten ermöglichen. Dies sind *einfache Erlernbarkeit und Anwendbarkeit*, *hoher Automatisierungsgrad und Werkzeugunterstützung*, *Mehrfachverwendung* sowie *Verständlichkeit und Nachvollziehbarkeit*.

Für das Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen resultieren daraus die Konzept-Anforderungen 4 und 5:

/KA4/ Handhabbarkeit: *Die Spezifikation funktionaler Sicherheitsanforderungen muss einfach und effizient handhabbar sein.*

Durch die Einfachheit des Verfahrens sollen Spezifikationsfehler vermieden werden. Bei der Anwendbarkeit des Verfahrens ist zudem auf einen vertretbaren Spezifikationsaufwand zu achten. Der Spezifikationsprozess muss durch ein Werkzeug unterstützt werden, das automatisierbare Vorgehensschritte umsetzt. Die Mehrfachverwendbarkeit der Spezifikationsergebnisse sollte unterstützt werden, d. h. die Verwendung von Spezifikationen in anderen Projekten [ScEi99]. Damit das Verfahren für die Spezifikation funktionaler Sicherheitsanfor-

derungen einfach erlernbar ist, müssen Methoden und Vorgehen der Ausbildung von Automatisierungstechnik-Ingenieuren gerecht sein.

Wie bereits in Abschnitt 3.1 ausgeführt wurde, müssen Sicherheitsanforderungen von verschiedenen Personengruppen interpretiert werden. Dabei ist nicht nur die eindeutige Interpretierbarkeit wichtig. Die Interpretierbarkeit der Sicherheitsanforderungen muss zudem einfach sein, sodass nach Möglichkeit Interpretationsfehler vermieden werden. Des Weiteren müssen in der Teamarbeit, bei Wartungs- und Pflegearbeiten sowie im Zulassungsprozess die Spezifikationsschritte nachvollzogen werden können. Insbesondere Wartungsingenieure müssen die Spezifikation schnell verwenden, verstehen und ggf. auch adaptieren können. Deshalb ist auch die einfache Nachvollziehbarkeit von besonderer Bedeutung. Für eine Anwendung in der industriellen Praxis ist zudem förderlich, dass das Interpretations-Verfahren einfach erlernbar ist und automatisierbare Schritte zur Interpretation durch Werkzeuge unterstützt werden. Dies kann in der 5. Konzept-Anforderung zusammengefasst werden:

/KA5/ Verständlichkeit und Nachvollziehbarkeit: *Funktionale Sicherheitsanforderungen müssen einfach verständlich und die Spezifikation nachvollziehbar sein.*

/KA4/ und /KA5/ können auch unter dem Begriff **Anwendbarkeit** zusammengefasst werden. Im Folgenden wird zusammengefasst, welche Bedeutung die Anwendbarkeit für ein Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen hat:

- Systementwickler (Ingenieure) sollen bei der Vermeidung von Spezifikations- und Interpretationsfehlern unterstützt werden.
- Bei der Anwendbarkeit des Verfahrens ist auf einen vertretbaren Spezifikations- und Interpretationsaufwand zu achten.
- Das Verfahren muss durch ein Werkzeug unterstützt werden, das automatisierbare Vorgehensschritte umsetzt.
- Die Mehrfachverwendbarkeit von Spezifikationsergebnissen soll unterstützt werden.
- Das Konzept zur Spezifikation und Interpretation muss einfach erlernbar sein.
- Das Verfahren schließt eine klare Vorgehensweise mit ein.

In diesem Kapitel wurden die Anforderungen für ein Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme hergeleitet. Diese Konzept-Anforderungen wurden ausformuliert und deren Bedeutung herausgestellt. Auf der Grundlage dieser Anforderungen kann mit Berücksichtigung des aktuellen Stands der Wissenschaft der geeignete Verfahrens-Ansatz entwickelt werden.

In der Logik kam es mir wunderlich vor, dass ich diejenigen Geistesoperationen, die ich von Jugend auf mit der größten Bequemlichkeit verrichtete, so auseinanderzerren, vereinzeln, gleichsam zerstören sollte, um den rechten Gebrauch derselben einzusehen.

J. W. v. Goethe, Dichtung und Wahrheit, 6. Buch

4 Stand der Forschung für die anwendbare formale Spezifikation von Anforderungen in Temporallogik

Temporallogik eignet sich zur präzisen und formal verifizierbaren Spezifikation von Anforderungen für Automatisierungssysteme. Allerdings ist die Verwendung dieser Spezifikationssprache mit erheblichen Schwierigkeiten verbunden. Dies wird im ersten Unterkapitel erläutert. Anschließend werden ausgehend von den aufgestellten Konzept-Anforderungen unterschiedliche Ansätze für eine anwendbare formale Spezifikation von Anforderungen für Automatisierungssysteme in Temporallogik erörtert. In der Forschung wurden verschiedene Ansätze entwickelt, die eine Spezifikation und Verifikation temporallogischer Ausdrücke erleichtern sollen. Solche Ansätze wurden identifiziert, durch die Ableitung von Ansatzarten gruppiert und bewertet. Dabei werden sowohl die einzelnen Ansätze sowie das Potenzial erörtert, das in den verschiedenen Ansatzarten enthalten ist. Abschließend werden die Ansatzarten miteinander verglichen, und es werden Schlussfolgerungen für die Entwicklung eines Verfahrens zur anwendbaren Spezifikation von Sicherheitsanforderungen gezogen.

4.1 Schwierigkeiten bei der formalen Spezifikation mit Temporallogik

In Abschnitt 2.4 wurde Temporallogik als geeignete formale Sprache zur Spezifikation von Anforderungen für Automatisierungssysteme beschrieben. In solchen Anforderungen müssen zeitliche Relationen ausgedrückt werden können. Dies ist in Temporallogik mit formaler Exaktheit möglich, sodass eine formale Verifikation durchführbar ist. Temporallogik ist die Grundlage, um mit dem in Abschnitt 3.2 als ingenieurgerecht herausgestellten formalen Verifikationsverfahren Model-Checking zeitbehaftete Aussagen überprüfen zu können [HuRy00, S. 150]. Wie in Abschnitt 2.2.5 ausgeführt wurde, wird auch in den Normen für funktionale Sicherheit Temporallogik als Sprache zur formal verifizierbaren Spezifikation der Sicherheitsanforderungen herausgestellt. In den Normen wird zudem jedoch aufgezeigt, dass die Beherrschung und damit der korrekte Gebrauch formaler Sprachen schwierig ist. Temporallogik ermöglicht die präzise und formal verifizierbare Spezifikation von Sicherheitsanforderungen, allerdings treten bei der Anwendung von Temporallogik leicht Fehler auf. In [HKL+98] wird von der Model-Checking-Fallstudie [DDH+92] berichtet, in der durch das Model-Checking nicht nur

Fehler im Modell der Systemfunktionen aufgedeckt wurden, sondern insbesondere auch Fehler in den in Temporallogik formulierten Anforderungen. Solche Fehler bei der Spezifikation der Anforderungen sind auf die schwierige Verwendung von Temporallogik zurückzuführen. In [Holt99] werden diese Schwierigkeiten so begründet: "*[…] many target users of verification technology are not logicians, although they may well have clear and precise intuitions about the properties they wish to verify.*"

Eine Ursache für die geringe Verbreitung der Anwendung formaler Verifikation liegt in der Schwierigkeit der Verwendung von Temporallogik für Entwickler von Automatisierungssystemen, die in der Regel Ingenieure ohne detaillierte Kenntnisse in höherer Logik sind, vgl. z. B. [Béra01, S. 28], [Bits00], [Bits01], [BiGö02], [DAC99], [DaHa99], [FMR00a]. In [Heit98a] wird diese Schwierigkeit jedoch keineswegs auf Ingenieure beschränkt, sondern auch auf Experten für formale Methoden bezogen: „*Expressing certain properties in Temporal Logic is complex and error-prone, not only for practitioners but for formal methods experts as well.*"

Temporallogische Ausdrücke sind leicht unübersichtlich, sodass Aussagen temporallogischer Ausdrücke nicht vollständig von Entwicklern erfasst und verstanden werden [Bare93, S. 47] [Pere95, S. 32]. Ein wesentlicher Grund für diese Schwierigkeiten ist in der Beherrschung der komplexen und ausdrucksmächtigen Semantik von Temporallogik zu sehen. Im nächsten Unterkapitel werden Ansätze des aktuellen Stands der Wissenschaft aufgezeigt, in denen diesen Schwierigkeiten begegnet wird. Einzelne Ansätze werden in Ansatzarten kategorisiert und beschrieben. Das Potenzial der verschiedenen Ansatzarten wird bewertet.

4.2 Ansätze zur anwendbaren formal verifizierbaren Spezifikation von Anforderungen

Um die in diesem Unterkapitel beschriebenen Ansätze und deren Unterschiede zu verdeutlichen, wird die folgende Sicherheitsanforderung an einen Bahnübergang mithilfe der einzelnen Ansätze spezifiziert: „*Ist bei einem Bahnübergang eine Störung der Schrankenanlage aufgetreten, so darf dieser im automatischen Betrieb solange nicht erneut von einem Eisenbahnfahrzeug befahren werden, bis der Bahnübergang repariert wurde.*"

Dabei soll ein Systemmodell mit den booleschen Variablen `stoerung`, `befahren` und `reparieren` angenommen werden. Ist `stoerung = 1` so hat dies die Bedeutung, dass eine Störung der Schrankenanlage vorliegt. Bei `befahren = 1` wird der Bahnübergang im automatischen Betrieb des Bahnübergangs erneut befahren. Wurde der Bahnübergang repariert, so ist `reparieren = 1`. Somit lautet die modellspezifische Sicherheitsanforderung: „*Nach dem Auftreten von* `stoerung = 1` *muss* `befahren = 0` *solange gelten, bis* `reparieren = 1` *eintritt.*" Diese Sicherheitsanforderung wird in den folgenden Abschnitten „Referenzanforderung" genannt.

4.2.1 Verwendung von Textbausteinen in einer Terminologie der natürlichen Sprache

4.2.1.1 Strukturierter Text von Flake und Müller

In [RFP00] wird unterstrichen, dass temporallogische Formeln schwer zu formulieren und zu verstehen seien. Als eine Ursache hierfür wird gesehen, dass Temporallogik „viel umfassender als notwendig“ ist. Deshalb wird der schwierigen Spezifikation in Temporallogik durch die Einführung von Textbausteinen zur Spezifikation begegnet [FMR00a] (vgl. [BiGö02]). Der Entwickler muss nicht mehr direkt in Temporallogik spezifizieren, sondern er verwendet Textbausteine für temporallogische Grundkonstrukte. Ein Textbaustein ist in natürlicher Sprache formuliert und besitzt jeweils eine temporallogische Bedeutung. Ausdrücke werden nach Reihenfolgeregeln durch die Aneinanderreihung von Textbausteinen mithilfe von Werkzeugunterstützung gebildet.

Für die Referenzanforderung „*nach dem Auftreten von* `stoerung = 1` *muss* `befahren = 0` *solange gelten, bis* `reparieren = 1` *eintritt*“ ist die passende Formulierung mit strukturiertem Text in Abbildung 4.1 dargestellt. Dabei wird eine Aussage über die Sequenz `A` gemacht, die niemals ausgeführt werden darf. Die Sequenz `A` umfasst `stoerung = 1`, `reparieren = 0` und `befahren = 1`, d. h. der Bahnübergang wird nach einer Schrankenstörung befahren, ohne dass eine Reparatur stattgefunden hat. Somit ist in diesem Fall eine Umformung der Formulierung erforderlich. Im Gegensatz zur ursprünglichen Formulierung muss hier das unerwünschte Verhalten statt dem erforderlichen spezifiziert werden.

It is at all times	true that	a strong sequence *A* with interval [0,∞]	does not occur.

Abbildung 4.1: Spezifikation der Referenzanforderung mit strukturierten Textbausteinen

Strukturiert-natürlichsprachliche Sätze sind die Grundlage zur Spezifikation einer Untermenge von temporallogischen Ausdrücken [RFP00]. Im Gegensatz zum direkten Gebrauch von Temporallogik ist die Notation des strukturierten Textes den Entwicklern vertraut. Das Erlernen der Anwendung ist weniger kompliziert, da die Bedeutung der Textbausteine einfacher erfasst werden kann, als Kombinationen temporallogischer Operatoren. Von den Entwicklern des Verfahrens wird die Anwendbarkeit jedoch dahingehend eingeschränkt, dass strukturiert-natürlichsprachliche Sätze leicht „länglich“ sind und „künstlich […] klingen“ [RFP00].

4.2.1.2 PROSPER-System von Holt, Klein und Grover

Ein ähnliches Verfahren wird in [Holt99] und [HoKl99] vorgestellt[2] (vgl. [BiGö02]). Ausgangspunkt ist, dass viele Software- und Hardwareentwickler keine Fachexperten in formaler Logik

[2] In [HoKl99] werden dabei ähnliche Verfahren erwähnt, für die hier stellvertretend nur auf das in [HoKl99] vorgeschlagene eingegangen wird.

sind. Ihnen muss die Verwendung formaler Verifikationswerkzeuge zugänglich gemacht werden. Die Intention ist die Bereitstellung einer durch Model-Checking formal verifizierbaren Spezifikationssprache, die minimalen Aufwand für das Erlernen erfordert [GHKM00]. Model-Checking wird in [Holt99] und [HoKl99] als vielversprechende Technik zur Verifikation herausgestellt, die um ein anwendbares Spezifikationsverfahren ergänzt werden müsse.

Als Lösungsansatz wird eine Untermenge der natürlichen englischen Sprache zur Spezifikation von Anforderungen zugelassen. Ebenso wird eine Untermenge der englischen Grammatik verwendet. Mithilfe eines Softwarewerkzeugs sollen Mehrdeutigkeiten und Vagheiten in der Formulierung von Anforderungen interaktiv aufgedeckt und verbessert werden [GHKM00]. Nachdem eine Anforderung durch interaktive Dialoge korrekt und geeignet formuliert wurde, findet eine werkzeuggestützte Transformation nach Temporallogik statt. Die Referenzanforderung „*nach dem Auftreten von* `stoerung = 1` *muss* `befahren = 0` *solange gelten, bis* `reparieren = 1` *eintritt*“ kann mit diesem Ansatz folgendermaßen spezifiziert werden, vgl. [Tidh00]: „After `stoerung` becomes active `befahren` should not become active until `reparieren` becomes active“.

In [Holt99] werden jedoch Schwierigkeiten mit der Identifikation von Mehrdeutigkeiten und hinreichender Interaktivität eingeräumt. Die Erkennung von Mehrdeutigkeiten und die Verarbeitung kontextabhängiger Ausdrücke sind jedoch für die korrekte Transformation wesentlich (vgl. [FMR00b]). Für die Ermittlung korrekter Formulierungen, die auch für die Transformation in Temporallogik geeignet sind, sind unter Umständen viele Interaktionen mit dem System erforderlich, wodurch die Spezifikation mit diesem Verfahren aufwendig ist.

4.2.1.3 Sicherheitsfachsprache von Mertke et al.

In [HMD01] und [MDH01] wird ein Verfahren vorgestellt, bei dem eine Sicherheitsfachsprache zur Spezifikation von Sicherheitsanforderungen für Steuerungsprogramme verwendet wird (vgl. [Bits03a]). Die Zielsetzung ist die Verifikation der Anforderungen durch Model-Checking. In dieser Sicherheitsfachsprache sind für temporallogische Grundkonstrukte Entsprechungen in natürlicher Sprache definiert. Zudem werden für aussagenlogische Anteile in der Formulierung von Anforderungen natürlich-sprachliche Konstrukte projektspezifisch definiert und verwendet. Mithilfe von zwei Klassifikationskriterien können Aussagen in dieser Sicherheitsfachsprache ausgewählt, zusammengesetzt und als Sicherheitsanforderungen ausgedrückt werden.

Die Sicherheitsfachsprache wurde speziell für Anforderungen an Speicher-Programmierbare Steuerungen (SPS) konzipiert, sodass damit „typische steuerungstechnische Problemstellungen“ [HDM01] spezifiziert werden können. Es werden feste Satzstrukturen vorgegeben. Der Benutzer wählt zunächst eine von achtzehn Satzstrukturen aus. Anschließend werden die Freiräume innerhalb der Sätze mit „applikationsspezifischen Wortphrasen“ [MDH01] ausgefüllt. Diese stehen für SPS-Variablen und entsprechende Werte (z. B. steht „Taster“ für eine bestimmte

Variable und „gedrückt“ für den Wert „wahr“ und „nicht-gedrückt“ für den Wert „falsch“). Aufgrund der Definition einer formalen Syntax und Semantik können die in der Sicherheitsfachsprache formulierten Anforderungen nach Temporallogik automatisch übersetzt werden. Durch die Nutzung der natürlichen Sprache für die Anforderungsspezifikation soll eine verständliche und auch für Nichtspezialisten lesbare Darstellungsform erreicht werden.

Der Ansatz umfasst zudem eine sehr ausgereifte Lösung zur Spezifikation boolescher Ausdrücke durch fachspezifische Wortphrasen. Allerdings ist, wie in Kapitel 3 gezeigt wurde, die Schwierigkeit bei der formalen Spezifikation nicht im Umgang mit Aussagenlogik, sondern im Umgang mit temporalen Operatoren zu sehen. Von einem Automatisierungstechnik-Ingenieur kann die Formulierung einfacher Bedingungen wie `x_1 = 1` ohne weiteres erwartet werden, wobei hier `x_1` eine SPS-Variable ist, die einen bestimmten Taster repräsentiert.

Die Auswahl der erforderlichen Satzstruktur erfolgt über die Bestimmung eines Zeit- und eines Modalparameters. Für die Referenzanforderung ist der Zeitparameter „Beobachtungsintervall ist fremdbegrenzt“, d. h., es beginnt mit der Erfüllung einer Bedingung `B1` und endet mit der Erfüllung einer weiteren Bedingung `B2`. Der Modalparameter ist „einfaches Verbot“ einer sogenannten Folgerung `F`, die von den Bedingungen `B1` und `B2` abhängt. Das Ergebnis ist folgende Satzstruktur*: „Wenn irgendwann* <`B1`> *war, dann darf nicht irgendwann* <`F`> *werden, bis* <`B2`> *ist.*“ Durch eine Nomendeklaration werden die Variablen festgelegt und die Bedeutung ihres Wahrheitswerts festgelegt: `B1` = „*eine Störung der Schrankenanlage eines Bahnübergangs*“, `F` = „*von einem Eisenbahnfahrzeug*“, `B2` = „*der Bahnübergang*“, „*aufgetreten*“ = `TRUE`, „*repariert*“ = `TRUE` und „*der Bahnübergang im automatischen Betrieb erneut befahren*“ = `TRUE`. Aufgrund dessen kann folgender Anforderungstext formuliert werden: „*Wenn irgendwann eine Störung der Schrankenanlage eines Bahnübergangs aufgetreten war, dann darf nicht irgendwann von einem Eisenbahnfahrzeug der Bahnübergang im automatischen Betrieb erneut befahren werden, bis der Bahnübergang repariert ist.*“ Damit passende Nomen deklariert werden können, muss also die ursprüngliche Formulierung der Sicherheitsanforderung umgestellt werden. Entsprechend der Satzstruktur und der Nomendeklaration erfolgt eine Übersetzung in Temporallogik.

4.2.1.4 Beurteilung der Ansatzart

Die drei vorgestellten einzelnen Ansätze lassen sich zu einer Ansatzart zusammenfassen, in der temporallogische Teilausdrücke durch Teilsätze oder einzelne Wörter in natürlicher Sprache repräsentiert werden. Für die Spezifikation von Anforderungen werden die natürlich-sprachlichen Teilausdrücke miteinander kombiniert. Dabei werden Aussagen über Variablen des operationellen Modells gemacht, dessen Korrektheit überprüft wird. Somit unterstützt diese Ansatzart die Forderung der Normen, formale Spezifikationen in natürlicher Sprache zu beschreiben.

Aussagen in natürlicher Sprache sind intuitiv und deshalb einfacher verständlich als Aussagen in Temporallogik. Die Notation wirkt vertraut. Allerdings ist zu berücksichtigen, dass natürliche Sprache missverstanden werden kann, wenn Worte nur eine bestimmte eingeschränkte Bedeutung besitzen sollen. Die korrekte Verwendung einer solchen Spezifikationssprache muss erlernt werden. Die Ansätze bieten dafür keine Hilfestellung durch Expertenwissen. Schwierigkeiten können hinsichtlich des künstlichen Satzbaus, der künstlichen Wortwahl sowie hinsichtlich großer Formulierungslängen auftreten. Künstliche, umständliche und unnatürliche Formulierungen bilden eine Gefahr für Fehlinterpretationen von Sicherheitsanforderungen.

In [MDH01] wird gezeigt, wie ein Ansatz dieser Ansatzart auf Sicherheitsanforderungen eingeschränkt werden kann. Beschränkt sich ein solches Spezifikationsverfahren nicht nur auf sicherheitsrelevante Anforderungen, hat das den Nachteil, dass das geeignete Spezifikationskonstrukt schwieriger identifizierbar ist, da es mehr Spezifikationsmöglichkeiten gibt. Dadurch sind außerdem Mehrdeutigkeiten der Anforderungssprache schwieriger zu kontrollieren. In keinem der vorgestellten Ansätze wird die Spezifikation aller sicherheitsrelevanten temporallogischen Ausdrucksmöglichkeiten garantiert. Bedeutende Spezifikationsprobleme der Automatisierungstechnik bleiben teilweise unbehandelt. In den Ansätzen von *Holt et al.* und *Mertke et al.* wird die Spezifikation von Anforderungen mit quantitativen Zeitangaben sowie von Anforderungen für zeitgesteuerte Systeme nicht direkt unterstützt. In den Ansätzen von *Flake et al.* und *Mertke et al.* gibt es keine direkte Unterstützung zur Differenzierung bei der zeitlichen Aufeinanderfolge von Bedingungen, Ereignissen oder Aktionen, ob der Nachfolger bereits zusammen mit dem Vorgänger gültig ist oder nicht. Diese Unterscheidung spielt z. B. bei Bussystemen eine Rolle vgl. [Bits01].

4.2.2 Verwendung von Spezifikationssprachen in textueller Notation

4.2.2.1 RT-OCL von Flake

Der Grundgedanke dieser Ansatzart ist, dass man eine bekannte Textnotation, die Entwicklern bekannt und vertraut ist, um temporallogische Aspekte erweitert, sodass sie für die formal verifizierbare Anforderungsspezifikation verwendet werden können. Eine solche Textnotation, die mit der UML eine weite Verbreitung gefunden hat, ist die Object Constraint Language (OCL). Sie ermöglicht die Spezifikation von Bedingungen mit deklarativem Charakter, d. h. in Form nicht-ausführbarer Beschreibungen. So können in OCL Invarianten für Attribute in UML-Klassendiagrammen spezifiziert werden. Die OCL erlaubt die Spezifikation von Vor- und Nachbedingungen für Operationen. Dies sind Bedingungen, die vor bzw. nach der Ausführung der betreffenden Operation erfüllt sein müssen. Dabei bietet die OCL Operatoren über Mengen, Sequenzen und *Bags*[3] [RFP00] [WaKl99].

[3] Multimengen – in einer solchen Menge können Elemente mehrfach vorkommen.

OCL kann zwar in verhaltensbeschreibenden Diagrammen eingesetzt werden, z. B. in UML-Zustandsdiagrammen für Bedingungen an Zustandsübergängen. Es ist jedoch nicht möglich, dynamische oder zeitbehaftete Verhaltenseigenschaften mit OCL auszudrücken [FlMu01]. OCL ist keine temporallogische Sprache. Deshalb wurde von *Flake* und *Müller* [FlMu02] ein Ansatz entwickelt, in dem die OCL zu RT-OCL (Real-Time OCL) erweitert wurde, sodass sich diese in eine temporallogische Sprache abbilden lässt[4]. So können zustandsbasierte, zeitbehaftete Modelleigenschaften formuliert werden [FlMu01]. Somit lassen sich zeitbehaftete Anforderungen an Verhaltensbeschreibungen spezifizieren. Die resultierende Syntax wurde in der Weise entwickelt, dass sie sich mit der offiziellen OCL-Notation deckt.

Die Intention der OCL-Erweiterung ist, mit OCL auf einer bereits weit verbreiteten und bekannten Sprache aufzubauen [FlMu01]. Deshalb wird von der Erweiterung eine für den praktischen Einsatz verwendbare Spezifikationssprache erwartet, mit der auch komplexe, zeitbehaftete Modelleigenschaften formuliert werden sollen, ohne dass man sich direkt mit Temporallogik auseinander setzen muss. Die Referenzanforderung „*nach dem Auftreten von* `stoerung = 1` *muss* `befahren = 0` *solange gelten, bis* `reparieren = 1` *eintritt*“ kann in RT-OCL wie folgt spezifiziert werden:

```
inv: let requiredSequence = Sequence{not befahrenConf,
reparierenConf} in stoerung implies self@post→forAll(s:OclPath |
s→includes(requiredSequence) or s→excludes(befahrenConf))
```

Mit `inv` (Invariante) wird ausgesagt, dass die Anforderung immer gelten muss. Dann wird eine Sequenz beschrieben, über die anschließend eine Aussage folgt. `stoerung = 1` impliziert, dass in allen möglichen folgenden Systemverhaltenspfaden `s` die Sequenz gilt, dass der Bahnübergang nicht befahren werden darf (`not reparierenConf`), bis dieser repariert wird (`reparierenConf`). Wird die Sequenz nicht erfüllt, darf der Bahnübergang nie mehr befahren werden. `befahrenConf` bzw. `reparierenConf` beschreiben jeweils die System-Konfiguration, die bei der Erfüllung von `befahren` bzw. `reparieren` gültig ist.

4.2.2.2 Beurteilung der Ansatzart

Durch diese Ansatzart werden neue temporallogische Notationen geschaffen. Die Syntax dieser Sprachen kann einfacher erlernbar sein als die von Temporallogik, da sie sich an dem Notationsschema bekannter Sprachen orientiert. Allerdings muss auch bei einer solchen neuen temporallogischen Notation die temporallogische Semantik beherrscht werden. Dabei ist der Entwickler vollständig auf sich selbst gestellt und erhält keinerlei Unterstützung durch Expertenwissen beim Beherrschen und Erlernen einer solchen Sprache. Heutzutage werden immer mehr grafische Notationen für die Modellierung verwendet. Es ist fraglich, ob geeignete textuelle Spezifikationsnotationen gefunden werden können, die eine große Verbreitung gefun-

[4] In [FlMu01] werden ähnliche Verfahren erwähnt, für die hier stellvertretend nur auf die Arbeiten von *Flake* et al. eingegangen wird.

den haben und deshalb vielen Ingenieuren vertraut sind. Zudem kann die Integration temporal-logischer Aspekte in das Schema einer bestehenden Sprache die Spezifikation zeitlich-logischer Ausdrücke leicht kompliziert machen. Bestehende Sprachen verfolgen eine bestimmte Intention und besitzen eine bestimmte Ausrichtung sowie ein bestimmtes Anwendungsfeld. Diese Ausrichtung muss nun für die Spezifikation zeitlich-logischer Ausdrücke erweitert werden.

Ein Punkt, bei dem man den bestehenden Ansatz noch verbessern könnte, wäre eine Spracherweiterung für die ausschließliche Spezifikation von Sicherheitsanforderungen. Dadurch wäre der Sprachumfang geringer und die Sprache deshalb zumindest ein wenig leichter zu beherrschen. Für Spezifikationsprobleme der Automatisierungstechnik könnten zudem noch spezielle Sprachkonstrukte für die Differenzierung bei der zeitlichen Aufeinanderfolge von Bedingungen, Ereignissen oder Aktionen bereitgestellt werden, ob der Nachfolger bereits zusammen mit dem Vorgänger gültig sein soll oder nicht (vgl. Abschnitt 4.2.1.4).

4.2.3 Verwendung von Spezifikationssprachen in grafischer Notation

4.2.3.1 Symbolic Timing Diagrams von Schlör et al.

Die Spezifikation von Eigenschaften in Temporallogik wird in [SJW98] als fehleranfällig, inakzeptabel für gewöhnliche Entwickler und lediglich geeignet für Spezialisten dargestellt. Als Ausweg wird die Verwendung von domänenspezifischen Sprachen mit geringerer Ausdrucksmächtigkeit gesehen, die jedoch dazu ausreicht, typische Anforderungen der Anwendungsdomäne zu spezifizieren.

In [ScDa93] und [SJW98] wird speziell für die formale Verifikation von Hardwareentwürfen mit Model-Checking ein Verfahren vorgeschlagen, bei dem zur Spezifikation von Anforderungen die grafische Notation *Symbolic Timing Diagrams* (STD) verwendet wird. STD erlauben das Ausdrücken qualitativer zeitlicher Zusammenhänge und basieren auf der Darstellungsweise von Zeitdiagrammen, wie sie in der Hardwareentwicklung gewöhnlicherweise zur Beschreibung zeitlicher Relationen verwendet werden. Somit wird eine Notation verwendet, mit der Hardwareentwickler vertraut sind [ScDa93]. Für die Spezifikation kommt ein Softwarewerkzeug zum Einsatz, das einen entsprechenden STD-Editor umfasst. Die Semantik von STD wird durch Transformationsvorschriften nach Temporallogik formal definiert. Abbildung 4.2 zeigt die Spezifikation für die Referenzanforderung „*nach dem Auftreten von* `stoerung = 1` *muss* `befahren = 0` *solange gelten, bis* `reparieren = 1` *eintritt*“.

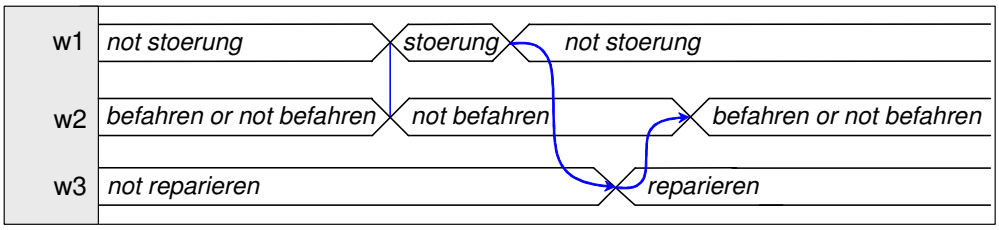


Abbildung 4.2: Spezifikation der Referenzanforderung mit STD

`w1`, `w2` und `w3` bezeichnen jeweils einen Grafen. Ein Graf beschreibt einen Zeitstrahl, der durch eine Sequenz mehrerer Bereiche dargestellt ist [SJW98]. Im Beispiel beschreibt der Graf `w3` z. B. eine Sequenz, in der zunächst im ersten Bereich die Variable *`reparieren`* nicht erfüllt ist. Im folgenden Bereich ist *`reparieren`* gültig. Die senkrechte Linie zwischen den Grafen `w1` und `w2` bezeichnet Gleichzeitigkeit. Das Eintreten einer Störung bringt ein gleichzeitiges Verbot des Befahrens des Bahnübergangs mit sich. Die Pfeile drücken aus, dass das Ereignis, bei dem der jeweilige Pfeil beginnt, vor dem Ereignis stattfinden muss, auf welches das Pfeilende zeigt.

4.2.3.2 Life Sequence Charts von Damm, Harel und Klose

In [DaHa99] und [DaWe03] wird ein Verfahren zur formalen Spezifikation von Anforderungs-Sequenzen in grafischer Notation vorgeschlagen. Hierfür werden Message Sequence Charts (MSC) [ITU93] zu Life Sequence Charts (LSC) erweitert. Insbesondere besteht die Erweiterung darin, dass die Modalität einer Sequenz beschrieben werden kann. Es kann also ausgedrückt werden, ob eine Sequenz ein Szenario beschreibt, das möglich sein muss (*existential*) oder notwendigerweise (*mandatory*) gelten muss. Diese Unterscheidung kann auch für einzelne Bedingungen sowie Nachrichten gemacht werden. Zudem können Bedingungen angegeben werden, die für den Ablauf einer Sequenz erfüllt sein müssen. Diese Bedingungen werden in Prädikatenlogik oder wiederum durch LSC, sog. Pre-Charts, ausgedrückt. LSC besitzen eine rigorose formale Semantik.

Die Intention von LSC ist die Spezifikation von Intra-Objekt-Verhalten [DaHa99]. In LSC können zeitliche Abläufe in der Interaktion zwischen Objekten dargestellt werden. Dabei steht die Kommunikation zwischen Objekten im Vordergrund. Sicherheitsanforderungen beziehen sich jedoch häufig nur auf ein Objekt (System oder Systemkomponente) und beschreiben nur einzelne Relationen. Ein wesentlicher Unterschied zu STD ist, dass auch Lebendigkeitsanforderungen spezifiziert werden können. In Lebendigkeitsanforderungen können Aussagen über Eigenschaften gemacht werden, die schließlich irgendwann gelten sollen. Abbildung 4.3 zeigt die Spezifikation der Referenzanforderung „*nach dem Auftreten von* `stoerung = 1` *muss* `befahren = 0` *solange gelten, bis* `reparieren = 1` *eintritt*“ in LSC Notation. Dabei wird davon ausgegangen, dass *`reparieren`* im Teilsystem *`Bahnuebergang`* verursacht wird und im Teilsystem *`Schrankenanlage`* als Ereignis eintritt.

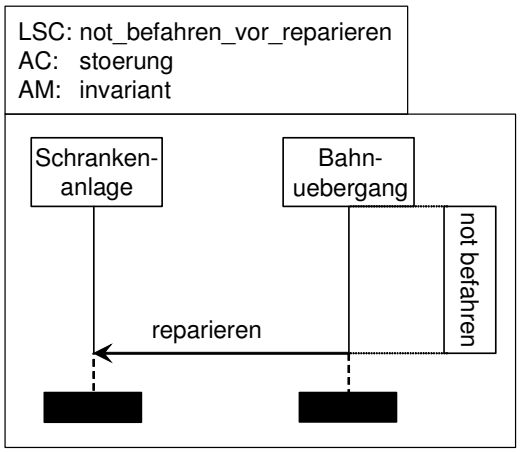


Abbildung 4.3: Spezifikation der Referenzanforderung mit LSC

`AC` bedeutet *activation condition* und ist Voraussetzung für die Geltung der beschriebenen Sequenz. Im Beispiel ist dies, dass eine Störung (`stoerung = 1`) vorliegt. `AM` steht für *activation mode*, wodurch festgelegt wird, ob die Sequenz *initial* (nur bei Systemstart), *invariant* (jederzeit) oder *iterativ* aktivierbar sein soll. Die durchgezogene Rahmenlinie zeigt an, dass es sich um eine notwendige Sequenz handelt.

4.2.3.3 MSC_{CTL} von Canver

In [BCM00] und [Canv99] wird ein weiteres Verfahren zur formalen Spezifikation von Anforderungs-Sequenzen in grafischer und textueller Notation vorgeschlagen. Dabei werden zur grafischen Spezifikation MSC (Message Sequence Charts) als Mittel zur Beschreibung von Szenarien aus Interaktionen zwischen Objekten verwendet. In [BCM00] wird herausgestellt, dass sich MSC nur begrenzt zur Spezifikation von Anforderungen an ein System eignen, da nicht festgelegt ist, unter welchen Rahmenbedingungen ein Sequenzdiagramm gelten soll. Aus diesem Grund wurde im LSC-Konzept (vgl. Abschnitt 4.2.3.2) die Syntax von MSC entsprechend verändert und erweitert. Im MSC_{CTL}-Konzept wird hingegen die MSC-Notation unverändert übernommen und in eine MSC_{CTL}-Formel eingebettet.

In MSC_{CTL}-Ausdrücken wird textuell beschrieben, unter welchen Rahmenbedingungen eine beschriebene Sequenz interpretiert werden muss. Mit den aus CTL bekannten Pfadquantoren `A` und `E` wird dabei festgelegt, ob eine beschriebene Sequenz immer gelten soll oder ob sie eine mögliche Sequenz darstellt, z. B.: `A MSC1`. Zudem können in MSC_{CTL}-Ausdrücken MSC negiert werden, wenn bestimmte Abfolgen nicht eintreten sollen. Sie können konjugiert werden, wenn mehrere Abfolgen gleichzeitig gelten sollen und sie können mit einer Disjunktion verknüpft werden, wenn alternative Abläufe möglich sein sollen. In MSC_{CTL}-Ausdrücken können zudem Bedingungen für den Sequenzablauf als CTL-Formeln beschrieben werden. Die formale Semantik der MSC_{CTL}-Formeln ergibt sich aus ihrer Umsetzung in die Temporallogik CTL. Abbildung 4.4 zeigt die Referenzanforderung „*nach dem Auftreten von* `stoerung = 1` *muss* `befahren = 0` *solange gelten, bis* `reparieren = 1` *eintritt*“ als Formulierung mit MSC_{CTL}.

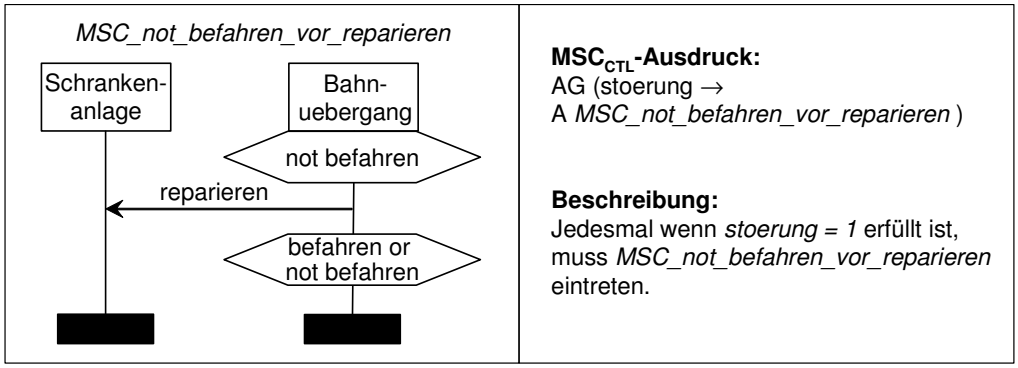


Abbildung 4.4: Spezifikation der Referenzanforderung mit MSC_{CTL}

Im DFG-Projekt *SafeRail* wurden Werkzeuge entwickelt [ABG02], die eine Auswertung von Sequenzdiagrammen aus dem Werkzeug Rhapsody und eine Transformation von MSC_{CTL}-Ausdrücken in CTL-Ausdrücke sowie in das μ-Kalkül ermöglichen.

MSC_{CTL} weist noch Mängel hinsichtlich der Ausdrucksmöglichkeiten auf. So kann in einer Sequenz z. B. nur spezifiziert werden, dass eine Aktion oder Nachricht im nächsten Systemzustand nach dem Eintritt einer Bedingung gelten soll. Es kann jedoch nicht beschrieben werden, dass sie bereits ab dem Zeitpunkt des Bedingungseintritts gelten soll. Die korrekte Einbettung der MSC in einen MSC_{CTL}-Ausdruck birgt eine wesentliche Schwierigkeit bei der Anwendung in sich. Zur Beschreibung der Rahmenbedingungen, unter denen die MSC gelten sollen, müssen temporallogische Konstrukte aus CTL direkt verwendet werden.

4.2.3.4 Beurteilung der Ansatzart

Die drei vorgestellten Ansätze haben die Gemeinsamkeit, dass jeweils eine grafische Notation verwendet wird, um die Bedeutung von zeitlich-logischen Ausdrücken in der Spezifikation zu veranschaulichen. Komplizierte Bedeutungen werden somit während des Spezifizierens visualisiert. Gleichwohl muss der korrekte Gebrauch der Notationen erlernt werden. Es ist notwendig, den selbstständigen Umgang mit der temporallogischen Semantik zu beherrschen, die sich, wenn auch in eingeschränkter Weise, hinter den grafischen Notationen verbirgt (vgl. [Bits01]). Die beschriebenen grafischen Notationen besitzen eine formale Definition, welche eindeutige Beschreibungen erlaubt. Diese Eindeutigkeit ist formal korrekt und ein Argument gegen die Verwendung natürlicher Sprache. Jedoch ist die Eindeutigkeit für viele Entwickler nicht nutzbar, weil sie die formale Definition, die auf Temporallogik basiert, nicht verstehen.

Das Spezifizieren einzelner einfacher Relationen, wie sie häufig in Sicherheitsanforderungen vorkommen, ist mit den vorgestellten Beschreibungsmitteln umständlich. In keinem der vorgestellten Ansätze wird die Spezifikation aller sicherheitsrelevanten temporallogischen Ausdrucksmöglichkeiten garantiert. Für bedeutende Spezifikationsprobleme der Automatisierungstechnik stehen zum Teil keine speziellen grafischen Konstrukte zur Verfügung. Mit STD lassen sich keine Erreichbarkeitsanforderungen spezifizieren. Mit den grafische Beschreibungsmitteln von

MSC_{CTL} lässt sich nicht differenzieren, ob bei der zeitlichen Folge einer Aktion oder Nachricht nach eine Bedingung der Nachfolger bereits zusammen mit dem Vorgänger gültig ist oder nicht.

Im Unterschied zu den bestehenden Ansätzen lässt sich auch ein Ansatz verwirklichen, der auf sicherheitsrelevante Anforderungen spezialisiert ist. Dies hätte den Vorteil, dass die Ausdrucksmächtigkeit der grafischen Notation reduziert und damit die Kontrollierbarkeit der Notationen verbessert werden könnte.

4.2.4 Werkzeugführung zur direkten Spezifikation in Temporallogik

4.2.4.1 Wizard (MuWiz) zur Spezifikation von µ-Kalkül-Formeln

In [Tche03] wurde ein Verfahren zur werkzeuggestützten direkten formalen Spezifikation von Anforderungen im µ-Kalkül beschrieben. Der Benutzer wird bei der Aufstellung eines temporallogischen Ausdrucks im µ-Kalkül schrittweise durch ein Werkzeug angeleitet. Dabei werden nicht alle Ausdruckmöglichkeiten des µ-Kalküls unterstützt, sondern nur eine praxisrelevante Teilmenge des µ-Kalküls, das r.u.c. (*restricted unnested closed*) Fragment genannt wird.

Das µ-Kalkül wird jedoch als schwer verstehbar bewertet und es werden Erfahrungen berichtet, dass selbst Experten für formale Methoden mehrere Stunden benötigten, um einen gewöhnlichen Ausdruck im µ-Kalkül zu verstehen [BrSt90, S. 9]. Deshalb wurde das Werkzeug MuWiz (Mu-Calculus Formula Constructor Wizard) entwickelt, das bei der Aufstellung von Ausdrücken im r.u.c. Fragment des µ-Kalküls anleitet. Voraussetzungen dabei sind, dass der Benutzer ein Grundverständnis für den Aufbau temporallogischer Ausdrücke besitzt und speziell hinreichende Kenntnisse des µ-Kalküls hat. Zum Beispiel wird dem Benutzer des Werkzeugs die µ-Kalkül spezifische Frage gestellt, ob die Formulierung eine „*fixpoint recursion*“ enthalten soll. Eine Aufforderung, in der ein Grundverständnis in Temporallogik vorausgesetzt wird, ist:

```
Select:   - „eventually on some paths (EF)“
          - „Progress (AF)“
          - „Safety (AG)“
          - …
```

4.2.4.2 Beurteilung der Ansatzart

Durch die werkzeuggestützte Führung ist der Benutzer beim Spezifizieren nicht auf sich selbst gestellt, sondern wird durch Expertenwissen unterstützt. Das Werkzeug ist somit Entscheidungshilfe beim direkten Spezifizieren in Temporallogik. Allerdings setzt die korrekte Verwendung eines solchen Werkzeugs Kenntnisse in Temporallogik und ein Grundverständnis über den Aufbau temporallogischer Ausdrücke voraus. Der bestehende Ansatz erlaubt die Spezifikation aller möglichen CTL-Ausdrücke und geht darüber noch hinaus. Für bedeutende Spezifikationsprobleme der Automatisierungstechnik (Anforderungen mit quantitativen Zeitangaben, Anforderungen für zeitgesteuerte Systeme und Differenzierung in zeitlichen Folgen, vgl.

Abschnitt 4.2.1.4), für die Temporallogik in bestimmter Weise angewandt werden muss, besteht jedoch keine spezielle Unterstützung.

Es ist auch ein Spezifikations-Werkzeug denkbar, das auf die Spezifikation sicherheitsrelevanter Anforderungen spezialisiert ist. Dies hätte den Vorteil, dass weniger Ausdrucksmöglichkeiten durch das Werkzeug kontrolliert werden müssten. Dadurch könnten die Interaktionen mit dem Benutzer und damit der Spezifikationsaufwand reduziert werden.

Bei der Interpretation temporallogischer Ausdrücke bietet der Ansatz keine direkte Unterstützung. Solch ein Spezifikations-Werkzeug kann für die Interpretation lediglich in der Weise genutzt werden, indem versucht wird, einen temporallogischen Ausdruck mithilfe des Werkzeugs nachzuspezifizieren. Dies ist jedoch als zeitaufwendig zu werten.

4.2.5 Verwendung von temporallogischen Formeln

4.2.5.1 Grundeinteilung temporallogischer Ausdrücke nach Manna und Pnueli

Manna und *Pnueli* klassifizierten temporallogische Ausdrücke in LTL [MaPn95] (vgl. [Bits01]). Damit schufen sie Temporallogik-Formeln, die Entwickler beim Spezifizieren in Temporallogik unterstützen können. Dabei werden sieben temporallogische Formeln bereitgestellt. Die Verwendung dieser Formeln setzt fundierte Kenntnisse in Temporallogik voraus. Für jede Formel werden Beispiele beschrieben und es sind umfangreiche textuelle Erläuterungen vorhanden. Die Einteilung erfolgt nach grundlegenden syntaktischen Unterschieden anstatt von einem informellen Verständnis der Bedeutung unterschiedlicher Ausdrücke auszugehen (vgl. [DAC98]). Für die Praxis ist es unnatürlich, Spezifikations-Probleme von der Syntax her anzugehen, sodass die Ermittlung der geeigneten Formel mit einer gewissen Schwierigkeit verbunden ist. Der Benutzer selektiert die geeignete Formel somit nicht problemspezifisch. Stattdessen muss er sich Gedanken über die geeignete Darstellung in LTL machen. Aufgrund der Art und Weise der Klassifizierung ist diese nur für LTL-Ausdrücke anwendbar.

Eine der Formeln wird z. B. *safety formula* genannt: $\Box$[5] p. Dabei wird davon ausgegangen, dass wenn ein temporallogischer Ausdruck eine bestimmte Struktur besitzt, es sich um einen sicherheitsrelevanten Ausdruck handle. Solch ein Ausdruck kann jedoch z. B. auch für Zuverlässigkeitsanforderungen verwendet werden. Somit handelt es sich nicht unbedingt um Sicherheitsrelevanz im Sinne der Automatisierungstechnik. Es ließe sich lediglich der logische Schluss ziehen, dass für eine Anforderung, die Sicherheitsrelevanz besitzt, eine bestimmte formale Formulierung erforderlich ist, nicht jedoch umgekehrt [Bits01].

[5] Das Symbol „$\Box$" ist dabei gleichbedeutend mit dem in Abschnitt 2.4.1.3 eingeführten Symbol G.

4.2.5.2 Pattern-System von Dwyer, Avrunin und Corbett

Dwyer, *Avrunin* and *Corbett* [DAC98] [DAC99] haben Formeln zur Spezifikation von Eigenschaften mit Temporallogik für die Verifikation durch Model-Checking bereitgestellt (vgl. [Bits01]). Die Gesamtheit dieser Formeln nennen sie „Pattern-System“. Ähnlich zu dem Ansatz von *Manna* und *Pnueli* sollen Entwickler bei der Verwendung von Temporallogik unterstützt werden. Dabei ist ihre Klassifizierung mit 55 Formeln und zwei Klassifikationsebenen wesentlich feiner als die von *Pnueli* und *Manna*. Die Klassifikation ist so pragmatisch gewählt, dass sie für unterschiedliche temporallogische Sprachen angewendet werden kann. Allerdings lassen sich mit den Formeln von *Manna* und *Pnueli* Ausdrücke bilden, die nicht in dem Pattern-System vorkommen [DAC99]. Durch eine feinere Klassifikation ließe sich das Auswählen der geeigneten Formel noch wesentlich verbessern.

Der Ansatz ist nicht auf die Spezifikation von Sicherheitsanforderungen beschränkt. Somit gibt es Formeln, die für Sicherheitsanforderungen irrelevant sind. Dadurch müssen mehr Spezifikationsmöglichkeiten berücksichtigt werden. Das Auswählen der geeigneten Formeln ist aufwendiger, als wenn der Ansatz auf die Spezifikation von Sicherheitsanforderungen beschränkt würde. Die Auswahl der geeigneten Formel erfolgt nicht werkzeuggestützt, sondern manuell aus einer Auflistung. Typische Spezifikationsprobleme der Automatisierungstechnik, wie z. B. die Spezifikation mit expliziter Zeit, werden nicht durch spezielle Formeln unterstützt. Ein anderes Beispiel ist die fehlende Differenzierung bei der zeitlichen Aufeinanderfolge von Bedingungen, Ereignissen oder Aktionen, ob der Nachfolger bereits zusammen mit dem Vorgänger gültig ist oder nicht. Diese Unterscheidung spielt z. B. bei Bussystemen eine Rolle [Bits01].

Für die die Spezifikation der Referenzanforderung „*nach dem Auftreten von* `stoerung = 1` *muss* `befahren = 0` *solange gelten, bis* `reparieren = 1` *eintritt*“, muss beispielsweise über die Klassen “Existence” und “*p* becomes true: Between *r* and *q*” ausgewählt werden. “Existence” bezieht sich dabei darauf, dass die Variable `reparieren` schließlich irgendwann wahr werden muss zwischen `stoerung = 1` und `befahren = 1`. Um die geeignete Klassifikation zu finden, steht somit statt wie in der ursprünglichen Formulierung der Sicherheitsanforderung die Gültigkeit der Variable `befahren` die Variable `reparieren` im Zentrum der Betrachtung.

4.2.5.3 Beurteilung der Ansatzart

Die in diesem Unterkapitel vorgestellten Ansätze unterstützen Entwickler beim Spezifizieren von Anforderungen durch die Bereitstellung von Formeln. Der Entwickler muss somit nicht mehr vollständig selbstständig spezifizieren, sondern wird durch das in den Formeln inhärente Expertenwissen unterstützt. Voraussetzung für die Verwendung eines solchen Ansatzes ist ein Grundverständnis für Temporallogik. Der korrekte Gebrauch der Formeln muss erlernt werden. Die korrekte Anwendung der Formeln hängt wesentlich von der richtigen Interpretation der

Formeln ab. Bei den angeführten Ansätzen werden dabei entweder zu wenige Informationen gegeben oder die Informationen sind wenig übersichtlich und wenig strukturiert und daher schwer erfassbar. Deshalb ist die Selektion der Formeln zeitaufwendig. In keinem der vorgestellten Ansätze wird die Spezifikation aller sicherheitsrelevanten temporallogischen Ausdrucksmöglichkeiten garantiert. Auch wenn im Ansatz von *Dwyer et al.* wesentlich feiner differenziert wird als bei *Manna* und *Pnueli*, werden bedeutende Spezifikationsprobleme der Automatisierungstechnik nicht durch spezielle Formeln unterstützt (Anforderungen mit quantitativen Zeitangaben, Anforderungen für zeitgesteuerte Systeme und Differenzierung in zeitlichen Folgen, vgl. Abschnitt 4.2.1.4).

Entgegen den bestehenden Ansätzen ließe sich auch ein Ansatz verwirklichen, der auf sicherheitsrelevante Anforderungen spezialisiert ist. Dadurch würde die Anzahl an Formeln verringert werden. Damit kann der Spezifikations- und Interpretationsaufwand reduziert werden.

4.3 Zusammenfassende Bewertung der vorgestellten Ansätze und Schlussfolgerungen für ein geeignetes Verfahren

4.3.1 Kriterien und Bewertung

Die Betrachtung der unterschiedlichen Ansätze hat zum einen verschiedene Ansatzpunkte gezeigt, um den Schwierigkeiten bei der Spezifikation von Sicherheitsanforderungen in Temporallogik zu begegnen. Zum anderen wurden Probleme deutlich, die bei der Entwicklung eines geeigneten Verfahrens zu berücksichtigen sind. Das aussichtsreichste Verfahren wird sich nicht durchsetzen, wenn die Anwendbarkeit im Vergleich zur herkömmlichen direkten Spezifikation in Temporallogik nicht deutlich verbessert wird. Zudem müssen Aspekte der Einführbarkeit in die industrielle Praxis Berücksichtigung finden. Im Nachfolgenden werden die verschiedenen Bewertungskriterien erläutert und auf die verschiedenen Ansatzarten angewandt.

Keine direkte Spezifikation in Temporallogik

Ein Bewertungskriterium ist, ob eine Ansatzart die direkte Spezifikation in Temporallogik erfordert. Ist dies der Fall, müssen zumindest Grundkenntnisse über die Spezifikation in Temporallogik vorhanden sein. Dies kann ein Hindernis für die Einführung des Verfahrens in die industrielle Praxis sein. Zum Gebrauch des Verfahrens müsste zunächst die Verwendung von Temporallogik erlernt werden.

Lediglich bei zwei Ansatzarten wird nicht direkt in Temporallogik spezifiziert. Im einen Fall wird stattdessen eine Terminologie der natürlichen Sprache verwendet. Im anderen Fall wird in grafischer Notation formuliert. Bei der Spezifikation in einer adaptierten textuellen Notation

wird zwar keine der bekannten temporallogischen Notationen verwendet. Allerdings stellt eine solche Notation letztlich eine neue temporallogische Sprache dar, bei der ggf. die Syntax vertrauter ist.

Beherrschen der temporallogischen Semantik nicht erforderlich

Wie bereits in Abschnitt 4.1 gezeigt wurde, ist die Schwierigkeit von Temporallogik in der komplexen und ausdrucksmächtigen Semantik von Temporallogik begründet. Deshalb ist zu bewerten, ob in den verschiedenen Ansatzarten dieses Grundproblem behandelt wird. Selbst wenn ein Ansatz nicht die direkte Spezifikation in der herkömmlichen Notation von Temporallogik erfordert, kann dennoch nach wie vor die Beherrschung der möglichen unterschiedlichen Ausdrucksmöglichkeiten eine entscheidende Schwierigkeit darstellen. Dies ist insbesondere dann der Fall, wenn durch den Ansatz lediglich die Syntax von Temporallogik z. B. durch eine grafische Notation ersetzt wird. Dann wird zwar eine andere Notation verwendet, hinter der sich allerdings die schwierige temporallogische Semantik verbirgt, deren Ausdrucksmöglichkeiten durch den Spezifizierer nach wie vor zu beherrschen sind.

Lediglich bei der Verwendung vorgegebener temporallogischer Formeln und bei der geführten direkten Spezifikation in Temporallogik muss die temporallogische Semantik nicht vollständig beherrscht werden. In beiden Fällen ist die Beherrschung jedoch zumindest eingeschränkt erforderlich. Zudem sind mindestens Grundkenntnisse in Temporallogik erforderlich: im einen Fall, um die Formeln zu verstehen und korrekt zu gebrauchen und im anderen Fall, um die werkzeuggestützte Führung überhaupt verwenden zu können.

Erlernbarkeit der korrekten Anwendung

Für die Einführung eines neuen Verfahrens in die industrielle Entwicklung ist von entscheidender Bedeutung, dass der Entwicklungsaufwand und damit die Entwicklungskosten gering gehalten werden. Dies kann zum Problem werden, wenn die Schulungskosten zur Einführung der Methode zu hoch sind oder die Schulungskosten kontinuierlich anfallen, da das Beherrschen des Verfahrens kontinuierliches Training oder Weiterbildung erfordert. Hohe Entwicklungskosten können zudem entstehen, wenn ein Verfahren nur von Spezial-Experten korrekt angewandt werden kann, da das Erlernen des Verfahrens durch Entwickler der Anwendungsdomäne zu hohe Schulungskosten erfordern würde. Aus diesen Gründen ist ein weiteres Kriterium zur Bewertung der Ansatzarten die *Erlernbarkeit der korrekten Anwendung*.

In allen Fällen ist ein Erlernen des korrekten Gebrauchs des jeweiligen Verfahrens erforderlich. Der korrekte Gebrauch einer adaptierten textuellen Notation, einer Werkzeugführung zur direkten Spezifikation in Temporallogik und von temporallogischen Formeln muss jeweils erlernt werden. Zudem müssen in diesen Fällen temporallogische Grundkenntnisse erlernt werden. Auch Textbausteine sind aufgrund ungewöhnlicher und künstlicher Formulierungen nicht immer intuitiv anwendbar. Die Spezifikation in einer grafischen Notation muss ebenso

erlernt werden, wobei hier das Erlernen einfacher erscheint, da durch die grafische Darstellung komplizierte Bedeutungen visualisiert werden.

Unterstützung durch Expertenwissen

Wenn Entwickler beim Gebrauch der formalen Spezifikationssprache vollständig auf sich selbst gestellt sind, können durch die mangelnde Beherrschung der Sprache leicht Spezifikationsfehler entstehen. Dem kann entgegengewirkt werden, indem das Verfahren Expertenwissen hinsichtlich des korrekten Gebrauchs der Spezifikationssprache nutzt, sodass der Entwickler bei der Anwendung des Verfahrens angeleitet und geführt wird sowie Informationen in den einzelnen Verfahrensschritten erhält. Dies wird unter dem Kriterium *Unterstützung durch Expertenwissen* zusammengefasst.

Bei den Ansatzarten mit einer grafischen oder textuellen Spezifikationssprache findet keine Unterstützung durch Expertenwissen statt. Im Fall der Ansatzarten, in denen Textbausteine oder temporallogische Formeln verwendet werden, wird jeweils Expertenwissen durch vorformulierte Spezifikationen bereitgestellt. Die Ansatzart mit einer geführten direkten Spezifikation in Temporallogik wird durch Expertenwissen in Form der schrittweisen Anleitung bei der Spezifikation unterstützt. Allerdings werden jeweils dennoch Vorkenntnisse vorausgesetzt, die erlernt werden müssen. Deshalb kann keine der untersuchten Ansatzarten positiv bewertet werden.

Unterstützung der Spezifikation in natürlicher Sprache

Wie bereits im Abschnitt 2.2.5 ausgeführt wurde, wird in den Normen die Beschreibung formaler Spezifikationen in natürlicher Sprache gefordert. Durch das Kriterium *Unterstützung der Spezifikation in natürlicher Sprache* wird bewertet, ob die verschiedenen Ansatzarten diese Spezifikation in natürlicher Sprache unterstützen. Wird dies nicht berücksichtigt, so können leicht Fehler bei der Beschreibung geschehen. Selbst wenn eine Ansatzart zur Vereinfachung der korrekten Spezifikation und Interpretation von Sicherheitsanforderungen führen würde, könnten sich dann leicht Fehler durch unpräzise und mehrdeutige Dokumentation in natürlicher Sprache ergeben. Lediglich bei der Ansatzart *Textbausteine in einer Terminologie der natürlichen Sprache* wird die Formulierung in natürlicher Sprache unterstützt.

Aufwand

Neben den bereits aufgeführten Faktoren, durch die Entwicklungskosten entstehen, ergeben sich weitere Kosten durch den Aufwand, den ein Verfahren bei der Entwicklung eines Automatisierungssystems mit sich bringt. Das aussichtsreichste Verfahren wird sich nicht durchsetzen, wenn der Aufwand die Entwicklungszeiten so hoch werden lässt, dass die Entwicklung unwirtschaftlich ist und den Nutzen nicht rechtfertigt. Durch das Kriterium *Aufwand* wird deshalb die Entwicklungsleistung bewertet, die zur Anwendung der Ansatzarten erforderlich ist.

Bei der Anwendung von Textbausteinen oder Formeln ist die Ermittlung der geeigneten Formulierungsbestandteile aufwendig. Durch Interpretationsschwierigkeiten bei Textbausteinen bzw. Formeln wird dabei der Aufwand nochmals erhöht. Da sich einzelne Schritte bei der geführten direkten Spezifikation in Temporallogik nicht als trivial gestalten, sondern Kenntnisse in Temporallogik erforderlich sind, ist auch hier der Aufwand zur Abarbeitung aller Spezifikationsschritte hoch. Durch die Schwierigkeiten, die sich im Fall von textuellen Spezfikationssprachen ergeben (vgl. alle anderen Bewertungskriterien), kann sich die Spezifikation einer Sicherheitsanforderung auch mit dieser Ansatzart sehr aufwendig gestalten. Lediglich bei der Verwendung grafischer Notationen kann der erforderliche Aufwand unter der Voraussetzung besser bewertet werden, dass die grafische Spezifikationssprache beherrscht wird. Es ist kein Abarbeiten bestimmter Spezifikationsschritte in einer starren Reihenfolge erforderlich. Zudem sind die Schwierigkeiten bei der Spezifikation geringer zu bewerten als bei der Verwendung textueller Notationen.

Alle Ansatzarten könnten durch Werkzeuge unterstützt werden. Auch wenn in einzelnen Ansätzen nicht alle Spezifikationsmöglichkeiten zur Formulierung von Sicherheitsanforderungen berücksichtigt werden, so besitzt jede Ansatzart das Potenzial hierfür (vgl. Abschnitt 5.1). Deshalb werden diese beiden Kriterien (Werkzeugunterstützung und Spezialisierung auf Sicherheitsanforderungen) nicht für den zusammenfassenden Vergleich herangezogen.

4.3.2 Zusammenfassender Vergleich der Ansatzarten

In Tabelle 4.1 werden abschließend alle betrachteten Spezifikationsansätze gegenübergestellt und hinsichtlich der aufgestellten Kriterien bewertet. Dabei wird für jede Ansatzart bewertet, ob das zu Grunde gelegte Kriterium gut (+), neutral (o) oder mangelhaft (–) erfüllt wird.

Aus der Bewertung in Tabelle 4.1 wird ersichtlich, dass hinsichtlich der Bewertungskriterien keiner der Ansatzarten eine adäquate Lösung bietet. Folglich ist es erforderlich, ein neues Konzept zu entwickeln. Zu bestimmten Kriterien weisen verschiedene Ansatzarten jedoch Vorteile gegenüber anderen Ansätzen auf. Bei jedem Kriterium gibt es mindestens eine Ansatzart die zumindest als neutral bewertet werden kann. Eine Vereinigung verschiedener Konzepte unter Beibehaltung der jeweiligen Vorteile würde somit einen wesentlich Fortschritt für die anwendbare Spezifikation und Interpretation von Sicherheitsanforderungen bringen.

Tabelle 4.1: Ansatzarten zur anwendbaren formalen Spezifikation im Vergleich

Ansatzart / *Bewertungskriterien*	**Textbausteine in einer Terminologie der natürlichen Sprache**	**Spezifikationssprachen in textueller Notation**	**Spezifikationssprachen in grafischer Notation**	**Führung zur direkten Spezifikation in Temporallogik**	**Temporallogische Formeln**
Keine direkte Spezifikation in Temporallogik	+	o	+	–	–
Beherrschen der temporallogischen Semantik nicht erforderlich	o	–	–	o	o
Erlernbarkeit der korrekten Anwendung	–	–	o	–	–
Unterstützung durch Expertenwissen	o	–	–	o	o
Unterstützung der Spezifikation in natürlicher Sprache	+	–	–	–	–
Aufwand	–	–	o	–	–

4.3.3 Schlussfolgerungen für ein Verfahren zur anwendbaren Spezifikation von Sicherheitsanforderungen

Aus den erkannten Defiziten der analysierten Ansatzarten können die in Kapitel 3 aufgestellten Konzept-Anforderungen, welche die Anwendbarkeit betreffen – /KA4/ *Handhabbarkeit* sowie /KA5/ *Verständlichkeit und Nachvollziehbarkeit* – konkretisiert werden:

- Die Verwendung des Verfahrens soll anwendbar sein, ohne dass Kenntnisse über die Spezifikation mit Temporallogik vorausgesetzt werden müssen.
- Für die Anwendbarkeit des Verfahrens soll das selbstständige Beherrschen der temporallogischen Semantik nicht erforderlich sein.
- Das Verfahren soll die korrekte Spezifikation und Interpretation von Sicherheitsanforderungen durch Expertenwissen unterstützen.
- Das Verfahren soll auch die Spezifikation in natürlicher Sprache mit einschließen.

Weitere zu erkannten Defiziten adäquate Anforderungen hinsichtlich Erlernbarkeit und Aufwand wurden bereits in Kapitel 3 definiert.

Die betrachteten Ansatzarten weisen die folgenden besonderen Vorteile auf. Falls möglich, ist eine Integration der verschiedenen Aspekte in ein neues Konzept nützlich:

- ***Textbausteine in einer Terminologie der natürlichen Sprache:*** Bei dieser Ansatzart wird eine vertraute Notation verwendet. Dadurch wird zumindest teilweise eine intuitive Verständlichkeit und Interpretierbarkeit erreicht.
- ***Spezifikationssprachen in grafischer Notation:*** Durch die Verwendung einer grafischen Notation können komplexe Bedeutungen visualisiert und veranschaulicht werden. Dies ist beim Spezifizieren hilfreich und erleichtert die Interpretation.
- ***Werkzeuggestützte Führung zur Spezifikation:*** Bei einer solchen Führung wird der Entwickler schrittweise angeleitet, sodass er nicht vollständig selbstständig spezifizieren muss.
- ***Temporallogische Formeln:*** Durch die Bereitstellung von Formeln wird dem Entwickler Arbeit zum kreativen selbstständigen Formulieren abgenommen.

In diesem Kapitel wurden bestehende Lösungsansätze, die eine anwendbare Spezifikation in Temporallogik verfolgen, beschrieben, verglichen und bewertet. Defizite der Ansätze wurden hinsichtlich der Zielsetzung dieser Arbeit ermittelt. Daraus wurden Schlussfolgerungen für die Entwicklung eines Verfahrens zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme gezogen und die Konzept-Anforderungen verfeinert. Auf der Grundlage dieser Betrachtungen wird im folgenden Kapitel ein neuer Ansatz zur anwendbaren formalen Spezifikation von Sicherheitsanforderungen für Automatisierungssysteme entwickelt.

[…] Muster bieten einfache und prägnante Lösungen […]. Sie erlauben die Wiederverwendung bewährter Lösungsstrategien […].

E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns

5 Anwendbare formale Spezifikation von Sicherheitsanforderungen durch Wiederverwendung

Ausgehend von den aufgestellten Konzept-Anforderungen wird in diesem Kapitel zunächst das adäquate Beschreibungsmittel zur formalen Spezifikation von Sicherheitsanforderungen für Automatisierungssysteme ermittelt. Aufgrund der im vorigen Kapitel gewonnen Erkenntnisse wird anschließend ein Ansatz zur anwendbaren formalen Spezifikation von Sicherheitsanforderungen entwickelt. Der Ansatz basiert auf der Vermittlung von Expertenwissen durch Wiederverwendung. Für die Herleitung des geeigneten Ansatzes werden bekannte Techniken zur Wiederverwendung aus der Softwareentwicklung analysiert. Bewährte Konzepte für die Verwendung von Mustern und Komponenten werden für die Spezifikation von Sicherheitsanforderungen adaptiert. Zudem werden Voraussetzungen erörtert, die für den Einsatz von Wiederverwendungstechniken für die Spezifikation von Sicherheitsanforderungen erforderlich sind. Entsprechende Ansätze für Maßnahmen zur Anwendbarkeit eines solchen Wiederverwendungs-Konzepts werden abgeleitet und erläutert.

5.1 Das adäquate Beschreibungsmittel zur formal verifizierbaren Spezifikation von Sicherheitsanforderungen für Automatisierungssysteme

Bei der Beschreibung von Eigenschaften von Automatisierungssystemen spielen zeitliche Relationen eine besondere Rolle (vgl. Abschnitt 2.1). Ein Automatisierungssystem unterliegt den zeitlichen Abläufen im technischen Prozess. Deshalb ist das zeitliche Verhalten bei Automatisierungssystemen von besonderer Bedeutung (vgl. [LaGö99b, S.1]). Dies muss folglich durch das Konzept für ein Verfahren entsprechend berücksichtigt werden. In Sicherheitsanforderungen müssen Aussagen über das Verhalten des Automatisierungssystems im Betrieb getroffen werden. Deshalb müssen in Sicherheitsanforderungen zeitbehaftete Aussagen formuliert werden können. Für die formale Spezifikation solcher Anforderungen ist also eine Logiksprache erforderlich, die zeitliche Aussagen ermöglicht. In Abschnitt 2.4 wurden Prädikatenlogik und Temporallogik als dafür geeignete formale Spezifikationssprachen herausgestellt. Der Vorteil von Temporallogik dabei ist, dass Aussagen über zeitliche Relationen von Variablen des zu verifizierenden Modells der Systemfunktionen gemacht werden können,

ohne Zeit explizit ausdrücken zu müssen. Dies entspricht der natürlichen Art und Weise der Formulierung, wie an folgendem Beispiel verdeutlicht wird.

In Abschnitt 2.4 wurde bereits die unterschiedliche Formulierung der Sicherheitsanforderung an ein pneumatisches Bremssystem [BiGu00] in Prädikaten- und Temporallogik gezeigt. In dem Ausdruck „*Einlassventil und Auslassventil dürfen niemals gleichzeitig geöffnet sein*“ wird keine explizite zeitliche Aussage gemacht. Es kommt keine Zeitvariable `t` oder eine bestimmte Uhrzeit vor. Dennoch enthält die Sicherheitsanforderung zeitliche Relationen. Durch das Wort „niemals“ wird ausgedrückt, dass zu keinem Zeitpunkt etwas Bestimmtes sein darf. Durch das Wort „gleichzeitig“ wird eine Relation über einen Zeitpunkt formuliert. Entgegen dieser natürlichen Formulierung muss in Prädikatenlogik der Ausdruck mithilfe einer expliziten Zeitvariablen `t` spezifiziert werden, da Prädikatenlogik kein Zeitkonzept besitzt [EN50128]. Anhand des folgenden Beispiels wird gezeigt, dass Ausdrücke in Prädikatenlogik mit zeitlichen Aussagen wesentlich komplizierter sind als in Temporallogik.

Ausdruck in Prädikatenlogik:

$$\forall t1 \geq 0: [p(t1) \rightarrow \exists t2: (t1 \leq t2 \wedge r(t2) \wedge \exists t3: [t2 \leq t3 \wedge q(t3)])]$$

Derselbe Ausdruck wird in Temporallogik wesentlich einfacher, kompakter und eleganter formuliert:

$$G\ (p \rightarrow F[r \wedge F\ q])$$

Der Grundgedanke von Temporallogik ist, dass Aussagen in einem zeitlichen Kontext formuliert werden [HuRy00, S. 150]. Es wird ausgedrückt, in welchen Zuständen eines Zustandsraums Aussagen erfüllt bzw. nicht erfüllt sein müssen oder dürfen. Dadurch können qualitative zeitliche Relationen ausgedrückt werden, ohne dass explizite Zeitaussagen in Abhängigkeit zu einer Systemuhr gemacht werden müssen. Wenn jedoch eine Zeitaussage mit expliziten zeitlichen Angaben gewünscht ist, kann dies in Temporallogik ebenso ausgedrückt werden.

Die Semantik von Temporallogik, die ohne explizite zeitliche Variablen auskommt, liegt auch dem Model-Checking zu Grunde. Deshalb verwenden sämtliche Model-Checker Temporallogik als Spezifikationssprache für die Formulierung der zu verifizierenden Anforderungen. Wie bereits in Abschnitt 2.2.5 beschrieben, wird diese Bedeutung von Temporallogik in der Weise auch in den Normen beurteilt. Es wird als Basis für die „formale Demonstration“ bewertet, dass ein System die aufgestellten Anforderungen erfüllt. Nach [HuRy00, S. 150] ist Temporallogik die Grundlage, um mit dem in Abschnitt 3.2 als ingenieurgerecht herausgestellten formalen Verifikationsverfahren Model-Checking zeitbehaftete Aussagen überprüfen zu können.

Wie bereits in Abschnitt 2.4.1.3 erläutert wurde, kommt Temporallogik zudem die Bedeutung zu, dass durch eine Menge temporallogischer Ausdrücke die Eigenschaften reaktiver Systeme formal beschrieben werden können, vgl. [EsSc03], [Moik02], [AlHe92], [Pnue77]. Da Auto-

matisierungssysteme reaktive Systeme sind (vgl. 2.1), lassen sich somit die Anforderungen an Automatisierungssysteme in Temporallogik ausdrücken. Sicherheitsanforderungen an Automatisierungssysteme sind die sicherheitsrelevante Teilmenge dieser Eigenschaften. Mit dem Verfahren zur Spezifikation von Sicherheitsanforderungen muss also die Spezifikation der sicherheitsrelevanten Ausdrucksmöglichkeiten in Temporallogik möglich sein. Temporallogik ermöglicht die eindeutige Spezifikation von Sicherheitsanforderungen, die zudem formal verifizierbar sind, und es erlaubt das Ausdrücken aller funktionalen Sicherheitsanforderungen, die für Automatisierungssysteme formuliert werden können. Die Konzept-Anforderungen /KA1/ (präzise Spezifikation), /KA2/ (formal verifizierbare Spezifikation) und /KA3/ (vollständige Spezifizierbarkeit) werden durch die Verwendung von Temporallogik als Spezifikationssprache erfüllt. Mängel weist Temporallogik jedoch hinsichtlich der Konzept-Anforderungen /KA4/ (Handhabbarkeit) und /KA5/ (Verständlichkeit und Nachvollziehbarkeit) auf (vgl. Abschnitt 4.1). Im Folgenden wird analysiert, wie diesen Schwierigkeiten bei der Spezifikation von Sicherheitsanforderungen für Automatisierungssysteme durch einen geeigneten Verfahrens-Ansatz begegnet werden kann.

5.2 Vermittlung von Expertenwissen für die Spezifikation

Nachdem in Kapitel 4 die Schwierigkeiten der verschiedenen Ansätze für die anwendbare formale Spezifikation von Sicherheitsanforderungen für Automatisierungssysteme herausgestellt wurden, drängt sich die Frage nach einem geeigneten alternativen Ansatz für ein Verfahren auf. Bei der Verwendung neuer textueller oder grafischer Notationen wird im Wesentlichen nur die Syntax geändert. Hinter dieser Syntax verbirgt sich nach wie vor die schwer zu beherrschende Semantik von Temporallogik. Für die Anwendbarkeit eines geeigneten Verfahrens zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme ist es deshalb unerlässlich, dass kein selbstständiges Beherrschen der zeitlich-logischen Semantik erforderlich ist. Ebenso dürfen für die Anwendbarkeit des Verfahrens keinerlei Kenntnisse über die Spezifikation mit Temporallogik vorausgesetzt werden. Deshalb muss das geeignete Verfahren Expertenwissen für die Spezifikation und Interpretation von Sicherheitsanforderungen in Temporallogik vermitteln (vgl. Abschnitt 4.3.3).

Ansatzweise wird dieser Anforderung bei der Verwendung von Spezifikations-Formeln (vgl. Abschnitt 4.2.5) sowie durch die Führung zur direkten Spezifikation in Temporallogik (vgl. Abschnitt 4.2.4) nachgekommen. In beiden Fällen muss jedoch die formale Sprache erlernt und zumindest teilweise beherrscht werden. Nur so können die Spezifikationsformeln korrekt verstanden und gebraucht werden. Nur wenn man Temporallogik zumindest teilweise beherrscht, kann man sich von einem Werkzeug bei der Spezifikation in Temporallogik korrekt führen lassen.

Für die Vermittlung von Expertenwissen sind aus der Softwareentwicklung die Verwendung von Unterstützungssystemen und Techniken zur Wiederverwendung bekannt. Durch die Ver-

mittlung von Expertenwissen bei der Softwareentwicklung wird dabei die Vereinfachung der Entwicklung, die Vermeidung von Entwicklungsfehlern und die Steigerung der Effizienz bei der Entwicklung verfolgt. In den folgenden beiden Unterkapiteln werden diese Konzepte aus der Softwaretechnik im Hinblick für einen geeigneten Verfahrensansatz untersucht.

5.2.1 Analyse von Unterstützungssystemen

Zur werkzeuggestützten Anwendung von Verfahren kommen in der Softwaretechnik Unterstützungssysteme zum Einsatz. Man unterscheidet verschiedene Kategorien von Unterstützungssystemen [Balz96, S. 612 ff.]: Hilfesysteme, Tutorsysteme, Beratungssysteme und Assistenzsysteme. Hilfesysteme unterstützen bei der Benutzung von Softwaresystemen durch explizite Erklärungen und Auskünfte. Tutorsysteme ermöglichen ein interaktives Einarbeiten in Verfahren bzw. Softwaresysteme. Beratungssysteme beraten bei der Lösung von Problemen. Assistenzsysteme erledigen einfache Teilaufgaben (Hilfsarbeiten).

Die Lösung der Schwierigkeiten bei der Spezifikation von Sicherheitsanforderungen durch die Einarbeitung in die schwer zu beherrschende Semantik temporaler Logik ist aufgrund der Komplexität zu aufwendig. Tutorensysteme sind deshalb nicht als Unterstützungssysteme geeignet. Ebenso sind Assistenzsysteme ungeeignet, da die Effizienzprobleme bei der Spezifikation im Wesentlichen nicht durch aufwendige automatisierbare Hilfsarbeiten begründet sind. Erklärungen und Auskünfte durch Hilfesysteme sind auf jeden Fall hilfreich, wenn es um die Benutzung eines Systems geht, für das keine große Einarbeitung erforderlich sein soll. Beratungssysteme bieten darüber hinaus eine weitergehende Unterstützung, indem sie Lösungen und Handlungsschritte zur Erreichung eines Ziels vorschlagen.

Unterstützungssystemen liegt jedoch immer ein Softwaresystem und damit ein Verfahren zu Grunde, dessen Benutzung durch das Unterstützungssystem vereinfacht wird. Sie sind also als eigenständiges Verfahren für die Spezifikation von Sicherheitsanforderungen ungeeignet. Durch ein Hilfe- und Beratungssystem kann jedoch ein geeignetes Verfahren unterstützt werden. Im folgenden Abschnitt werden die wesentlichen Verfahren zur Wiederverwendung, die in der Softwaretechnik Anwendung gefunden haben, analysiert.

5.2.2 Analyse von Techniken zur Wiederverwendung bewährter Lösungen

Aus der Softwaretechnik sind Techniken zur Wiederverwendung bewährter Lösungen bekannt. Dabei werden vorgefertigte generische funktionale Einheiten durch Instanziierung verwendet, ohne weder den inneren Aufbau noch das innere Verhalten der Einheiten kennen zu müssen bzw. selbstständig verstehen zu müssen. Stattdessen müssen die Funktionalität und die Schnittstellen dieser Einheiten klar definiert und beschrieben sein. Diese Aspekte kommen beim Einsatz von Mustern sowie bei der Verwendung von Softwarekomponenten zum Tragen. In den

folgenden beiden Abschnitten werden diese beiden Techniken aus der Softwareentwicklung mit der Zielsetzung vorgestellt und analysiert, geeignete Aspekte dieser Konzepte für die formale Spezifikation von Sicherheitsanforderungen zu adaptieren.

5.2.2.1 Wiederverwendung durch Komponenten

In der modernen Elektronik werden Systeme nicht mehr vollständig aus einzelnen Transistoren, Dioden und anderen einzelnen Bauelementen aufgebaut. Stattdessen werden universelle Bausteine (Integrated Circuits, ICs) verwendet. In Analogie dazu wird bei der komponentenbasierten Softwareentwicklung Software nicht komplett aus einzelnen Quellcodezeilen, sondern aus komplexeren Software-Komponenten erstellt. Softwaresysteme werden aus Softwarekomponenten mittels Auswahl, Konfigurieren und Verknüpfen zu einer Applikation zusammengebaut (vgl. Abbildung 5.1) [Göhn98].

Der Nutzen der komponentenbasierten Softwareentwicklung ist insbesondere die Wiederverwendbarkeit. Eine einmal erstellte Komponente kann mehrfach in verschiedenen Anwendungen eingesetzt werden. Dadurch wird der Implementierungsaufwand reduziert. Somit können Kosten gesenkt werden. Für solche Komponenten ist ein hoher Entwicklungs- und Qualitätssicherungsaufwand gerechtfertigt. Ein aus Komponenten aufgebautes System besitzt deshalb eine höhere Qualität, als wenn es individuell entwickelt worden wäre [Dujm02, S. 12-13].

Eine Softwarekomponente stellt einen wiederverwendbaren Systembaustein mit einer abgeschlossenen funktionalen Einheit dar, die einen bestimmten Problembereich repräsentiert [Göhn98]. Komponenten sind vorgefertigte Softwareeinheiten, die bewusst für die Mehrfachverwendung ausgelegt wurden. Durch Auswahl, Konfigurierung der Systemeigenschaften und Verknüpfung entstehen aus einzelnen generischen Komponenten vollständige Anwendungen [Göhn98]. Der Weg einer generischen zur konkreten Komponente führt im Wesentlichen über eine Instanziierung in Form einer Anpassung durch Konfigurierung und Verknüpfung. Über das Konfigurieren der Komponenteneigenschaften kann eine derartige Komponente an spezielle Anforderungen und Randbedingungen einer zu realisierenden Applikation angepasst werden.

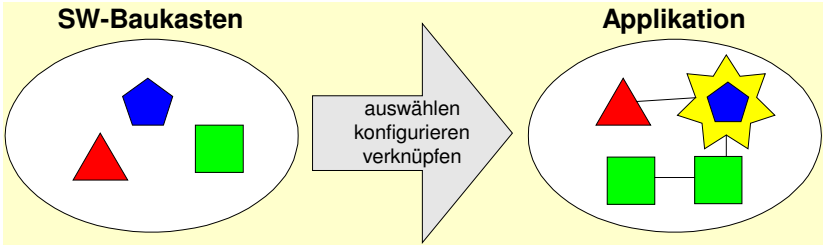


Abbildung 5.1: Komposition von Softwaresystemen aus Softwarekomponenten [Göhn98]

Um Softwaresysteme aus Komponenten aufbauen zu können, müssen die Softwarekomponenten bestimmte Eigenschaften aufweisen [Göhn98]. Die Schnittstellen der Softwarekomponente müssen vollständig spezifiziert sein. Sie muss gut dokumentiert und mehrfachverwendbar sein.

Letzteres bedeutet, dass die Softwarekomponente im Kontext unterschiedlicher Systeme verwendbar sein muss. Eine Softwarekomponente muss unabhängig von anderen Softwarekomponenten sein. Sie muss an eine gegebene Aufgabenstellung anpassbar sein und bezüglich bestimmter Eigenschaften konfigurierbar sein [Szyp97].

5.2.2.2 Wiederverwendung durch Muster

Für die effiziente Software- und Systementwicklung spielt die Wiederverwendung von Erfahrungen eine große Rolle. Die Lösungsfindung für bereits gelöste Probleme ohne Berücksichtigung bestehender Lösungen ist uneffektiv. Stattdessen repräsentieren Erfahrungen erfolgreiche und erprobte Lösungen, die für bestimmte Problembereiche eingesetzt werden können [Alex80]. Muster sind ein Mittel, um Erfahrungen zu dokumentieren. Sie sind allgemein betrachtet Lösungen, die sich in einem bestimmten Kontext als nützlich erwiesen haben [Balz99, S. 544] und auch für andere Zusammenhänge und die Entwicklung anderer Systeme geeignet erscheinen. Sie sind Lösungsvorschriften für ein typisches Problem [Gamm94], die durch Spezifikation, textuelle Erläuterung und Beispiele vermittelt werden. Somit sind Muster wiederverwendbare Lösungskonzepte. In der Softwaretechnik beschreiben Muster Strukturen, die sich in Softwaresystemen wiederholt finden und zur Lösung bekannter Probleme dienen. Die Zielsetzung ist die Qualitätsverbesserung durch Wiederverwendung, Wartbarkeit, Testbarkeit und Transparenz. Die Verwendung von Mustern verkürzt Entwicklungszeiten und bewirkt somit Kostenreduktion durch Wissenstransfer, Verbesserung der Dokumentation und Kommunikation.

Im Unterschied zur komponentenbasierten Softwareentwicklung ist die innere Struktur von Mustern vollständig zugänglich. Dabei sind Muster häufig Teillösungen während Komponenten im Vergleich zu Mustern komplexer sind und fertige Lösungsbausteine darstellen. Im Gegensatz zu Softwarekomponenten sind Muster niemals fertig codierte Lösungen, sondern Beschreibungen von Lösungsansätzen [Göhn05a]. Sie enthalten generalisierte Lösungsideen zu immer wiederkehrenden Softwareentwicklungsproblemen. Muster-Beschreibungen enthalten folgende Bestandteile (vgl. [Göhn04]):

- Nennung eines eindeutigen Namens zur Identifikation (eindeutige Bezeichnung)
- Beschreibung der Motivation des Musters (Erläuterung der Zielsetzung)
- Nennung der charakteristischen Merkmale (Festlegen der spezifischen Eigenschaften)
- Erläuterung anhand eines oder mehrerer Beispiele

5.2.2.3 Bedeutung der komponenten- und musterbasierten Entwicklung für die Spezifikation von Sicherheitsanforderungen

Für die Vermittlung von Expertenwissen bezüglich der Formalisierung von Sicherheitsanforderungen weisen beide Wiederverwendungs-Konzepte geeignete Aspekte auf, die sich adaptieren lassen. Expertenwissen kann ohne jegliche Voraussetzungen hinsichtlich formaler Spezi-

fikation vermittelt werden, indem vorgefertigte Spezifikationen zur Wiederverwendung bereitgestellt werden. Die Erstellung von vorgefertigten formalen Spezifikationen geschieht durch Experten für formale Methoden. Dabei ist ein hoher Spezifikationsaufwand gerechtfertigt, da solche Spezifikationen mehrfach wiederverwendet werden können. Entsprechend kann diese Erstellung mit der erforderlichen Sorgfalt geschehen.

Solche vorgefertigten Spezifikationen können wie Softwarekomponenten verwendet werden, von denen nur die entsprechenden Parameter für das Konfigurieren der Komponenteneigenschaften bekannt sein müssen [Bits00]. Eine Spezifikationskomponente besteht im „Inneren“ aus einer generischen Formulierung in Temporallogik. Die Konfigurierung der Parameter erfolgt durch die Angabe entsprechender Systemvariablen, über welche die generische Formulierung an das jeweilige zu entwickelnde Automatisierungssystem, d. h. an die gegebene Aufgabenstellung, angepasst wird. Eine Spezifikationskomponente ist eine generische Sicherheitsanforderung, die zu einer konkreten Sicherheitsanforderung durch Instanziierung überführt werden kann. Solche Spezifikationskomponenten können dabei für komplexere Sicherheitsanforderungen miteinander „verschaltet“ werden, in dem sie durch aussagenlogische Konnektoren miteinander verknüpft werden und ineinander eingesetzt werden.

Es reicht jedoch nicht aus, temporallogische Formeln bereitzustellen. Erstens muss das Verfahren auch die Spezifikation in natürlicher Sprache unterstützten (vgl. Abschnitt 4.3.3). Zweitens müssen die Komponenteneigenschaften bekannt sein und verständlich sein, ohne den inneren Aufbau der Komponente kennen zu müssen. Die vorgefertigten Spezifikationen müssen interpretiert werden können, ohne den formalen Aufbau verstehen zu müssen und ohne die dafür verwendete formale Sprache gebrauchen können zu müssen (vgl. Abschnitt 4.3.3). Vielmehr müssen die Bedeutung und die Anwendung der vorgefertigten Spezifikation intuitiv verständlich sein. Wesentlich für das Gelingen dieses Ansatzes ist folglich, dass diese generischen Sicherheitsanforderungen im Detail und verständlich dokumentiert werden, sodass die Bedeutung und die Anwendung dieser Spezifikationskomponenten jeweils klar ist.

Allerdings sollte auch berücksichtigt werden, dass es Entwickler gibt, die Experten für formale Methoden sind oder die zumindest Grundkenntnisse in Temporallogik besitzen. Auch wenn selbst für diese Personengruppen die korrekte Anwendung von Temporallogik schwierig ist [Heit98a] so geben sich solche Entwickler nicht unbedingt mit einem Verfahren zufrieden, bei dem die Spezifikation in Temporallogik gänzlich verborgen ist. Sie wollen verstehen, wie in Temporallogik spezifiziert wird. Das gesuchte Verfahrens-Konzept muss für verschiedene Personengruppen geeignet sein. Dabei kann durch den Einsatz eines Unterstützungssystems für die Wiederverwendung vorgefertigter Spezifikationen von Sicherheitsanforderungen Expertenwissen adaptiv für die unterschiedlichen Personengruppen vermittelt werden. Die Personen der einen Gruppe sollen nicht mit Temporallogik konfrontiert werden. Sie wollen nicht den Aufbau der vorgefertigten Spezifikationen kennen. Andere Personengruppen möchten hingegen den Formalismus der vorgefertigten Spezifikationen verstehen, um ihr vertrauen zu können oder aus

reinem Interesse. Für sie müssen die vorgefertigten Spezifikationen offen zugänglich sein. Für diese Personengruppe können keine Sicherheitsanforderungs-Komponenten in dem Sinn verwendet werden, dass der Aufbau der Komponente nicht zugänglich ist. Vielmehr gleichen unter diesem Gesichtspunkt vorgefertigte Sicherheitsanforderungen Mustern [Bits01]. Die Zielsetzungen Wissenstransfer, Verbesserung der Kommunikation und Dokumentation (vgl. Abschnitt 5.2.2.2) sind genau für die behandelte Problematik zutreffend. Entwicklern soll bei der Spezifikation von Sicherheitsanforderungen Erfahrungswissen vermittelt werden hinsichtlich der korrekten formalen Spezifikation. Dabei soll die Spezifikation so dokumentiert werden, dass sie in der Teamarbeit und bei Wartungsarbeiten von anderen Entwicklern klar verstanden werden kann.

Mustereigenschaften sind für solche vorgefertigten generischen Spezifikationen für Sicherheitsanforderungen in dem Sinn zutreffend, dass sie generalisierte Lösungsideen zu wiederkehrenden Spezifikationsproblemen sind. Sie können sinnvollerweise wie ein Muster durch einen eindeutigen Namen, durch die Zielsetzung, durch die charakteristischen Eigenschaften und ein oder mehrere Beispiele erläutert werden. Allerdings sind solche generischen Sicherheitsanforderungen im Gegensatz zu herkömmlichen Mustern nicht nur Beschreibungen von Lösungsansätzen, sondern umfassen fertig in Temporallogik codierte Lösungen, die lediglich noch instanziiert werden müssen. Im Übrigen sind für solche generischen Sicherheitsanforderungen alle Mustereigenschaften zutreffend. Ein geeigneter Name zur Bezeichnung dieser Art von Mustern (engl.: *pattern*) zur Spezifikation von Sicherheitsanforderungen ist „Safety-Pattern“.

Mit dem Verfahrens-Ansatz zur Spezifikation funktionaler Sicherheitsanforderung durch Wiederverwendung vorgefertigter Spezifikationen können somit die in Abschnitt 3.2 definierten und in Abschnitt 4.3.3 verfeinerten Anforderungen an die Anwendbarkeit eines geeigneten Verfahrens (/KA4/ und /KA5/) erfüllt werden. Es wird Expertenwissen für die Formulierung und Formalisierung von Sicherheitsanforderungen vermittelt, ohne dass eine direkte Spezifikation in einer formalen Sprache erforderlich ist. Das Konzept der Wiederverwendung verspricht ein effizientes, einfach erlernbares Verfahren. Dies kann zudem durch den Einsatz eines Beratungs- und Hilfesystem unterstützt werden. Damit auch die Anforderung /KA3/ *vollständige Spezifizierbarkeit* erfüllt wird, muss die Menge der Safety-Pattern die Spezifikation aller funktionalen Sicherheitsanforderungen für Automatisierungssysteme ermöglichen. Im Folgenden wird der Ansatz zur Spezifikation von Sicherheitsanforderungen mithilfe von Safety-Pattern analysiert und konkretisiert.

5.3 Formale Spezifikation von Sicherheitsanforderungen mit Safety-Pattern

5.3.1 Verwendung von Safety-Pattern zur Spezifikation von Sicherheitsanforderungen

Damit Safety-Pattern Mustereigenschaften besitzen, muss die Beschreibung der Safety-Pattern mindestens Folgendes umfassen (vgl. Abschnitt 5.2.2.2):

- Eindeutige Bezeichnung
- Generische Formulierung
- Nennung und Erläuterung der charakteristischen Eigenschaften sowie der Zielsetzung
- Erläuterung durch Musterbeispiel

Wesentlich für die komponenten- und musterbasierte Softwareentwicklung ist zudem eine Katalogisierung. Im Katalog müssen die Safety-Pattern eindeutig identifiziert werden können. Der eindeutigen Bezeichnung sowie den charakteristischen Eigenschaften kommt dabei Bedeutung für die eindeutige Identifizierbarkeit der Safety-Pattern zu. Die Erläuterung der charakteristischen Eigenschaften sowie die Verwendung von Musterbeispielen muss die klare Verständlichkeit der Safety-Pattern-Bedeutung und –Anwendung zum Ziel haben. Der wesentliche Kern eines Safety-Pattern bildet die generische Formulierung, mit deren Hilfe konkrete Sicherheitsanforderungen durch Instanziierung gebildet werden. Bevor im Detail untersucht wird, wie die Beschreibungen der Safety-Pattern im Einzelnen erfolgen müssen, wird zunächst noch eine Voraussetzung für die Eignung des Ansatzes behandelt. Der Ansatz behandelt die Problematik der Spezifikation von Sicherheitsanforderungen für Automatisierungssysteme nur dann hinreichend, wenn durch die Safety-Pattern alle Spezifikationsprobleme für die Formulierung von Sicherheitsanforderungen für Automatisierungssysteme abgedeckt werden können (vgl. Konzept-Anforderung /KA3/ *vollständige Spezifizierbarkeit*). Dabei muss die Menge der Safety-Pattern endlich sein. Sonst ist sie nicht realisierbar.

Der Grundgedanke des Ansatzes kann wie folgt zusammengefasst werden: Funktionale Sicherheitsanforderungen für Automatisierungssysteme werden mithilfe einer endlichen Menge von *Safety-Pattern* spezifiziert. Die Entwickler müssen in einem ersten Schritt das erforderliche Safety-Pattern aus einem Katalog selektieren. Im zweiten Schritt folgt die Instanziierung (vgl. Abbildung 5.2). Dabei werden Variablen des Systemmodells in die generischen Formulierungen eingesetzt. Gegebenenfalls folgt eine Komposition von Formulierungen durch die Verknüpfung mit aussagenlogischen Konnektoren oder das Einsetzen von Formulierungen in andere.

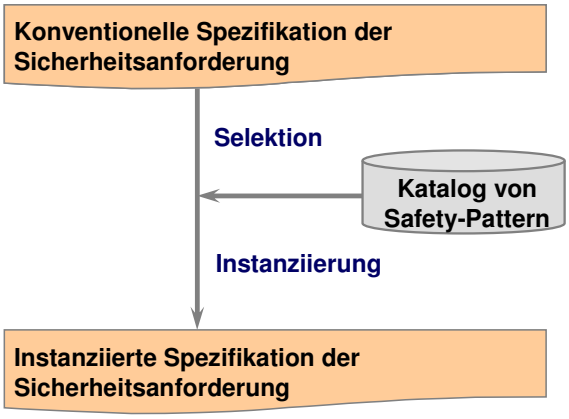


Abbildung 5.2: Grundschema der Spezifikation mithilfe von Safety-Pattern

Die unterschiedlichen Personenzielgruppen müssen bei der Entwicklung des Aufbaus der Safety-Pattern, der Auswahl der adäquaten Beschreibungsmittel für die generischen Formulierungen sowie für die Erläuterungen berücksichtigt werden, für die das Verfahren bestimmt ist. Gemäß Abschnitt 1.1 können im Entwicklungsprozess folgende Rollen unterschieden werden: Entwickler, Teamkollegen in der Entwicklung, Wartungsingenieure sowie Gutachter von Aufsichtsbehörden.

Entwickler müssen das passende Safety-Pattern finden, es verstehen und für die Spezifikation der Sicherheitsanforderungen korrekt anwenden können. Die Teamkollegen des Entwicklers müssen verstehen und nachvollziehen können, was dieser spezifiziert hat. Bei Wartungsingenieuren kommt es darauf an, die Bedeutung spezifizierter Sicherheitsanforderungen, schnell erfassen zu können, indem sie die hierzu gehörenden Safety-Pattern schnell ermitteln und die Bedeutung nachschlagen können. Ggf. haben auch Wartungsingenieure die Aufgabe Safety-Pattern für die Spezifikation neuer Sicherheitsanforderungen korrekt zu verstehen und anzuwenden. Gutachter von Aufsichtsbehörden müssen spezifizierte Sicherheitsanforderungen verstehen und das Spezifizieren von Entwicklern und Wartungsingenieuren nachvollziehen. Die für die Spezifikation verwendeten Safety-Pattern müssen also eindeutig identifiziert werden können. Der Instanziierungsschritt muss ausgehend von den generischen Formulierungen hin zu den konkreten Sicherheitsanforderungen klar nachvollzogen werden können.

Neben der Differenzierung dieser verschiedenen Rollen im Entwicklungsprozess ist für die Bestimmung der adäquaten Beschreibungsmittel des Verfahrens insbesondere der Kenntnisstand über den Gebrauch von Temporallogik wichtig. Es gibt bei allen vier aufgezählten Rollen folgende Personengruppen (Verfeinerung der Unterteilung aus Abschnitt 5.2.2.3):

1. *Experten für formale Methoden:* Diese Personen spezifizieren ggf. direkt in Temporallogik bzw. interpretieren direkt die temporallogische Formulierung. Sie wollen das Spezifikationskonzept ggf. verwenden, um zu prüfen, ob ihre Spezifikation bzw. Interpretation korrekt ist. Insbesondere kann das Verfahren für sie die Bedeutung haben, dass sie im Rahmen von Sicherheitsnachweisen die Richtigkeit der selbstständig formulierten oder interpretierten

Spezifikation systematisch belegen wollen. Dies erreichen sie, indem sie zeigen, dass man unter Verwendung des Verfahrens zu dem gleichen Ergebnis kommt. Durch das Verfahren, können sie die Spezifikationsschritte nachvollziehbar machen.

2. *Personen mit Grundkenntnissen in Temporallogik:* Diese Personen verwenden das Verfahren für die Spezifikation und Interpretation. Mithilfe ihrer Kenntnisse in Temporallogik wollen sie die Ergebnisse, die durch das Verfahren erzielt werden, nachvollziehen.

3. *Personen ohne Kenntnis in Temporallogik:* Diese Personen interessieren sich nicht für die temporallogische Formulierung. Sie wollen nur, wie es in den Normen gefordert ist, formal spezifizieren. Bei der Verwendung vorgefertigter Spezifikation müssen sie die Bedeutung derselben kennen und verlassen sich auf deren Korrektheit. Sie möchten formale Korrektheitsnachweise durchführen und nutzen, ohne des formalen Spezifizierens mächtig zu sein.

Safety-Pattern sind aus Sicht der Entwickler ohne Interesse am formalen Aufbau Spezifikations-Komponenten. Entwickler, welche den formalen Aufbau nachvollziehen möchten, gebrauchen Safety-Pattern als Muster.

5.3.2 Bedeutung der eindeutigen Charakterisierung der Safety-Pattern für die Anwendbarkeit

Ziel der Katalogisierung in der komponenten- und musterbasierten Softwareentwicklung ist, dass die Komponenten bzw. Muster klar voneinander unterschieden werden können und das gesuchte Objekt einfach und schnell identifiziert werden kann. Aus diesen Gründen ist auch eine Katalogisierung der Safety-Pattern wichtig. Der Safety-Pattern-Ansatz ist nur dann anwendbar, wenn die Safety-Pattern eindeutig identifizierbar sind und sich systematisch und einfach selektieren lassen.

Voraussetzung für die geordnete Katalogisierung ist eine Klassifizierung, d. h. die Safety-Pattern müssen klassifiziert werden. Jedes Safety-Pattern muss zu diesem Zweck über Eigenschaften eindeutig identifizierbar sein. Jedes Safety-Pattern besitzt Eigenschaften, die es von allen anderen unterscheidet und die es eindeutig charakterisieren. Diese Eigenschaften können als Klassen aufgefasst werden. Aus Klassen, die dasselbe Thema behandeln, können Klassifikationskriterien abgeleitet werden. Ein solches Kriterium ist z. B. „Art der Systemsteuerung“. Die entsprechenden Klassen sind in diesem Fall „zeitgesteuertes System“ und „ereignisgesteuertes System“.

Wird nun aus dieser Klassifikation ein Klassifikationsbaum aufgebaut, können die Safety-Pattern anhand dieses Baumes systematisch selektiert werden. Dabei erfolgt die Charakterisierung des gesuchten Safety-Pattern über die Charakterisierung durch die zutreffenden Klassen. Die Knoten des Klassifikationsbaums sind dabei die Klassifikationskriterien. Die Blätter des Baumes sind die Safety-Pattern (vgl. Abbildung 5.3).

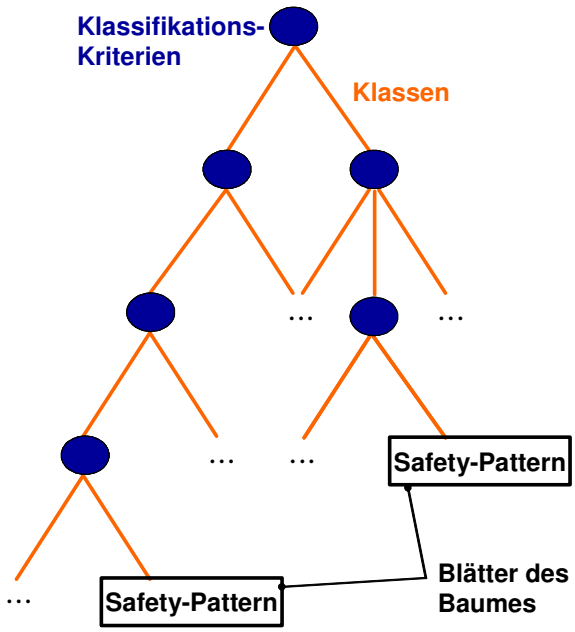


Abbildung 5.3: Klassifikationsbaum zur Safety-Pattern-Selektion

Ein weiteres wesentliches Merkmal des Komponenten- und Musterkonzepts in der Softwareentwicklung ist, dass die Komponenten bzw. Muster gut dokumentiert sind. Die Eigenschaften eines Safety-Pattern müssen nicht nur aufgelistet, sondern auch das Ergebnis der jeweiligen Eigenschaftskombination beschrieben werden (vgl. Abschnitt 5.3.7).

5.3.3 Bedeutung der Vollständigkeit des Safety-Pattern-Katalogs für die Anwendbarkeit

Der Ansatz ist für die vorgestellte Problematik nur dann hinreichend, wenn alle Arten von funktionalen Sicherheitsanforderungen mithilfe einer endlichen Menge von Safety-Pattern beschrieben werden können (vgl. Abschnitt 5.3.1). Die Safety-Pattern-Menge muss Grundlage für die Spezifikation aller funktionalen Sicherheitsanforderungen für Automatisierungssysteme sein. Wie bereits in Abschnitt 5.1 ausgeführt wurde, eignet sich Temporallogik zur Spezifikation aller funktionalen Sicherheitsanforderungen für Automatisierungssysteme. Mithilfe der Safety-Pattern-Menge müssen folglich alle temporallogischen Formulierungen gebildet werden können, die für die Spezifikation funktionaler Sicherheitsanforderungen erforderlich sind. Dafür muss die Safety-Pattern-Menge mindestens genau die Safety-Pattern umfassen, deren Komposition dies erlaubt. Durch die Bestimmung aller sicherheitsrelevanten grundlegenden atomaren temporallogischen Ausdrücke wird dies erreicht. Dabei bezeichnet „atomare Formulierungen“ die Formulierungen, für die eine Aufteilung in noch kleinere Ausdrücke unmöglich oder nicht sinnvoll ist. Welches diese grundlegenden atomaren temporallogischen Ausdrucksmöglichkeiten sind, ist bei der Realisierung des Konzepts für ein Verfahren genauer zu untersuchen (s. Abschnitt 6.1.2). Die Bestimmung kann wiederum in zwei Teilschritte gegliedert werden:

a. Bestimmung, wodurch grundlegende atomare Formulierungen in Temporallogik gekennzeichnet sind.

b. Bestimmung, wodurch sicherheitsrelevante Formulierungen charakterisiert sind.

Aus dieser Bestimmung resultiert dann die *Menge der notwendigen generischen Formulierungen.*

Unter Berücksichtigung der Konzept-Anforderungen hinsichtlich *Anwendbarkeit* (/KA4/ *Handhabbarkeit* und /KA5/ *Verständlichkeit und Nachvollziehbarkeit* vgl. Abschnitt 3.2) muss die Menge der Safety-Pattern zudem eine handhabbare Größe besitzen. Das bedeutet einerseits, dass die Safety-Pattern-Menge nicht zu klein sein darf. Sonst kann sich die erforderliche Komposition schwierig gestalten. Es müssen also zusätzlich zu der Menge der notwendigen generischen Formulierungen weitere Ausdrücke bestimmt werden, welche die Anwendbarkeit der Safety-Pattern-Menge erhöhen. Andererseits darf die Safety-Pattern-Menge nicht zu groß sein, damit eine einfache Selektion möglich ist (vgl. [Bits03a]). Die zusätzlichen Ausdrücke sind dabei keine gänzlichen neuen Ausdrücke, sondern aus der Menge der notwendigen ableitbar. Sie bilden die *Menge der nützlichen generischen Formulierungen*.

5.3.4 Vermittlung von Expertenwissen durch generische Formulierungen

5.3.4.1 Generische Spezifikation in formalen Sprachen

Der Kern der Safety-Pattern ist die generische Formulierung in Temporallogik. Hierdurch werden vorspezifizierte Formulierungen bereitgestellt. Durch sie wird der Entwickler von den Schwierigkeiten entlastet, die mit ihrer Erstellung zusammenhängen. Diese Formulierungen können mit einer Sorgfalt erstellt werden, für die der Entwickler nicht die erforderliche Zeit aufbringen kann. Insbesondere fehlen Entwicklern jedoch die erforderlichen Kenntnisse und insbesondere die notwendigen Erfahrungen für den korrekten Gebrauch von Temporallogik. Somit findet durch diese generischen Formulierungen in Temporallogik ein Wissens- und Erfahrungstransfer statt. Durch die Anwendung dieser generischen Spezifikation wird die formale Spezifikation von Sicherheitsanforderungen ermöglicht, welche die Grundlage für formale Sicherheitsnachweise ist. Eine solche generische Formulierung ist z. B.:

$$A\ (\text{not}\ b\ W\ a) \qquad (5.1)$$

Dabei sollen für die im Safety-Pattern vorkommenden Variablen (Safety-Pattern-Variablen) a und b boolesche Systemvariablen eingesetzt werden, womit eine bestimmte Relation zwischen diesen Systemgrößen gefordert werden kann. In diesem Fall ist die Forderung, dass b erst ab dem Zeitpunkt wahr sein darf, zu dem a gültig ist. a ist die *Bedingungsvariable*. Durch die Aus-

sage a wird die Bedingung festgelegt, durch welche die Geltung der Eigenschaft, welche durch die *Eigenschaftsvariable* b beschrieben wird, bedingt ist. Somit ist b abhängig von a: $b(a)$

Die Formel (5.1) ist in CTL formuliert. Im Hinblick auf die formale Verifikation ist eine solche Formulierung nur für solche Verifikationswerkzeuge nützlich, bei denen die zu verifizierenden Anforderungen in CTL formuliert werden können. Um die Entwickler bei der Verwendung von Safety-Pattern nicht auf die Verwendung von Werkzeugen festzulegen, welche die verwendete temporallogische Sprache unterstützen, ist es erforderlich die generischen Formulierungen in den verschiedenen gängigen temporallogischen Sprachen anzugeben [Bits03a]. Zudem können dann mithilfe von Safety-Pattern Sicherheitsanforderungen, die in einer bestimmten temporallogischen Sprache bereits spezifiziert sind, leicht in andere temporallogische Sprachen umgesetzt werden (Übersetzungsvorschrift).

Die generischen Formulierungen sind im erforderlichen Format für formale Verifikationswerkzeuge, sodass sie direkt für die formale Verifikation verwendet werden können. Zudem müssen sie für die Personengruppen zugänglich sein, die direkt in Temporallogik spezifizieren oder den Gebrauch des verwendeten Formalismus verstehen möchten (vgl. 1. und 2. Personengruppe in Abschnitt 5.3.1).

In dieser Weise vermitteln die generischen Formulierungen Expertenwissen hinsichtlich der formalen Spezifikation von Sicherheitsanforderungen. Gemäß Abschnitt 4.3.3 muss das geeignete Verfahren zudem die Spezifikation von Sicherheitsanforderungen in Worten der natürlichen Sprache unterstützen. Der Safety-Pattern-Ansatz vermittelt auch Expertenwissen hinsichtlich der Spezifikation in natürlicher Sprache, wenn die Safety-Pattern neben generischen Formulierungen in Temporallogik auch generische Spezifikationen in natürlicher Sprache umfassen.

5.3.4.2 Generische Spezifikation in Normsprache

In den Abschnitten 2.2.5 und 4.3 wurde der Nutzen und die Notwendigkeit ausgeführt, für die formale Spezifikation einer Sicherheitsanforderung eine Formulierung in Worten oder Sätzen der natürlichen Sprache anzugeben, deren Bedeutung äquivalent zur formalen ist. Die Forderung der Normen wird erfüllt und das einfache intuitive Verständnis für formale Sicherheitsanforderungen durch eine Dokumentation der Sicherheitsanforderung ermöglicht. Dabei können Fehler in der Formulierung sowie nicht äquivalente Formulierungen vermieden werden, indem der Ansatz die Formulierung in Worten der natürlichen Sprache unterstützt und dies nicht den Entwicklern überlässt. Beim Safety-Pattern-Ansatz geschieht diese Unterstützung, indem die Safety-Pattern auch generische Formulierungen in Worten der natürlichen Sprache umfassen, die für die Spezifikation konkreter Sicherheitsanforderungen instanziiert werden müssen. Somit stellen Safety-Pattern eine Brücke zwischen der Anwendung natürlicher Sprache und formaler Sprache zur Spezifikation von Sicherheitsanforderungen dar. Dies ist insbesondere für die Anwender des Verfahrens erforderlich, die nicht mit Temporallogik konfrontiert werden

möchten (vgl. Abschnitt 5.3.1). Aber auch für Anwender, die nur den Formalismus nachvollziehen möchten, soll die Spezifikation nicht in Temporallogik geschehen. Für Experten für formale Methoden schließlich, können solche generischen Spezifikationen auch für das korrekte Verständnis beim Spezifizieren hilfreich sein.

Ungeschickte und mehrdeutige Formulierungen in natürlicher Sprache können für diese generischen Formulierungen vermieden werden, indem die natürliche Sprache in Form einer Normsprache eingeschränkt wird. Für die generischen Spezifikationen ist es deshalb sinnvoll eine Normsprache zu verwenden. Diese Normsprache ist nicht für den Kontext der Entwicklung eines bestimmten Systems, sondern für die Spezifikation von Sicherheitsanforderungen mit Safety-Pattern definiert. Es ist somit eine Safety-Pattern-Normsprache. Der Aufbau der Sprache erfolgt nicht willkürlich, sondern anhand klarer und wissenschaftlich hergeleiteter Regeln [Ortn97]. Alle Begriffe der Terminologie dieser Safety-Pattern-Normsprache werden dabei klar definiert, sodass jeder Begriff der normierten Sprache eine eindeutige Bedeutung besitzt. Dadurch wird eine präzise natürlich-sprachliche Spezifikation von Sicherheitsanforderungen unterstützt.

Im Gegensatz zu anderen Ansätzen (vgl. Abschnitt 4.2.1) werden hiermit keine einzelnen Worte oder Teilsätze vorgegeben, die der Benutzer selbstständig anwenden muss und die zu unnatürlichen Formulierungen führen. Stattdessen wird für jedes Safety-Pattern ein vollständiger Satz oder ggf. mehrere Sätze angegeben. Satzbau und die Wortwahl werden spezifisch für das Safety-Pattern gewählt. Dadurch werden künstliche Formulierungen vermieden und die Natürlichkeit der Formulierungen unterstützt. Zudem ist es sinnvoll Formulierungsvarianten anzugeben, sodass der Benutzer die Formulierung wählen kann, die vom Satzbau her als auch von den verwendeten Begriffen, für die zu spezifizierende Sicherheitsanforderung am natürlichsten wirkt.

Somit unterstützen Safety-Pattern nicht nur die formale Spezifikation von Sicherheitsanforderungen, sondern auch präzise formulierte Spezifikationen in natürlicher Sprache (vgl. [BiGö02]). Die Angabe der generischen Formulierungen in weiteren textuelle Beschreibungsmitteln (vgl. Abschnitt 2.3.1) ist nicht sinnvoll. Natürliche Sprache bringt keine weiteren Vorteile im Vergleich zur Safety-Pattern-Normsprache, sondern aufgrund von Mehrdeutigkeiten nur Nachteile. Der Safety-Pattern-Ansatz kann auch den Vorteil der Spezifikation in grafischer Notation unterstützen, wenn die Safety-Pattern neben generischen Formulierungen in der Safety-Pattern-Normsprache für die Dokumentation von formalen Sicherheitsanforderungen auch generische Spezifikationen in grafischer Notation umfassen.

5.3.4.3 Generische Spezifikation in grafischer Notation

Wenn die Safety-Pattern auch generische Spezifikationen in geeigneter grafischer Notation mit einschließen, kann der Vorteil genutzt werden, die Bedeutung der generischen temporal-

logischen Spezifikationen zu veranschaulichen. Während des Spezifizierens werden auf diese Weise die jeweiligen komplizierten Zusammenhänge visualisiert. Damit wird ähnlich wie mit der Angabe generischer Formulierungen in der Safety-Pattern-Normsprache die Dokumentation einer formalen Sicherheitsanforderung durch eine äquivalente Beschreibung unterstützt. Immer mehr gewinnen in der Spezifikation von technischen Systemen grafische Notationen an Bedeutung, vgl. [Ehri04]. Somit wird durch die Bereitstellung generischer Spezifikationen in grafischer Notation, diesem Trend, der letztlich auf die Anforderungen der Anwender zurückzuführen ist, Rechnung getragen. Dabei lässt sich nicht objektiv und mit allgemeiner Gültigkeit feststellen, ob eine textuelle oder eine grafische Beschreibung einfacher verständlich ist. Dies kann von verschiedenen Personen unterschiedlich empfunden werden. Durch die Verwendung von textueller und grafischer Notation für die einfach verständliche Dokumentation von Sicherheitsanforderungen werden die Vorteile beider Ansätze unterstützt.

Aufgrund der Veranschaulichung ist diese Art der Spezifikation insbesondere für die Personengruppe vorteilhaft, welche Grundkenntnisse in Temporallogik besitzen und die formale Spezifikation verstehen möchten. Zudem ist es für die Personengruppe ohne Kenntnis in Temporallogik hilfreich, um die Korrektheit der Interpretation der generischen Spezifikation in der Safety-Pattern-Normsprache zu prüfen.

An eine solche grafische Notation müssen die folgenden Anforderungen gestellt werden. Den Anforderungen 1 und 2 ist dabei gegenüber den Anforderungen 3 und 4 eine höhere Bedeutung beizumessen:

1. *Einfach interpretierbare Notation:* Die Notation muss möglichst intuitiv verständlich sein, sodass Spezifikationen einfach und ohne großen Lernaufwand interpretiert werden können.

2. *Spezifizierbarkeit aller Safety-Pattern:* Die grafische Notation ist nur dann für die Integration in den Safety-Pattern-Ansatz hinreichend, wenn für alle Safety-Pattern eine generische Formulierung in dieser grafischen Notation angegeben werden kann.

3. *Verwendbarkeit auch ohne Softwarewerkzeug:* Die Notation soll auch ohne Softwarewerkzeug verwendbar sein, z. B. mit einem einfachen Zeichenprogramm oder mit Bleistift und Papier einfach. Hierfür ist es von Vorteil, wenn Farben keine Semantik zugeordnet wird.

4. *Wenig aufwendige Spezifikation:* Für die Realisierbarkeit ist es hilfreich, wenn die Spezifikationen an sich wenig aufwendig sind. Dies kann zudem für die einfache und schnelle Interpretierbarkeit von Vorteil sein.

An die Art und Weise der Darstellung werden die folgenden Anforderungen definiert:

- *Eignung zur Darstellung zeitlicher Abhängigkeiten:* Bei den generischen temporallogischen Formulierungen werden zeitlich-logische Zusammenhänge formuliert. Diese müssen in der grafischen Notation darstellbar sein.

- *Verwendung einer zustandsbasierten Darstellung:* Mit der grafischen Notation sollen temporallogische Ausdrücke abgebildet werden. In solchen Ausdrücken werden Aussagen über einem Zustandsraum gemacht, vgl. Abschnitt 2.4.1.3. Das bedeutet, dass die Denkweise bei der Spezifikation mit Temporallogik zustandsbasiert ist, vgl. [Béra01]. Entsprechend ist die Prüfung bei der formalen Verifikation durch Model-Checking *zustandsbasiert.* Folglich müssen in der geeigneten grafischen Notation die zu fordernden Relationen zwischen Systemvariablen zustandsbasiert ausgedrückt werden können.

Auf der Grundlage dieser Anforderungen muss bei der Entwicklung des Safety-Pattern-Konzepts eine geeignete grafische Notation ermittelt werden.

5.3.5 Instanziierung von Sicherheitsanforderungen

Nach der Safety-Pattern-Selektion folgt die Instanziierung (vgl. Abbildung 5.2). Bei der Instanziierung geht es darum, generische Formulierungen in die Spezifikation für konkrete Sicherheitsanforderungen zu überführen. Dabei müssen die Systemvariablen, für welche die Relation ausgedrückt werden soll, in die selektierte generische Formulierung eingesetzt werden. Dies können einzelne Systemvariablen oder aussagenlogische Verknüpfungen von Systemvariablen (boolesche Ausdrücke) sein. In die generische Formulierung 5.1 `A (not b W a)` können z. B. die folgenden Ausdrücke mit Variablen eines Systemmodells für eine Bahnübergangssteuerung eingesetzt werden:

```
a := (Bahnuebergang.Bahnuebergang_wurde_passiert or
      FFB-Zentrale.Ausschaltbefehl)

b := Bahnuebergang.Schranke_oeffnen
```

Hier wird, wie dies bei der Spezifikation für viele Verifikationswerkzeuge üblich ist, vor der eigentlichen Variablen das Teilsystem bzw. das Teilmodell genannt, zu dem diese Variable gehört. Das Resultat ist die Formulierung 5.2. Diese Sicherheitsanforderung drückt aus, dass die Schranken eines Bahnübergangs so lange nicht geöffnet werden dürfen, bis der Bahnübergang das Signal erhält, dass er vom Eisenbahnfahrzeug passiert wurde oder der Bahnübergang von der FFB-Zentrale einen Ausschaltbefehl erhält.

```
A (not Bahnuebergang.Schranke_oeffnen W
   (Bahnuebergang.Bahnuebergang_wurde_passiert or
    FFB-Zentrale.Ausschaltbefehl))            (5.2)
```

Zur Vermeidung von Fehlern bei der Instanziierung ist es wichtig, dass die Instanziierung werkzeuggestützt geführt und kontrolliert wird. Das bedeutet, dass der Benutzer bei der Erstellung korrekter boolescher Ausdrücke angeleitet werden muss. Hierfür muss sichergestellt werden, dass Verknüpfungs-Operatoren in korrekter Weise für die unterschiedlichen Variablentypen gebraucht werden. Zum Beispiel dürfen Integer-Variablen nicht direkt mit UND-

Operatoren verknüpft werden. Für solche Variablen muss zunächst durch Vergleichsoperatoren (relationale Operatoren: <, > oder =) eine boolesche Aussage über Integervariablen erfolgen. In dieser Weise muss die korrekte Syntax von booleschen Ausdrücken bei der Instanziierung sichergestellt werden (vgl. Abschnitt 2.4.1.1).

Bei der Instanziierung muss es zudem möglich sein, Aussagen über die Werteänderung von Systemvariablen (z. B. Eintritt in oder Austritt aus Zustand) zu treffen. In der Praxis sind oft Signaländerungen Auslöser von Ereignissen. Ein Beispiel wird aus dem folgenden Auszug einer Zugtürsteuerung-Spezifikation deutlich: „*Die Türen werden bei Wegnahme des Seitenfreigabesignals blockiert.*“

Beim Instanziierungsschritt muss zudem die Möglichkeit der Komposition von Safety-Pattern zur Spezifikation komplexer Sicherheitsanforderungen bestehen. Das bedeutet, dass Safety-Pattern aussagenlogisch miteinander verknüpft oder ineinander eingesetzt werden [Bits03a]. Auch hierbei muss der Benutzer für eine korrekte Komposition geführt werden. Es müssen entsprechende Kontrollmechanismen vorhanden sein, sodass der Entwickler durch die von ihm vorgenommene Verknüpfung, die gewünschte Bedeutung erzielt.

5.3.6 Interpretation von Sicherheitsanforderungen

Die Interpretation von Sicherheitsanforderungen muss für Experten für formale Methoden direkt auf der Grundlage der temporallogischen Spezifikation erfolgen können. Für die Mehrheit der Benutzer des Verfahrens muss die Spezifikation auf der Grundlage der Dokumentation in der Safety-Pattern-Normsprache oder in der grafischen Notation einfach verständlich sein. Zur Kontrolle des Verständnisses dieser Dokumentationen ist es wichtig, dass eindeutig ist, auf Basis welcher Safety-Pattern die formale Spezifikation erfolgte. Deshalb muss bei der Spezifikation auch mit angegeben werden, welche Safety-Pattern verwendet wurden. Dann kann die Bedeutung der formal spezifizierten Sicherheitsanforderungen einfach nachgeschlagen werden. Dies ist einer der Gründe dafür, dass zu jedem Safety-Pattern eine ausführliche und detaillierte Beschreibung der Bedeutung mit angegeben werden muss. Die Art und Weise der Beschreibung wird im nächsten Unterkapitel analysiert.

Für temporallogische Formulierungen von Sicherheitsanforderungen, die nicht mithilfe von Safety-Pattern erstellt wurden, ist es zudem nützlich, dass über Suchfunktionen die zu temporallogischen Formulierungen entsprechenden Safety-Pattern selektiert werden können, und so die Bedeutung der Formulierung nachgeschlagen werden kann.[6]

[6] Solch ein Nachschlagen ist nicht für alle beliebigen Formulierungen in Temporallogik erfolgreich. Eine Sicherheitsanforderung kann in Temporallogik häufig durch verschiedene Formulierungen ausgedrückt werden (vgl. Abschnitt 6.1.2.1). In den Safety-Pattern findet sich i. d. R. nur eine von mehreren äquivalenten Formulierungen.

5.3.7 Erläuterung der Safety-Pattern zur Unterstützung der klaren Verständlichkeit von Safety-Pattern

5.3.7.1 Erforderlichkeit der Erläuterung von Safety-Pattern

Die Zielsetzung bei der Erläuterung der Muster ist die Vermittlung eines klaren Verständnisses für die Bedeutung der generischen und instanziierten Formulierungen. Die ausführliche Erläuterung gehört zum Grundprinzip der komponenten- und musterbasierten Entwicklung, vgl. Abschnitt 5.2.2.3. Es ist sinnvoll, zusätzlich zur generischen Spezifikation in natürlicher Sprache und in grafischer Notation noch Erläuterungen zu verwenden, da die Zielsetzung eine andere ist. Bei den generischen Spezifikationen geht es um kurze und prägnante wiederverwendbare Formulierungen, die eine Dokumentation der formal spezifizierten Sicherheitsanforderungen erlauben sollen. Zusätzlich ist es sinnvoll und für die Anwendung des Ansatzes hilfreich, in Erläuterungen für die Safety-Pattern ausführliche Erklärungen zu geben, um die Bedeutung des Safety-Pattern mit seinen Eigenschaften verständlich zu machen. Solche Erläuterungen werden zum einen gebraucht, um die Bedeutung spezifizierter Sicherheitsanforderungen im Detail verstehen zu können. Zum Beispiel ist es insbesondere für Gutachter wichtig, anhand umfassender Erläuterungen die von den Entwicklern verwendeten Spezifikationen prüfen zu können. Zum anderen bietet solch eine Erläuterung für Entwickler die Kontrolle, ob über die Auswahl der erforderlichen Safety-Pattern-Eigenschaften das richtige Safety-Pattern selektiert wurde. Ziel der Erläuterungen ist deshalb die klare Verständlichkeit durch die Angabe von Detailinformationen. Verschiedene Aspekte können hier im Detail und aus unterschiedlichen Perspektiven beschrieben werden.

5.3.7.2 Möglichkeiten zur Erläuterung von Safety-Pattern und ihre Bewertung

Generell ist eine Erläuterung in folgender Form möglich:

- *Formale Spezifikationssprache:* Die Beschreibung der Erläuterung in formaler Sprache ist sinnlos. Das Verfahren soll Schwierigkeiten bei der formalen Spezifikation von Sicherheitsanforderungen beseitigen. Diesen Schwierigkeiten kann man nicht wiederum durch einen anderen Formalismus begegnen.

- *Natürliche Sprache:* Natürliche Sprache ist intuitiv verständlich. Dabei ist für solche Erläuterungen die Gefahr von der Mehrdeutigkeit vernachlässigbar, da es in diesen Erläuterungen nicht auf kurze prägnante wiederverwendbare Formulierung ankommt, sondern ausführlich, diversitär und detailliert beschrieben werden kann. Zudem ist wegen der Wiederverwendbarkeit für die Erstellung der Safety-Pattern ein hoher Entwicklungsaufwand gerechtfertigt, sodass die Formulierungen durch Experten für die formale Spezifikation von Sicherheitsanforderungen sorgfältig ausgewählt werden können.

- *Normsprache:* Der Einsatz von Normsprachen ist nur dann sinnvoll, wenn der Terminologieumfang überschaubar und leicht erfassbar ist. Deshalb ist der Einsatz für generische Spezifikationen sinnvoll. Für die ausführliche Erläuterung der Safety-Pattern sind jedoch zu viele Worte erforderlich, sodass der Lernaufwand zur korrekten Beherrschung der Normsprache zu hoch wäre.

- *Erläuternde Grafiken:* Der Einsatz von Grafiken ist generell sinnvoll, da komplizierte Zusammenhänge veranschaulicht werden können. Grafiken sind dann intuitiv verständlich, wenn vertraute Darstellungsweisen verwendet werden. Im Gegensatz zur generischen Spezifikation in grafischer Notation sind hier keine allgemein gültigen generischen Ausdrücke erforderlich. Stattdessen kann die Veranschaulichung durch Beispiele geschehen, in denen die logischen Zusammenhänge erfüllt sind. Dabei müssen insbesondere die folgenden beiden Aspekte visualisiert werden, um die Relation zwischen den Safety-Pattern-Variablen zu veranschaulichen:

  1. Die zeitliche Veränderlichkeit der Safety-Pattern-Variablen
  2. Verschiedene Möglichkeiten zeitlicher Verläufe

- *Grafische Spezifikationssprache:* Der Einsatz einer solchen Sprache für die dokumentierende Spezifikation von Sicherheitsanforderungen wurde in Abschnitt 5.3.2 begründet. Erläuterungen aus anderen Perspektiven sind jedoch nur dann möglich, wenn hierfür die grafische Notation um redundante Beschreibungsmittel angereichert wird oder noch weitere grafische Notationen verwendet werden. Da der Lernaufwand in beiden Fällen groß ist, ist diese Art der Erläuterung nicht sinnvoll.

- *Beispiele für Modelle für Systemfunktionen:* Um die Bedeutung zu verdeutlichen, können auch beispielhaft Modelle für Systemfunktionen aufgeführt werden, welche die jeweilige generische Sicherheitsanforderung erfüllen. Dabei ist die Verwendung gängiger Notationen zur Verhaltensmodellierung sinnvoll, wie UML-Zustandsdiagramme, Petri-Netzen, SFC etc. . Dabei müssten die Safety-Pattern-Variablen, durch welche der generische Formulierungscharakter erreicht wird, im Modell der Systemfunktionen als Ereignis-, Zustands-, Attributs-, Bedingungs- oder Konfigurationsvariablen vorkommen. Allerdings wird durch solche Modelle immer nur ein Beispiel einer Verhaltensstruktur gegeben, welche die generische Sicherheitsanforderung einhält. Eine temporallogische Formulierung kann durch unendlich viele verschiedene Modelle erfüllt werden, da die temporallogische Formulierung nur eine Relation zwischen Systemgrößen beschreibt. Ein Modell der Systemfunktionen beschreibt hingegen eine ganz bestimmte Lösung. Deshalb ist diese Art der Erläuterung nicht sehr hilfreich. Da immer das Beispiel eines Modells für Systemfunktionen angegeben werden kann, das neben der gefragten Relation auch noch andere Aspekte beschreibt, kann dies sogar eher irreführend sein.

Zusammenfassend ist die Erläuterung der Safety-Pattern in natürlicher Sprache sowie durch erläuternde Grafiken sinnvoll. Die Verwendung beider Möglichkeiten ist zweckmäßig, da so das korrekte Verständnis der einen Erläuterung durch die Interpretation der anderen erprobt werden kann. Zur Erläuterung der Eigenschaften des Safety-Pattern gehört auch die Abgrenzung gegenüber ähnlichen Safety-Pattern. Hierzu ist es hilfreich, dass Safety-Pattern mit ähnlichen Eigenschaften in der Safety-Pattern-Erläuterung angegeben werden. Durch Vergleich kann die Selektion kontrolliert werden.

5.3.8 Anleitung zur Anwendung durch Musterbeispiele

Wesentlich für ein Musterkonzept ist, dass die Anwendung anhand mindestens eines Beispiels exemplarisch gezeigt wird. Dadurch wird die Bedeutung der generischen Formulierung im Kontext einer konkreten Anwendung deutlich gemacht.

Für das Safety-Pattern-Konzept bedeutet dies, dass ein Automatisierungssystem genannt wird, für das eine Sicherheitsanforderung konventionell spezifiziert ist. Ausgehend davon wird dann die Anwendung der generischen Formulierung durch Instanziierung verdeutlicht.

5.3.9 Bedeutung der Werkzeugunterstützung für die Anwendbarkeit

Bereits in den vorangehenden Unterkapiteln bezüglich der eindeutigen Charakterisierung, der Instanziierung und Interpretation von Safety-Pattern wurde deutlich, dass das Verfahren die Unterstützung durch ein Werkzeug erfordert. Es gibt automatisierbare Schritte, die für die Anwendbarkeit sowie für die effiziente Anwendung des Verfahrens durch ein Softwarewerkzeug unterstützt werden müssen (Unterstützungssystem). Dadurch wird eine grundlegende Voraussetzung für die Anwendbarkeit des geeigneten Konzepts erfüllt (vgl. Abschnitt 3.2). Zusammenfassend müssen folgende Schritte werkzeuggestützt realisiert werden. Dabei lassen sich die Aufgaben in zwei Bereiche einteilen: zum einen die Selektion von Safety-Pattern und zum anderen die Instanziierung.

1. *Selektion:*

   - *Charakterisierung der zu spezifizierenden Sicherheitsanforderung:* Der Selektionsprozess, wie er in Abschnitt 5.3.2 beschrieben wurde, kann effizient erfolgen, wenn er dialogbasiert ist. Der Anwender wird systematisch nach den Eigenschaften der zu spezifizierenden Sicherheitsanforderung gefragt und das Werkzeug wählt dazu das geeignete Safety-Pattern aus.

   - *Möglichkeiten zur Kontrolle der Selektion:* Ein Werkzeug sollte den Aufruf der für die Abgrenzung angegebenen Safety-Pattern unterstützen (vgl. Abschnitt 5.3.2).

- *Nachschlagen über Safety-Pattern-Bezeichnung:* Zur Unterstützung der effizienten Interpretation spezifizierter Sicherheitsanforderungen müssen Safety-Pattern einfach und direkt aufrufbar sein. Sinnvollerweise geschieht dies über die eindeutige Safety-Pattern-Bezeichung.

- *Suchefunktionen:* Für die Interpretation temporallogischer Formulierungen, die ohne Safety-Pattern erstellt wurden (vgl. Abschnitt 5.3.6), sind Suchefunktionen wichtig. Diese können insbesondere auch für Experten für formale Methoden interessant sein, die direkt formal spezifizieren und hierfür zur Kontrolle oder als Beleg für die korrekte Spezifikation die jeweiligen Safety-Pattern aufrufen wollen.

- *Verwaltung selektierter Safety-Pattern:* Ein wichtiger Aspekt der Anwendbarkeit ist die Mehrfachverwendung (vgl. Abschnitt 3.2). Hierfür ist es wichtig, dass einmal selektierte Safety-Pattern zusammen mit dem Weg der Selektion abgespeichert werden können und bei Auftreten des gleichen Spezifikationsproblems wieder verwendet werden können.

2. *Instanziierung:*

- *Führung bei der Instanziierung:* Die Führung und Kontrolle bei der Aufstellung syntaktisch korrekter aussagenlogischer Ausdrücke und bei der Komposition von Safety-Pattern gemäß der Intention des Entwicklers kann nur über eine dialogbasierte Werkzeugunterstützung erfolgen. Für eine Kontrolle der Syntax muss der Typ der für die Instanziierung verwendeten Variablen bekannt sein. Deshalb muss die Werkzeugunterstützung auch die Deklaration von Variablen miteinschließen. Unter dieser Voraussetzung kann der korrekte Variablengebrauch bei der Instanziierung kontrolliert werden.

- *Verwaltung der instanziierten Sicherheitsanforderungen:* Für die Mehrfachverwendung ist es wichtig, dass instanziierte Sicherheitsanforderungen gespeichert werden können. Dabei muss das Verfahren auch eine schlüssige Verwaltung der Sicherheitsanforderungen umfassen, um die Sicherheitsanforderungen wieder einfach auffinden zu können. Naheliegend ist dabei die Speicherung der Sicherheitsanforderungen unter Bezugnahme auf das System, ggf. das Teilsystem und ggf. die Systemkomponenten, für welche die Sicherheitsanforderung gilt.

Der Benutzer erhält klare und einfache Anleitungen für die Anwendung des Verfahrens, wenn die Selektion und Instanziierung durch eine dialogbasierte interaktive Führung gesteuert wird. Der Benutzer muss dann nur den Anweisungen der dialogbasierten Führung folgen. Dadurch ist das Verfahren einfach erlernbar. Solch ein Unterstützungssystem ist ein Beratungssystem (vgl. Abschnitt 5.2.1). Dabei ist es sinnvoll die Anwendung des Werkzeugs durch ein Hilfesystem zu unterstützen.

5.4 Zusammenfassung des Kapitels

In diesem Kapitel wurde zunächst gemäß dem aktuellen Stand der Wissenschaft Temporallogik als geeignetes Beschreibungsmittel zur formalen Spezifikation von Sicherheitsanforderungen herausgestellt. Anschließend wurde der Safety-Pattern-Ansatz als geeigneter Verfahrens-Ansatz zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme eingeführt. Er basiert auf Techniken, die aus der muster- und komponentenbasierten Softwareentwicklung bekannt sind. Das Grundprinzip ist in Abbildung 5.4 dargestellt.

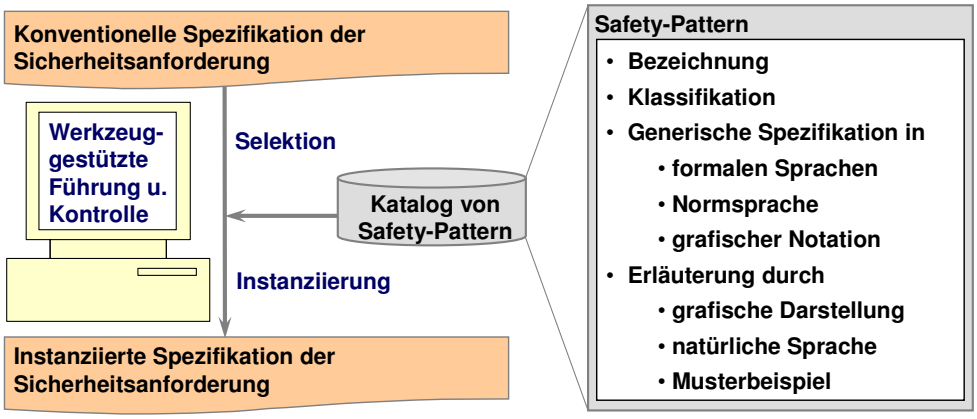


Abbildung 5.4: Grundprinzip des Safety-Pattern-Konzepts

Es werden vorspezifizierte generische Spezifikationen verwendet, die zur Spezifikation lediglich instanziiert werden müssen. Für komplexe Sicherheitsanforderungen ist dabei gegebenenfalls eine Komposition erforderlich. Durch eine systematische Katalogisierung gemäß der Eigenschaften der generischen Spezifikationen wird eine einfache Selektion der geeigneten Safety-Pattern erreicht. Der Safety-Pattern-Katalog muss so entwickelt werden, dass er durch die Verwendung einzelner generischer Formulierungen sowie durch deren Komposition eine vollständige Grundlage zur Spezifikation aller Arten funktionaler Sicherheitsanforderungen für Automatisierungssysteme bietet. Darüber hinaus muss der Katalog weitere generische Formulierungen beinhalten, durch welche die Handhabbarkeit des Verfahrens gewährleistet wird. Der Safety-Pattern-Ansatz muss zudem bei der Dokumentation von Sicherheitsanforderungen mithilfe generischer Spezifikationen anleiten. Dazu müssen eine Safety-Pattern-Normsprache und eine geeignete grafische Notation entwickelt werden. Durch Erläuterungen mithilfe der natürlichen Sprache, grafischer Notation und Musterbeispielen müssen umfassende Informationen zur Bedeutung und Anwendung jedes Safety-Pattern gegeben werden. Eine Werkzeugunterstützung für die Selektion und Instanziierung von Safety-Pattern ist ein wesentlicher Bestandteil des Safety-Pattern-Konzepts, um die Konzept-Anforderungen hinsichtlich der Anwendbarkeit (effiziente Anwendbarkeit, einfache Erlernbarkeit und Anwendbarkeit, hoher Automatisierungsgrad und Mehrfachverwendung) an ein geeignetes Verfahren erfüllen zu können.

6 Entwicklung des Safety-Pattern-Konzepts

Der in Kapitel 5 hergeleitete Ansatz zur anwendbaren formalen Spezifikation von Sicherheitsanforderungen in Temporallogik wird in diesem Kapitel umgesetzt. Das Ergebnis ist das Safety-Pattern-Konzept. Die Safety-Pattern-Menge wird in der Weise hergeleitet, dass die Menge die vollständige Basis für die anwendbare formale Spezifikation von Sicherheitsanforderungen darstellt. Zunächst wird gemäß des bereits eingeführten Vorgehens die Menge der notwendigen generischen Formulierungen hergeleitet, die unerlässlich ist, um alle für Automatisierungssysteme erforderlichen Sicherheitsanforderungen durch Komposition dieser Formulierungen formalisieren zu können. Zudem wird die Menge der nützlichen generischen Formulierungen aufgestellt, durch welche die anwendbare Spezifikation von Sicherheitsanforderungen in Temporallogik gewährleitstet wird. Auf der Basis der resultierenden Safety-Pattern-Menge und -Klassifikation wird die Safety-Pattern-Normsprache definiert. Entsprechend wird eine grafische Notation zur Spezifikation von Sicherheitsanforderungen entwickelt. Die unterschiedlichen adäquaten Erläuterungsarten für Safety-Pattern werden eingeführt. Geeignete Techniken für die Erläuterung durch Grafiken werden konzipiert.

6.1 Bestimmung der vollständigen Safety-Pattern-Menge

6.1.1 Vorgehen zur Ermittlung der Safety-Pattern-Menge

Das Vorgehen zur Ermittlung der Safety-Pattern-Menge wurde bereits in Abschnitt 5.3.3 eingeführt und erläutert. Zusammenfassend sind folgende Schritte erforderlich:

1. Ermittlung aller sicherheitsrelevanten atomaren Ausdrücke durch:

    a. Bestimmung, wodurch grundlegende atomare Formulierungen in Temporallogik gekennzeichnet sind.

    b. Bestimmung, wodurch sicherheitsrelevante Formulierungen charakterisiert sind.

    Daraus resultiert die *Menge der notwendigen generischen Formulierungen.*

2. Ermittlung von Ausdrücken, die aus der Menge der Notwendigen ableitbar sind und die Anwendbarkeit der Safety-Pattern-Menge erhöhen. Das Resultat ist die *Menge der nützlichen generischen Formulierungen*.

Schritt 1 wird im folgenden Abschnitt 6.1.2, Schritt 2 und die insgesamt resultierende Safety-Pattern-Menge in den darauf folgenden Abschnitten 6.1.3 und 6.1.4 behandelt.

6.1.2 Bestimmung der Menge der notwendigen Formulierungen für das Safety-Pattern-Konzept

6.1.2.1 Was sind grundlegende atomare Formulierungen?

Zur Bestimmung, wodurch grundlegende atomare Formulierungen gekennzeichnet sind, müssen die Ausdrucksmöglichkeiten von Temporallogik mit dem Ziel untersucht werden, die atomaren Ausdrücke zu bestimmen, auf deren Grundlage alle anderen Ausdrücke zur Spezifikation von Sicherheitsanforderungen durch Komposition gebildet werden können [ABG04]. Welche temporallogischen Sprachen dafür hinreichend sind, muss zuvor analysiert werden.

In den Abschnitten 2.4.1.3 und 2.4.2.2 wurden bereits die Computation Tree Logic (CTL) und die Linear-time Temporal Logic (LTL) als die temporallogischen Sprachen beschrieben, die sich in der Praxis der formalen Verifikation durch Model-Checking durchgesetzt haben. Dies geht aus der Betrachtung hervor, welche temporallogischen Sprachen beim Model-Checking Anwendung finden. Für eine prototypische Realisierung des Safety-Pattern-Konzepts ist folglich die Spezifikation der generischen Formulierungen der Safety-Pattern in LTL und CTL wichtig, um die erforderlichen Model-Checker zu unterstützen. Um als Ausnahme auch den Model-Checker Mucke direkt zu unterstützen (vgl. Abschnitt 2.4.2.2), ist zudem auch die Spezifikation im μ-Kalkül sinnvoll. Hierfür erfolgte in Kooperation mit dem Institut für Theoretische Informatik, TU Dresden, eine Umsetzung in das μ-Kalkül [Tche03].

Für die Bestimmung der Menge der notwendigen generischen Formulierungen des Safety-Pattern-Konzepts reicht vom Gesichtspunkt der Praxisrelevanz her gesehen die Analyse der Ausdrucksmöglichkeiten von LTL und CTL aus. Um jedoch auszuschließen, dass darüber hinaus in anderen temporallogischen Sprachen funktionale Sicherheitsanforderungen für Automatisierungssysteme ausgedrückt werden können, für die es in CTL und LTL keine geeigneten Ausdrucksmöglichkeiten gibt, werden im Folgenden die unterschiedlichen Arten von temporallogischen Sprachen aufgelistet. Auf dieser Grundlage wird diskutiert, welche Sprachen für die Erfüllung der Konzept-Anforderung bezüglich vollständiger Spezifizierbarkeit /KA3/ bei der Entwicklung der Safety-Pattern-Menge Berücksichtigung finden müssen.

Bewertung unterschiedlicher temporallogischer Sprachen

In diesem Abschnitt werden die verschiedenen Arten von temporallogischen Sprachen aufgelistet und bewertet. Generell können folgende temporallogische Sprachen unterschieden werden:

1.) *Lineare-Zeit Temporallogik (z. B. LTL):* Dies sind solche Temporallogik-Arten, mit denen nur Aussagen über einer linearen Zeitachse gemacht werden.

2.) *Verzweigende Temporallogik (z. B. CTL):* Im Unterschied zu 1) erlaubt diese Art von Logik Aussagen über unterschiedliche mögliche zeitliche Verläufe. Dies ist z. B. für Erreich-

barkeitsaussagen relevant: „Von bestimmten Zeitpunkten aus, muss das Erreichen eines bestimmten Systemzustands möglich sein."

3.) *Zeitbehaftete Temporallogiken (z. B. CCTL [RuKr97]):* Bei diesen Logiken können Aussagen mit expliziten Zeitbedingungen getroffen werden.

4.) *Stochastische Temporallogiken (z. B. PCTL [KNP01] [HaJo94] und CSL [ASSB00]):* Bei diesen Logiken können logische Relationen mit stochastischen Bedingungen versehen werden.

5.) *Intervallbasierte Temporallogiken (z. B. ITL [CMZ02]):* Während 1) und 2) zeitpunkt-basierte Logiken sind, werden hier Aussagen über Intervalle gemacht.

6.) *Vergangenheits-Temporallogiken (z. B. PTL [RBH+01, S. 664]):* Mit diesen Logiken sind nicht nur Gegenwarts- und Zukunftsaussagen möglich, sondern auch Vergangenheits-aussagen.

7.) *Temporal Logic for Actions (TLA) [Lamp94]:* In dieser Logik wird explizit zwischen Zuständen und Aktionen unterschieden. TLA bringt Vorteile für das Theorem-Proving mit sich, da temporallogische Verifikationsbedingungen auf prädikatenlogische Beweisver-pflichtungen reduziert werden können [WiMe01].

8.) *μ-Kalkül*[7] *[Tche03] [Canv99]:* Der μ-Kalkül ist eine Logik, die Fixpunkte verwendet. Diese Logik ist ausdrucksmächtiger als 1) und 2) [Tche03].

Bewertung von 1) und 2):

Für 1) hat sich die Notation von LTL und für 2) die CTL-Notation etabliert. Wie am Anfang dieses Abschnitts gezeigt wurde, sind diese beiden Temporallogiken die praxisrelevanten Temporallogiken. So werden CTL und LTL auch in [EsSc03] als die temporallogischen Sprachen beschrieben, die sich in der Praxis durchgesetzt haben.

Bewertung von 3) und 4) – zeitbehaftete und stochastische Temporallogiken:

Zeitbehaftete temporallogische Sprachen erlauben es, Zeitbedingungen separat von den anderen Bedingungen anzugeben [Béra01, S. 68]. Zeitbedingungen können jedoch auch einfach in der-selben Weise wie alle anderen Bedingungen in CTL und LTL spezifiziert werden, vgl. [Bits03a], [IEC61508]. Entsprechendes gilt auch für stochastische Temporallogiken, bei denen stochastische Bedingungen separat spezifiziert werden. Stochastische Temporallogiken sind zudem irrelevant, da in funktionalen Sicherheitsanforderungen keine stochastischen Angaben gemacht werden. Dies ist stattdessen in den Angaben für das Versagen der Sicherheitsfunktion durch physikalische Fehler erforderlich (vgl. Abschnitt 2.2.3).

---

[7] Neben den aufgeführten Temporallogik-Arten und -Sprachen gibt es noch CTL*. Da jedoch alle CTL*-Ausdrücke im μ-Kalkül abgebildet werden können [Reff00], erfolgt keine zusätzliche Diskussion.

Bewertung von 5) – intervallbasierte Temporallogiken:

Nach [Bels97, S.71] und [Pere95, S.72] lassen sich alle Intervallrelationen auf zeitpunktbasierte Darstellungen, wie sie in gewöhnlicher Temporallogik ausgedrückt werden können, zurückführen. Intervallrelationen besitzen lediglich eine andere Sichtweise. Diese Art von Temporallogik ermöglicht im Vergleich zu anderen ein eleganteres Spezifizieren von Intervallen. Somit lassen sich also in intervallbasierten Temporallogiken keine weiteren Sicherheitsanforderungen ausdrücken. Außerdem gibt es keine Model-Checker für diese Art von Temporallogik, sodass eine ingenieurgerechte formale Verifikation ausgeschlossen werden muss.

Bewertung von 6) – Vergangenheits-Temporallogiken:

Nach [Beck92] können alle Vergangenheitsaussagen auch einfach aus einer Zukunftsperspektive ausgedrückt werden, sodass sich keine weiteren Sicherheitsanforderungen ausdrücken lassen. Zudem ist das Ausdrücken von Sicherheitsanforderungen aus der Gegenwarts- und Zukunftsperspektive der natürliche Weg. Anforderungen, deren Gültigkeit in allen möglichen Systemzuständen überprüft werden soll, werden aus Sicht des Betriebs eines Automatisierungssystems vor der Inbetriebnahme ausgedrückt.

Ein Beispiel für einen Ausdruck mit Vergangenheitsaussage ist: „`b` darf nur eintreten, wenn es davor einen Zeitpunkt gab, zu dem `a` gültig war.“ Äquivalent hierzu ist der folgende Ausdruck mit Zukunftsaussage: „Erst ab dem Zeitpunkt, zu dem `a` gilt, darf `b` eintreten.“

Es ist zudem kein Model-Checking-Verfahren bekannt, das auf dieser Art von Temporallogik basiert. Deshalb muss eine ingenieurgerechte formale Verifikation auf der Grundlage von Anforderungen in Vergangenheits-Temporallogik ausgeschlossen werden.

Bewertung von 7) – Temporal Logic for Actions (TLA):

TLA zeichnet sich durch einen kleinen Satz logischer Operatoren aus [WiMe01]. Dadurch können allerdings manche Ausdrücke, die in CTL und LTL gebildet werden können, so in TLA nicht ausgedrückt werden. TLA unterscheidet explizit zwischen Zuständen und Aktionen. Auch in CTL und LTL können über Aktionen Aussagen getroffen werden. Dies wird dort lediglich durch die Notation nicht besonders kenntlich gemacht. Somit lassen sich mit TLA keine weiteren Sicherheitsanforderungen ausdrücken.

Bewertung von 8) – μ-Kalkül:

In [Tche03] wurde das μ-Kalkül auf die Konstrukte eingeschränkt, die von praktischer Relevanz sind. Somit wurde hier bereits die Aussage getroffen, dass die große Ausdrucksmächtigkeit des μ-Kalküls für die Praxis nicht relevant ist. Es wurde verglichen, ob mit diesem eingeschränkten μ-Kalkül Eigenschaften spezifizierbar sind, die nicht mit CTL ausgedrückt werden können. In [Tche03] wurde unter Verweis auf [Kaiv97] folgendes erläutert: „*«At every even moment» […] can not be expressed in terms of CTL operators*“. Das bedeutet, eine zyklische Eigenschaft wie

z. B., dass alle drei Zustandsübergänge eine bestimmte Aussage gelten soll, kann mit CTL nicht formuliert werden. Dasselbe gilt für LTL.

Dies ist jedoch für die Anforderungs-Spezifikation für Automatisierungssysteme auch nicht von praktischem Nutzen. Zyklische Eigenschaften über eine bestimmte Aussage, die in gewissem Zeitabstand gültig ist, muss es natürlich geben können. Das kann in CTL jedoch in folgender Weise ausgedrückt werden: `AG (Zykluszeit_abgelaufen → Aussage)`. Die Art von Eigenschaft, die in CTL nicht ausgedrückt werden kann, betrifft eine Ausdrucksform für eine modellierungstechnische Eigenschaft, d. h. die Art und Weise der Modellierung, nicht jedoch eine Systemeigenschaft. Eine Sicherheitsanforderung ist eine Anforderung auf Systemebene. Es ist eine Anforderung, die von einem Automatisierungssystem erfüllt werden muss. Eine zyklische Eigenschaft in einem Automatisierungssystem ist an eine Systemgröße gebunden, die zyklisch ist, wie z. B. ein Taktgeber oder die Zeit. Während im μ-Kalkül z. B. ausgedrückt werden kann, dass nach jedem fünften Zustand eine bestimmte Eigenschaft gelten muss, würde in CTL eine Bedingung im Kontext eines Taktgebers oder einer anderen Systemgröße angegeben.

Folglich ist diese Art von Eigenschaft, die im μ-Kalkül, aber nicht in CTL ausdrückbar ist, von keiner praktischen Bedeutung für das Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme. Es geht dabei lediglich um die Art und Weise, wie eine Systemeigenschaft logisch ausgedrückt werden kann.

Zusammenfassende Bewertung:

Entscheidend ist die Betrachtung von LTL und CTL. Andere Arten von Temporallogik erlauben lediglich eleganteres Spezifizieren für bestimmte Spezifikationsfälle (z. B. Aussagen mit expliziter Zeit sowie Aussagen mit Intervallen) sowie andere Betrachtungsformen, wie z. B. Vergangenheitsaussagen. Zudem lassen sich durch bestimmte temporallogische Sprachen noch zusätzliche Aussagen über Art und Weise der Modellierung bilden. Sie ermöglichen jedoch keine Spezifikation weiterer zusätzlicher Systemeigenschaften für Automatisierungssysteme.

Aufgrund der erläuterten Verbreitung und der Bedeutung von LTL und CTL zur Anforderungsspezifikation für das Model-Checking müssen alle sicherheitsrelevanten Ausdrücke, die mit diesen beiden Sprachen spezifiziert werden können, mit dem Verfahren zur Spezifikation funktionaler Sicherheitsanforderungen formulierbar sein. Deshalb müssen LTL und CTL zur Ermittlung der notwendigen Menge der generischen Formulierungen für das Safety-Pattern-Konzept herangezogen werden. Alle Sicherheitsanforderungen, die in LTL oder CTL spezifizierbar sind, müssen auch mit Safety-Pattern spezifizierbar sein.

Ermittlung der grundlegenden atomaren Ausdrucksmöglichkeiten in CTL und LTL

CTL und LTL besitzen eine endliche Menge temporallogischer Operatoren (z. B. LTL: `G`, `F`, `X`, `U` und `W`), mit denen sich jedoch unendlich viele logische Formulierungen erstellen lassen, da die

Länge eines Ausdrucks (Satzes) unbeschränkt ist. Deshalb kann die Menge der Safety-Pattern nicht alle formulierbaren Sätze enthalten, da sonst die Menge unendlich groß wäre. Stattdessen enthält die Safety-Pattern-Menge eine endliche Menge von Formulierungen, durch deren Kombinationen alle möglichen temporallogischen Ausdrücke gebildet werden können.

Die kleinste Menge an Formulierungen umfasst alle atomaren Ausdrucksmöglichkeiten, die sich in langen temporallogischen Formulierungen wiederholen. Nicht-atomare Formulierungen müssen auf der Grundlage dieser atomaren Ausdrücke durch einfache Kompositionen gebildet werden können. Es gibt in LTL z. B. folgenden atomaren Ausdruck mit `W`:

`p W q` „`p` gilt ununterbrochen bis zu dem Zeitpunkt, zu dem `q` gültig ist."

Statt `q` kann irgendein anderer Ausdruck eingesetzt werden, z. B. `G a`:

`p W G a` „`p` gilt ununterbrochen bis zu dem Zeitpunkt, zu dem gilt: In allen folgenden Zuständen gilt immer `a`."

Dabei enthält die Menge der „atomaren Ausdrücke" nur grundlegende Formulierungen (die aus den grundlegenden Kombinationsmöglichkeiten der temporallogischen Operatoren[8]), die sich in langen Formulierungen wiederholen. Im Folgenden wird weiter präzisiert, welche Formulierungen als „atomar" anzusehen sind.

Für die notwendige Menge generischer Formulierungen sind nur die atomaren Ausdrucksmöglichkeiten relevant, die eine Bedeutung besitzen, welche bei der Bildung der Menge noch nicht aufgetreten ist. Im Folgenden wird erläutert, dass es verschiedene Formulierungen geben kann, die dieselbe Bedeutung besitzen.

Gemäß dem linguistischen Schichtenmodell nach [Schn99, S.130] [Lako70] sind die folgenden Schichten zu unterscheiden:

1. Signal- / Lautebene
2. Wortebene
3. Satzebene
4. Bedeutungsebene

Für LTL ist die kleinste Einheit die Wortebene. Die möglichen temporallogischen Operatoren (`G`, `F`, `X`…) in LTL bilden für sich bereits ein Wort. In dieser Ebene geht es um Wortbedeutungen im System einer Einzelsprache. Für LTL ist deshalb die Signal- / Lautebene bedeutungslos. Im Fall von CTL sind die einzelnen Pfad- und Zeitquantoren auf der ersten Ebene, während erst die möglichen Symbolpaare auf der Wortebene sind (vgl. Abschnitt 2.4.1.3).

---

[8] Alle Formulierungen mit `U` lassen sich auf solche mit `W` zurückführen [Béra01, S. 31], sodass für die Praxis die Verwendung eines der Operatoren ausreicht (im ursprünglichen CTL gibt es nur einen „Until-Operator" [HuRy00, S. 152]). Deshalb wurde in der Analyse nur ein Operator der beiden betrachtet.

Die verschiedenen nach der temporallogischen Grammatik erlaubten Kombinationen der Operatoren ergeben unterschiedliche Sätze. In LTL sind dies somit alle möglichen Permutationen der LTL-Operatoren, welche die Grammatik der Sprache erlaubt. Dies ist die Sichtweise auf der Satzebene.

Darüber steht jedoch die Bedeutungsebene. Auf der Bedeutungsebene können unterschiedliche Ausdrücke der Satzebene für denselben Zweck (mit derselben Intention) verwendet werden. Es geht hier darum, was eine Information für den Empfänger bedeutet. Die Empfänger sind in unserem Fall Entwickler, die ein Automatisierungssystem modellieren und realisieren. Auf der Satzebene gibt es Unterscheidungsmöglichkeiten, die praktisch aus Sicht der Automatisierungstechnik gesehen, keine Rolle spielen. Für diese Unterscheidungsmöglichkeiten spezielle Formulierungen abzuleiten, ist nicht notwendig, würde aber die Zahl der Safety-Pattern unnötig hoch werden lassen. Unterschiedliche Ausdrücke in der Satzebene, d. h. unterschiedlich formulierte Sätze, können in der Bedeutungsebene äquivalent sein. Im Deutschen kann man z. B. folgende unterschiedlichen Sätze mit derselben Bedeutung bilden:

- „Das Eisenbahnfahrzeug darf im automatischen Betrieb niemals den Bahnübergang befahren, wenn er geöffnet ist.“
- „Im automatischen Betrieb darf das Eisenbahnfahrzeug niemals den Bahnübergang befahren, wenn er geöffnet ist.“
- „Niemals darf im automatischen Betrieb das Eisenbahnfahrzeug den Bahnübergang befahren, wenn er geöffnet ist.“
- „Wenn der Bahnübergang geöffnet ist, darf er im automatischen Betrieb niemals vom Eisenbahnfahrzeug befahren werden.“
- „Im automatischen Betrieb darf der geöffnete Bahnübergang niemals vom Eisenbahnfahrzeug befahren werden.“

Ein möglicher LTL-Ausdruck ist z. B. (6.1). Ein aus der Sicht der Bedeutungsebene gleichwertiger LTL-Ausdruck ist (6.2).

`G p (6.1)`

Erläuterung von (6.1) in der Satzebene: `p` muss in allen Systemzuständen gelten (vgl. Abbildung 6.1).

`GG p (6.2)`

Erläuterung von (6.2) in der Satzebene: Von jedem Systemzustand aus muss `p` in allen Systemzuständen gültig sein (vgl. Abbildung 6.2).

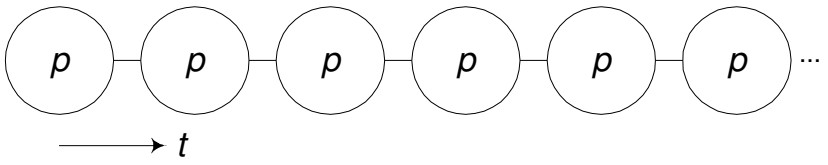


Abbildung 6.1: Veranschaulichung von (6.1) „p muss in allen Systemzuständen gelten“

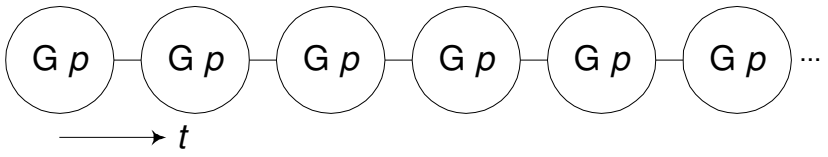


Abbildung 6.2: Veranschaulichung von (6.2) „von jedem Systemzustand aus muss p in allen Systemzuständen gültig sein“

Löst man alle „G p“ in Abbildung 6.2 nach ihrer Bedeutung auf, resultiert die Darstellung in Abbildung 6.3. Mit p & p = p folgt dieselbe Bedeutung wie Abbildung 6.1.

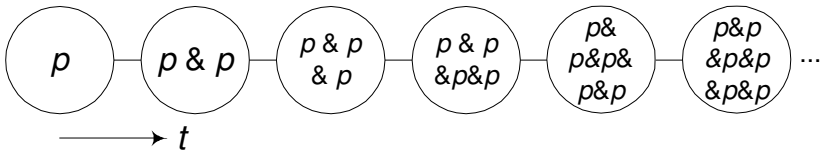


Abbildung 6.3: Veranschaulichung von (6.2) „von jedem Systemzustand aus muss p in allen Systemzuständen gültig sein“ mit Auflösung von „G p“

Das zweite G ist also ohne praktischen Nutzen in (6.2). Es führt nur zu einer Wiederholung der Forderung des ersten G. Die Semantik von (6.2) in der Bedeutungsebene und aus der Sicht der Automatisierungstechnik ist somit: „p muss immer gelten“.

Alle möglichen Permutationen von temporalen Operatoren in LTL und von Konnektorpaaren in CTL (vgl. Abschnitt 2.4.1.3) unterscheiden sich also auf der Satzebene. Auf der Bedeutungsebene werden hingegen nur die Permutationen mit verschiedenen Bedeutungen unterschieden. Die Ausdrücke der Menge der notwendigen generischen Formulierungen müssen sich auf der Bedeutungsebene unterscheiden. Die Unterschiede auf der Bedeutungsebene sind relevant, damit der Systementwickler die erforderliche Sicherheitsanforderung formulieren kann. Wie die Anforderung letztlich formuliert wird, ist dabei irrelevant.[9]

Bei der Analyse von LTL und CTL geht es um die Differenzierung der verschiedenen Permutationen aus den temporallogischen Operatoren auf der Bedeutungsebene. Es wird untersucht, was die verschiedenen Permutationen der temporalen Operatoren (z. B. LTL: G, F, … GF, GW, FG, FW, …, GFXW, …, X^n, …) bedeuten. Da es bei den Permutationsmöglichkeiten

[9] Zu demselben Ergebnis kommt man auch durch in [Stei97] differenzierten Ebenen der Semantik: strukturelle Semantik, generative Semantik und praktische Semantik.

Wiederholungen von Bedeutungen gibt, spielen bei der Untersuchung der Ausdrucksmöglichkeiten jedoch nur die Formulierungen eine Rolle, die sich im Vergleich zu den bisher bestimmten Ausdrücken für die Bildung von Safety-Pattern, auf der Bedeutungsebene unterscheiden. Diese Ausdrücke sind original[10].

Neben originalen Ausdrucksmöglichkeiten werden die atomaren Formulierungsmöglichkeiten durch die Einführung von Parametern abgeleitet. In LTL sind die Permutationen von `X` („im unmittelbar nächsten Zustand“) und nur mit `X` nie original. Durch Einführung eines Parameters n lässt sich jedoch ein generischer Ausdruck ableiten. Zum Beispiel sind die Formulierungen `G (a → X b)`, ..., `G(a → XXXXXX … X b)` durch `G (a →` X^n `b)` generalisierbar.

Für die Menge der notwendigen generischen Formulierungen sind zudem nur die formalen Ausdrucksmöglichkeiten entscheidend, die für die Spezifikation sicherheitsrelevanter Anforderungen relevant sind. Betrachtet man in LTL die sicherheitsrelevanten Permutationen aus `G` und `F` sowie in CTL die sicherheitsrelevanten Permutationen aus `AG`, `AF`, `EG` und `EF` nach dem in diesem Abschnitt beschriebenen Prinzip auf der Bedeutungsebene, lässt sich feststellen, dass sich diese auf einzelne temporale Operatoren (vgl. Formulierung 6.2) oder auf 2er-Permutationen zurückführen lassen. Alle Permutationen von X^n mit den anderen temporalen Operatoren lassen sich auf 3er-Permutationen zurückführen. Lediglich bei Instanziierungen mit `W` lassen sich längere Formulierungen nicht generell auf kleine Permutationen zurückführen. In diesem Fall werden deshalb in der Analyse nur die Grundkombinationen betrachtet und in die Menge der notwendigen Formulierungen aufgenommen. Längere Formulierungen können durch Komposition, d. h. durch die Verwendung aussagenlogischer Verknüpfungen oder durch das Einsetzen von Formulierungen in andere, gebildet werden (vgl. Abschnitt 6.1.3).

6.1.2.2 Was sind sicherheitsrelevante Formulierungen?

Nur genau die Ausdrucksmöglichkeiten in CTL und LTL sind relevant, mit denen sich Sicherheitsanforderungen im Sinne von *safety* ausdrücken lassen, vgl. [Bits01], [ABG04]. In Korrektheitsprüfungen von Sicherheitsanforderungen handelt es sich immer um funktionale Aspekte der Sicherheit. Bei funktionaler Sicherheit geht es um den Schutz vor Gefährdungen, die durch inkorrekte Funktionen entstehen (s. Abschnitt 2.2.3). In der Formulierung der Sicherheitsanforderungen müssen deshalb stets Formulierungen mit deterministischen Forderungen verwendet werden, sonst kann die Verhinderung gefährlicher Systemsituationen bzw. die Verhinderung von sicherheitskritischen Folgen und damit Sicherheit nicht gewährleistet werden.

Ein Beispiel für solch eine nicht-sicherheitsrelevante Formulierung ist der folgende Ausdruck (in LTL):

`F (p → F q)  (6.3)`

[10] „Orginal“ bedeutet nach [Dose90] „von besonderer, einmaliger Art“.

Bedeutung von (6.3): Es muss irgendwann mindestens einmal ein Systemzustand erreicht werden, zu dem gilt: Wenn `p` erfüllt ist, dann muss von dem Zeitpunkt an irgendwann `q` erreicht werden.

Die Forderung der Formulierung ist für sicherheitsrelevante Aspekte nicht verwendbar: Diese Forderung, dass *wenn* `p` *erfüllt ist, dass dann* `q` *irgendwann erreicht werden muss*, gilt nicht in allen, sondern nur in manchen Systemzuständen. Es wird nicht kontrolliert, wann die Forderungen gelten sollen (fehlende Bedingung für die Geltung der Forderung). Zum Beispiel sind die drei Fälle, die in Abbildung 6.4, Abbildung 6.5 und Abbildung 6.6 beschrieben sind, möglich[11]. Wann die Eigenschaft für das Erreichen von `q` gilt, ist absolut willkürlich.

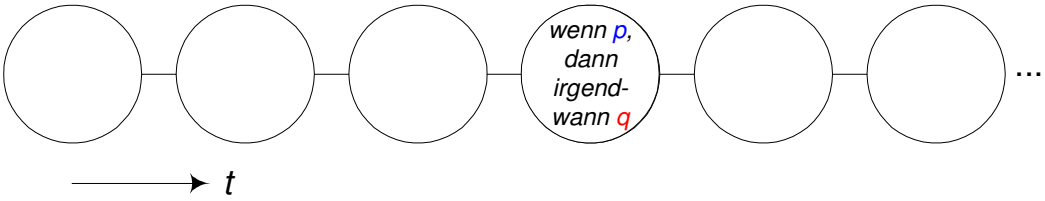


Abbildung 6.4: Erste mögliche Erfüllung von (6.3)

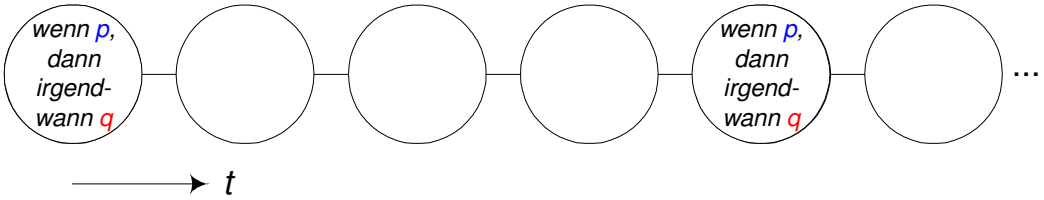


Abbildung 6.5: Zweite mögliche Erfüllung von (6.3)

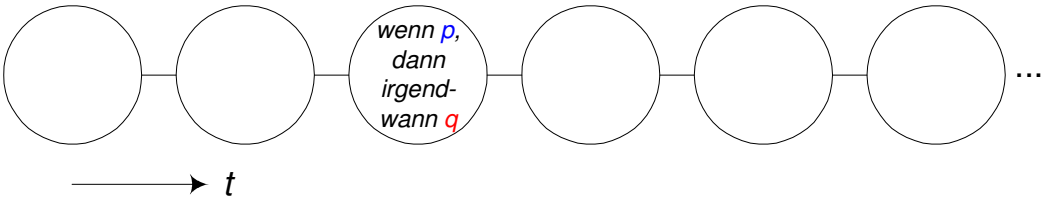


Abbildung 6.6: Dritte mögliche Erfüllung von (6.3)

Dieser Nichtdeterminismus der Forderung kann aus sicherheitstechnischer Sicht niemals geduldet werden. Es ist gefährlich, einem Systementwickler ohne besondere Kenntnisse in höherer Logik zu erlauben, solche Logik-Ausdrücke für Sicherheitsanforderungen zu verwenden, da die Geltung der Eigenschaft nicht für eine deterministische Bedingung gilt. Dies ist wiederum eine

[11] Diese Abbildungen sind dabei so zu interpretieren, dass auf einer Zeitachse Zustände aufgelistet sind, die ein beliebiges Automatisierungssystem nacheinander einnimmt. Die Zuständen, in denen eine bestimmte Relation zwischen `p` und `q` gilt, sind beschriftet.

Motivation für das Safety-Pattern-Konzept, in dem nur solche Formulierungen bereitgestellt werden und verwendbar sind, die zur Spezifikation von Sicherheitsanforderungen geeignet sind.

Im Falle von Sicherheit müssen Forderungen gemacht werden, in denen willkürliches Verhalten für die sicherheitsrelevante Eigenschaft ausgeschlossen ist. Sicherheitsanforderungen beschreiben somit Forderungen, deren Geltung deterministisch ist, d. h. zu bestimmten und nicht nur zu manchen Zeitpunkten muss die Forderung gelten. In folgenden Fällen ist dies nicht gegeben:

- *Erreichen oder Erreichbarkeit von Bedingungen:* Bei Sicherheitsanforderungen muss eindeutig bestimmt sein, wann die Bedingung für die Gültigkeit einer sicherheitsrelevanten Eigenschaft erfüllt ist. Dabei ist die Bedingung die Voraussetzung für die Erfüllung einer für die Sicherheit erforderlichen Eigenschaft in bestimmten Systemzuständen. Wenn wie im vorgestellten Beispiel (6.3) eine Bedingung zu unbestimmten Zeitpunkten erreicht werden soll, liegt ein Nichtdeterminismus vor. Dasselbe ist der Fall, wenn die Erreichbarkeit von Bedingungen ausgedrückt wird.

- *Erreichen von Eigenschaften ohne Spezifizierbarkeit einer deterministischen Bedingung:* Wenn eine deterministische Bedingung spezifizierbar ist, darf bei der Formulierung von Sicherheitsanforderungen auch das Erreichen einer Eigenschaft ausgedrückt werden. Ein Beispiel hierfür ist folgende Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb (vgl. [Bits03a]):

  <u>Formulierung in LTL:</u>

  ```
  G((train.receiving_message = level_crossing_not_protected)
    → F(train.velocity=0 and
         train.location < level_crossing.location_beginning))
  ```

  <u>Bedeutung:</u>
  Wenn der Zug die Nachricht erhält, dass der Bahnübergang nicht gesichert ist, so muss danach der Zustand erreicht werden, in dem die Geschwindigkeit des Zuges Null ist, während sich das Eisenbahnfahrzeug noch vor dem Ort des Bahnübergangs befindet.[12]

  Dabei wird ein temporallogisches Konstrukt verwendet, das das Erreichen einer Eigenschaft zu einem beliebigen Zeitpunkt erlaubt. Allerdings ist die Eigenschaft an eine physikalische Systemgröße gebunden. Dadurch wird genau bestimmt, wann diese Sicherheitseigenschaft gültig ist nämlich, bevor der Ort des Bahnübergangs erreicht wird.

  Beim Erreichen von Erreichbarkeitseigenschaften[13] zu undefinierten Zeitpunkten ist jedoch z. B. keine deterministische Bedingung spezifizierbar. Generell gibt es auch Sicherheits-

[12] Die Formulierung ist nur unter der Annahme für die Sicherheit hinreichend, dass der Zug den Bahnübergang nur einmal passieren kann.
[13] Mit dem Begriff „Erreichbarkeitseigenschaft“ wird eine Eigenschaft bezeichnet, bei der entweder generell oder ausgehend von einer Bedingung die Geltung einer Eigenschaft erreicht wird.

anforderungen, die eine Aussage über die Erreichbarkeit einer bestimmten Eigenschaft machen, die notwendig für die Gewährleistung der Sicherheit ist (z. B. Aktivierbarkeit eines Nothalts). Erreichbarkeitseigenschaften sind jedoch nur sicherheitsrelevant, wenn eindeutig festgelegt wird, in welchen Zuständen sie gelten. Wenn eine sicherheitsrelevante Erreichbarkeitseigenschaft nur in einer nicht genau bestimmten Auswahl von Zuständen gelten soll, so ist dieser Nichtdeterminismus ungeeignet für die Formulierung von Sicherheitsanforderungen. Anforderungen, in denen eine Erreichbarkeitseigenschaft in nicht exakt definierten Systemzuständen gilt, können nicht für die Gewährleistung der Sicherheit eines Systems verwendet werden. Dies ist in Formulierungen der Fall, in denen das Erreichen von Erreichbarkeitseigenschaften zu unbestimmten Zeitpunkten ausgedrückt wird. Erreichbarkeitseigenschaften werden nur dann in Sicherheitsanforderungen verwendet, wenn sie unter konkreten Systembedingungen gelten sollen.

- *Erreichbarkeit von Erreichbarkeit:* Wenn eine Erreichbarkeitseigenschaft lediglich erreichbar sein soll, jedoch nicht unter bestimmten Bedingungen gelten muss, ist dies nicht sicherheitsrelevant. Eine Erreichbarkeitseigenschaft, die nicht unbedingt gelten oder eintreten muss, besitzt einen Nichtdeterminismus, sodass solche Formulierungen nicht für Sicherheitsanforderungen verwendet werden.

Aus der Perspektive der Formulierung von Anforderungen für sichere Automatisierungssysteme ist es nicht sinnvoll, bestimmte Formulierungsmöglichkeiten zu differenzieren, die in Temporallogik unterschieden werden können. Bei sicherheitstechnischen Betrachtungen geht es um die Wirkung eines Systems. Die CTL-Formulierungen `AX EG p`, `EG AX p` und `EX AG p` haben dieselben Wirkungen: Es muss möglich sein, dass eine Eigenschaft `p` immer außer im Anfangszustand gilt (vgl. Ausführungen zur *praktischen Semantik / Bedeutungsebene* in Abschnitt 6.1.2.1).

Die nach den in den letzten beiden Abschnitten vorgestellten Regeln bestimmten Formulierungen sind die grundlegenden sicherheitsrelevanten temporallogischen Formulierungen. Diese Menge umfasst insgesamt 24 generische Formulierungen. Alle längeren Formulierungen für die Formulierung von Sicherheitsanforderungen sind auf der Grundlage dieser formulierbar. Die Schwierigkeit von Temporallogik liegt in der Verwendung der temporallogischen Operatoren begründet. Durch diese grundlegenden Formulierungen wird das selbstständige Kombinieren der einzelnen temporallogischen Operatoren vermieden. Längere Formulierungen werden durch die Verwendung aussagenlogischer Verknüpfungen oder durch das Einsetzen von Formulierungen in andere gebildet.

In den Abschnitten 6.1.2.1 und 6.1.2.2 wurde beschrieben, wie die Menge der notwendigen generischen Formulierungen des Safety-Pattern-Konzepts bestimmt wird. Die Menge der notwendigen generischen Formulierungen enthält bis auf eine Ausnahme (Formulierungen mit dem Zeitquantor `W`) alle für Sicherheitsanforderungen relevanten Kombinationen von LTL- bzw.

CTL-Konnektoren, die sich in der Bedeutungsebene unterscheiden. Für den Ausnahmefall sind jedoch die generischen Formulierungen enthalten, welche die Bildung komplexerer Formulierungen immer nach dem selben Kompositionsschema erlauben.

Auf der Grundlage dieser atomaren generischen Formulierungen und durch Kombination mit aussagenlogischen Konnektoren können komplexe Sicherheitsanforderungen spezifiziert werden. Da die Menge der notwendigen generischen Formulierungen die für die Spezifikation von Sicherheitsanforderungen atomaren Formulierungen enthält, stellt sie die vollständige Basis dar, um durch Komposition alle relevanten Formulierungen für Sicherheitsanforderungen abzuleiten.

Im folgenden Abschnitt geht es nun darum, Fehler bei der Anwendung und Komposition von Safety-Pattern durch die Bereitstellung zusätzlicher Safety-Pattern zu vermeiden.

6.1.3 Bestimmung der Menge der nützlichen generischen Formulierungen für die Anwendbarkeit des Safety-Pattern-Konzepts

Wie in [Bits03a] und [IEC61508] beschrieben, sind in formalen Anforderungsspezifikationssprachen auch Aussagen möglich, für deren Spezifikation die jeweilige Spezifikationssprache nicht in erster Linie entwickelt wurde (Beispiel: Spezifikation mit expliziter Zeitangabe in Temporallogik). Deshalb bieten die Spezifikationssprachen für solche Aussagen zwar keine besonderen sprachlichen Konstrukte, dennoch können diese Aussagen spezifiziert werden[14]. Da jedoch keine besonderen Sprachkonstrukte vorhanden sind, kann solch eine Spezifikation umständlich und mit Schwierigkeiten verbunden sein. Der Systementwickler muss Kenntnisse über die Anwendung der Spezifikationssprache in diesen besonderen Fällen besitzen. Da die Menge der notwendigen generischen Formulierungen auf der Grundlage der Analyse der Sprachmittel abgeleitet wurde, sind diese Fälle deshalb in dieser Menge nicht in Form spezieller generischer Formulierungen berücksichtigt.

Solche Fälle ergeben sich insbesondere durch bedeutende zeitliche Relationen sowie unterschiedliche Systemarten in der Automatisierungstechnik, die jeweils eine besondere Art der Spezifikation erfordern (Beispiele: Anforderungen an zeitgesteuerte Systeme oder Zeitanforderungen in Abhängigkeit zu einer Systemuhr). So sind aus der Automatisierungstechnik typische Unterscheidungsmerkmale zeitlicher Relationen bekannt (vgl. [LaGö99a, S. 227], [Beck92, S. 49, 51 ff.]), die sich nicht unbedingt in expliziten sprachlichen Konstrukten von Temporallogik widerspiegeln. Auch für solche Fälle sollte es spezifische Safety-Pattern geben, die somit dem Systementwickler die Spezifikation erleichtern. Zu viele verschiedene Arten von Safety-Pattern erschweren allerdings die Selektion der geeigneten Safety-Pattern für die zu spezifizierenden Sicherheitsanforderungen. Mit expliziten Safety-Pattern sollen jedoch zumin-

[14] Als Erläuterung hierzu, sei der Vergleich unterschiedlicher natürlicher Sprachen angeführt. Vielfach gibt es keine entsprechenden Worte in den verschiedenen Sprachen. Die Sprachen verfügen über unterschiedlich viele Kasus- und Verbalformen. Trotzdem ist es möglich, vollständig zu übersetzen. Die Sprachen unterscheiden sich jedoch darin, wie einfach etwas ausgedrückt werden kann [Lieb04, S. 124-125].

dest die schwierigen Spezifikationsfälle abgedeckt werden. Zudem soll die Instanziierung und Komposition für häufig auftretende Spezifikationsprobleme vereinfacht werden.

Um dem Systementwickler die Selektion der erforderlichen generischen Formulierungen aus der Safety-Pattern-Menge zu ermöglichen, muss die Menge durch eine Klassifikation geordnet werden [BiGö02]. Hierfür werden Klassifikationskriterien mit dazugehörigen Klassen bestimmt. Um eine in sich geschlossene Klassifikation zu erhalten, wird die Safety-Pattern-Menge dabei um weitere Spezifikationsmuster ergänzt, die eigentlich auch mithilfe anderer Safety-Pattern und/oder der Verwendung aussagenlogischer Konnektoren gebildet werden könnten. Zum Beispiel gibt es ein Safety-Pattern mit folgender generischer Formulierung:

1.) Generische Formulierung in LTL: `not p W q`

Bedeutung: `p` darf bis zu dem Zeitpunkt nicht gelten, zu dem `q` eintritt.

Dieselbe Formulierung erhält man auch, indem man in die folgende generische Formulierung `not p` für `a` einsetzt:

2.) Generische Formulierung in LTL: `a W q`

Bedeutung: `a` muss bis zu dem Zeitpunkt gelten, zu dem `q` eintritt.

Damit Entwickler bewusst zwischen Sicherheitsanforderungen mit eingeschränkt zulässigen Eigenschaften und solchen, die notwendig erforderliche Eigenschaften beschreiben, unterscheiden, ist es sinnvoll die beiden Safety-Pattern-Klassen *„Notwendige Geltung“* und *„Zulässige Geltung“* einzuführen (vgl. Einführung der Safety-Pattern-Klassen in Abschnitt 6.1.4). Zur Klasse *„Zulässige Geltung“* gehört dann auch ein Safety-Pattern mit der generischen Formulierung 1.), durch die ausgedrückt wird, dass die Geltung einer Eigenschaft bis zu einem bestimmten Zeitpunkt nicht gelten darf, und erst ab diesem Zeitpunkt *zulässig* ist.

Somit gibt es Safety-Pattern, die zwar nicht erforderlich sind, um die unterschiedlichen Sicherheitsanforderungen ausdrücken zu können, weil man die erforderlichen Formulierungen auch durch Komposition verschiedener Safety-Pattern sowie durch die Verwendung aussagenlogischer Konnektoren bei der Instanziierung bilden könnte. Dennoch sind diese Safety-Pattern wichtig, um den Systementwickler zum Nachdenken über Ausdrucksmöglichkeiten anzuhalten und anzuleiten und zur präzisen Spezifikation zu zwingen.

Eine Ursache für Spezifikationsfehler ist, dass Sachverhalte zu unpräzise durchdacht werden. Dann wird für die Beschreibung des Sachverhalts ein sprachliches Konstrukt verwendet, das einem gerade in den Sinn gekommen ist, ohne ausreichend geprüft zu haben, welche unterschiedlichen sprachlichen Konstrukte es im jeweiligen Kontext gibt, und was deren semantische Unterschiede im Detail sind.

Wenn z. B. ein Systementwickler die generische Formulierung „ab einem bestimmten Zeitpunkt darf das Ereignis *e* nur genau n-mal eintreten“ sucht, dann soll in dem Fall der Systementwickler durch Gebrauch des Safety-Pattern-Konzepts über die Frage explizit nachdenken müssen, ob das Wort „nur“ an der richtigen Stelle ist, oder ob die Formulierung nicht so lauten müsste: „nur ab einem bestimmten Zeitpunkt darf das Ereignis *e* genau n-mal eintreten“.

Dabei geht es auch um die Vermeidung schwieriger Kompositionen beim Gebrauch des Safety-Pattern-Konzepts. Zum Beispiel könnte die folgende generische Formulierung (1) auch durch die Komposition der Formulierungen (2) und (3) erfolgen. Diese Komposition kann jedoch unerfahrenen Spezifizierern schwierig erscheinen.

(1) *Bis zu dem Zeitpunkt, zu dem* `b` *gültig ist, darf* `a` *nur gelten, wenn* `a` *ununterbrochen bis zu diesem Zeitpunkt gilt.*

Korrekte Kompositionsmöglichkeit für die Bildung von (1):

(2) *Bis zu dem Zeitpunkt, zu dem* `b` *gültig ist, muss* `a` *ununterbrochen gültig sein.*

ODER

(3) *Bis zu dem Zeitpunkt, zu dem* `b` *gültig ist, darf* `a` *nicht gültig sein.*

Bei der Frage, für welche Spezifikationsprobleme spezielle Safety-Pattern-Klassen gebraucht werden und für welche nicht, muss entschieden werden, was vom Systementwickler erwartet werden kann und was nicht. Zum Beispiel hat, wie bereits in Abschnitt 2.4.1.1 erläutert, jeder Automatisierungstechnik-Ingenieur eine ausführliche Ausbildung in Aussagenlogik erhalten. Wollte man für einfache Probleme der Aussagenlogik die Anwendung spezieller Safety-Pattern-Klassen einführen, so würde dies durch die stark anwachsende Zahl der Safety-Pattern-Menge der Attraktivität des Verfahrens entgegenwirken. Triviale Spezifikationsprobleme wie z. B. einfache UND-Verknüpfungen aussagenlogischer Variablen möchte der Systementwickler selbstständig beschreiben. Die Bereitstellung von Spezifikationsmustern für triviale Spezifikationsprobleme würde die Anwendung des Ansatzes zu umständlich und zu aufwendig machen.

In Abschnitt 6.1.4 wird eine Klassifikation der Safety-Pattern eingeführt, anhand der die Safety-Pattern selektiert werden. Jede weitere Formulierungsart, die in der Menge der nützlichen generischen Formulierungen berücksichtigt werden soll, bedeutet eine weitere Safety-Pattern-Klasse. Jede Klasse hat für eine sinnvolle und vollständige Klassifikation auch hierzu antagonistische Klassen zur Folge. Generell gilt jedoch, dass es nicht zu viele Safety-Pattern geben darf, denn sonst ist die Anwendung des Safety-Pattern-Ansatzes zu aufwendig[15]. Die Einführung zusätzlicher Safety-Pattern ist nicht der einzige Weg zur Vermeidung von Fehlern bei der Komposition. Kompositionsfehler lassen sich auch durch eine werkzeuggestützte geführte und

[15] Antagonistische Klassen bringen ein Klassifikationskriterium mit sich. Jedes Klassifikationskriterium bedeutet für den Selektionsprozess, dass der Systementwickler eine Entscheidung treffen muss, welcher Art das gesuchte Safety-Pattern ist, vgl. Abschnitt 6.1.4.

kontrollierte Komposition vermeiden. Dabei kann beim Kompositionsschritt durch gezielte Fragen vermieden werden, dass dem Systementwickler Spezifikationsfehler unterlaufen.

Somit lassen sich die folgenden Einflussgrößen festhalten, durch welche die Menge der nützlichen generischen Formulierungen bestimmt wird (vgl. [ABG04]):

- *Die Safety-Pattern-Menge unterstützt auch die bekannten Spezifikationsprobleme der Automatisierungstechnik, für die keine speziellen temporallogischen Konstrukte vorgesehen sind, durch spezielle Safety-Pattern. Beispiel: Ausdrücken expliziter Zeitaussagen.*
- *Mithilfe der nützlichen generischen Formulierungen sollen schwierige Kompositionen beim Gebrauch des Safety-Pattern-Konzepts vermieden werden.*
- *Die Safety-Pattern-Menge muss schlüssig sowie vollständig und geschlossen klassifiziert sein.*
- *Durch die Entscheidung, welches Safety-Pattern das geeignete ist, soll der Systementwickler gezwungen werden, über verschiedenen Ausdrucksmöglichkeiten nachzudenken, um den korrekten Ausdruck für sein Spezifikationsproblem zu identifizieren.*
- *Eine zu große Safety-Pattern-Menge muss vermieden werden, um eine einfache Safety-Pattern Selektion und die Realisierbarkeit des Safety-Pattern-Katalogs zu gewährleisten.*

6.1.4 Klassifikation der Safety-Pattern und die resultierende Safety-Pattern-Menge

Auf Basis der Menge der notwendigen generischen Formulierungen und den in den letzten beiden Abschnitten beschriebenen Einflussgrößen wurde die Safety-Pattern-Klassifikation mit 13 Klassifikationskriterien mit jeweils 2-6 Klassen bestimmt. Anhang B enthält einen Ausschnitt aus der Klassifikation. Die Basis für die Safety-Pattern-Menge ist die Menge der notwendigen Formulierungen, die in Abschnitt 6.1.2 bestimmt wurde. In Abschnitt 6.1.3 wurden weitere Safety-Pattern-Arten (Klassen) ermittelt, die für die Anwendbarkeit des Verfahrens wichtig sind. Diese führen zu Safety-Pattern, die sich aus der Menge der Notwendigen ableiten lassen. Werden die Safety-Pattern-Klassen, die zueinander nicht widersprüchlich sind, miteinander kombiniert, ergibt sich hieraus die gesamte Safety-Pattern-Menge.

Durch die Kombination der verschiedenen Klassen von Sicherheitsanforderungen werden die charakteristischen Merkmale (vgl. Abschnitt 5.3.2) eines Safety-Pattern beschrieben, das dann entsprechend spezifiziert werden muss. Durch Kombination der in Abschnitt 6.1.4 beschriebenen Klassifikationskriterien entsteht somit ein Klassifikationsbaum der Safety-Pattern. Solch ein Klassifikationsbaum führt den Systementwickler zu dem Safety-Pattern, das für die zu spezifizierende Sicherheitsanforderung benötigt wird, indem die zu spezifizierende Sicherheitsanforderung den einzelnen Klassen zugeordnet wird.

Die Zusammenhänge werden im Folgenden anhand von Grafiken erläutert. In Abbildung 6.7 sind unterschiedliche Klassifikationskriterien für die unterschiedlichen Arten von Sicherheitsanforderungen durch Verzweigungen dargestellt. Dabei symbolisieren die verschiedenen Farben unterschiedliche Klassifikationskriterien. Jedes Klassifikationskriterium entscheidet über die Zuordnung einer Sicherheitsanforderung zu einer von zwei oder mehreren Klassen.

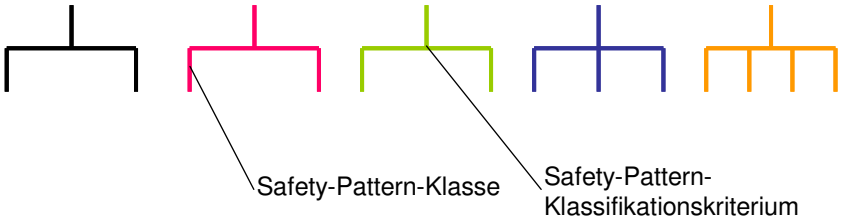


Abbildung 6.7: Darstellung von Safety-Pattern-Klassifikationskriterien

In Abbildung 6.8 werden diese Klassifikationskriterien miteinander kombiniert. Dadurch entsteht ein Klassifikationsbaum, der die charakteristischen Merkmale der Safety-Pattern festlegt. An unterster Stelle des Baumes kann jedem Zweig genau ein Safety-Pattern zugeordnet werden.

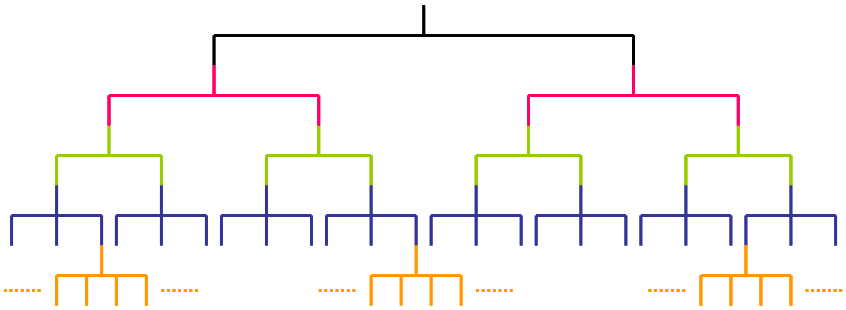


Abbildung 6.8: Kombination der Klassifikationskriterien

Allerdings sind nicht alle Kombinationen der Klassifikationskriterien logisch sinnvoll (vgl. Abbildung 6.9). Die Klasse „statische Sicherheitsanforderungen“ umschreibt zum Beispiel Sicherheitsanforderungen, welche die Geltung einer nicht-zeitlichen Eigenschaft in allen Zuständen des Zustandraums des Systemmodells beinhalten. Hier ist es zum Beispiel irrelevant, weiter zu klassifizieren, ob es in der Sicherheitsanforderung um den Gültigkeitsbeginn oder die Gültigkeitsdauer einer Eigenschaft geht.

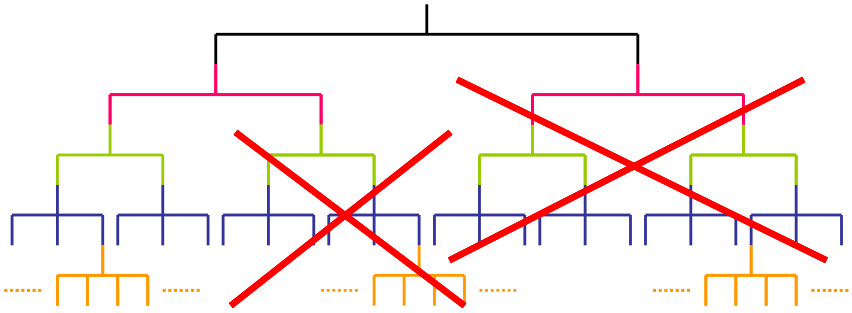


Abbildung 6.9: Ausschluss nicht sinnvoller Klassenkombinationen

Das Ergebnis ist ein Klassifikationsbaum. Er führt zu den einzelnen Safety-Pattern, die selektiert werden, indem die zu spezifizierende Sicherheitsanforderung den einzelnen Klassen zugeordnet wird (vgl. Abbildung 6.10)[16].

Die Klassifikationskriterien wurden in einem möglichen Klassifikationsbaum geordnet. Aus den möglichen Kombinationen ergibt sich die Zahl von 454 Safety-Pattern.

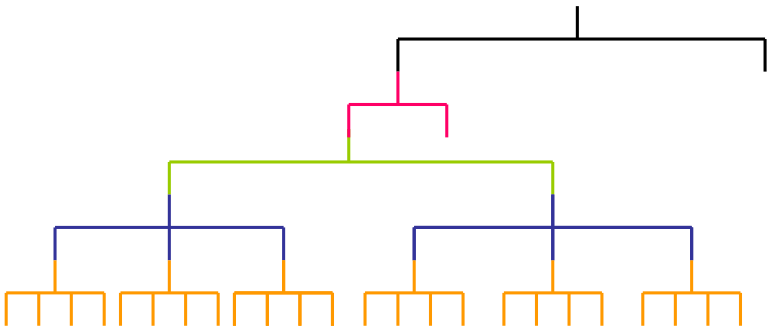


Abbildung 6.10: Schematische Darstellung eines Klassifikationsbaums zur Selektion des geeigneten Safety-Pattern

6.2 Entwicklung der Safety-Pattern-Normsprache

In Abschnitt 5.3.4.2 wurde die Bedeutung und der Nutzen einer Safety-Pattern-Normsprache herausgestellt. Formale Sicherheitsanforderungen können in einfacher intuitiv verständlicher und zudem präziser Art und Weise dokumentiert werden, indem Safety-Pattern neben generischen formalen Formulierungen ebenso generische Formulierungen in Normsprache umfassen. Dadurch wird eine direkte exakte Interpretation von Sicherheitsanforderungen ermöglicht.

Für Worte der natürlichen Sprache, die für die Spezifikation von Sicherheitsanforderungen eine Rolle spielen und die mehrere mögliche Interpretationen besitzen, müssen unterschiedliche adäquate Begriffe festgelegt werden.

Eine konventionelle Sicherheitsanforderung für eine Türschließsteuerung lautet z. B.: *Nur nach dem Empfang des Fernschließbefehls darf die Ventilschließfunktion 2b ausgeführt werden.*

Die Frage ist hier, was das Wort „nach“ bedeutet. Dieses Wort ist mehrdeutig. Es gibt drei mögliche Interpretationen hinsichtlich der Aufeinanderfolge der *Ventilschließung* auf *Fernschließbefehl_Empfang* (mit *Fernschließbefehl_Empfang* := p und *Ventilschließung* := q lässt sich das abgekürt als „q nach p“ ausdrücken):

1. „q ab p“: In der Aufeinanderfolge darf die Ventilschließfunktion ab dem Zeitpunkt ausgeführt werden (q darf gültig sein), zu dem der Fernschließbefehl empfangen wird (p ist

[16] Die Struktur des Baumes der Abbildung 6.10 entspricht dabei nicht dem realen Klassifikationsbaum, der insgesamt 13 und nicht nur 5 Klassifikationskriterien umfasst.

gültig), vgl. Abbildung 6.11. Es gibt also eine Zeit, zu der sich die Gültigkeit von p und q überschneiden. Formal kann diese Relation in CTL folgendermaßen ausgedrückt werden:

```
A(not Ventilschließung W Fernschließbefehl_Empfang)
```

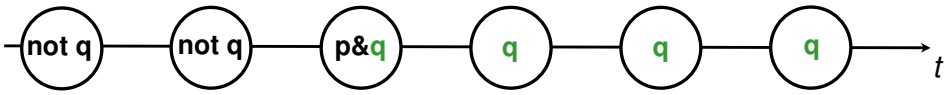


Abbildung 6.11: Veranschaulichung der Relation „q darf ab p gültig sein“

2. „q strikt nach p“: Zum Zeitpunkt des Fernschließbefehlempfangs (p ist gültig) sowie davor, darf die Ventilschließfunktion noch nicht ausgeführt werden (q ist nicht gültig), vgl. Abbildung 6.12. Danach ist eine Ausführung ab irgendeinem Zeitpunkt erlaubt. Formal kann diese Relation in CTL folgendermaßen ausgedrückt werden:

```
A(not Ventilschließung W (Fernschließbefehl_Empfang and not
Ventilschließung))
```

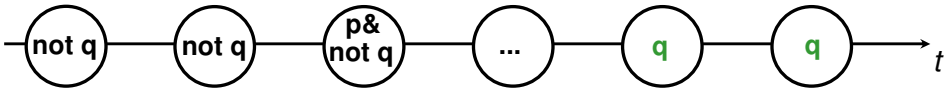


Abbildung 6.12: Veranschaulichung der Relation „q darf strikt nach p gültig sein“

3. „q exakt nach p“: In der Aufeinanderfolge darf die Ventilschießfunktion vor dem Zeitpunkt und in diesem Zeitpunkt, zu dem der Fernschließbefehl empfangen wird (p ist gültig), nicht ausgeführt werden (q ist nicht gültig). Bei zeitgetriggerten Systemen darf die Ventilschließfunktion nur im nächsten Zeitschritt ausgeführt werden. Für ereignisgetriggerte Systeme bedeutet dies, dass in der Aufeinanderfolge von p und q die Variable q nur eintreten darf, wenn keine andere definierte Systemvariable ihren Wert strikt nach p und vor dem Eintreten von q ändert. Formal kann diese Relation in CTL folgendermaßen ausgedrückt werden:

```
not Ventilschließung and AG (not Fernschließbefehl_Empfang →
AX not Ventilschließung)
```

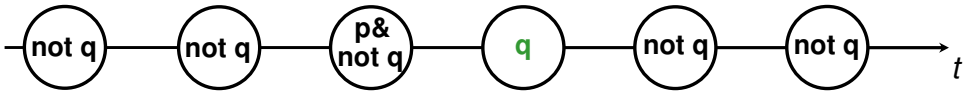


Abbildung 6.13: Veranschaulichung der Relation „q darf nur exakt nach p gültig sein“

Eine Herausforderung für das Safety-Pattern-Konzept ist es, den Entwickler auf solche Unterschiede bei der Spezifikation von Sicherheitsanforderungen aufmerksam zu machen und ihn zum Nachdenken zu zwingen, was exakt ausgedrückt werden soll. Dies geschieht bei der Auswahl der erforderlichen Safety-Pattern-Klassen und durch die genaue Unterscheidung von Begriffen in der Festlegung der Safety-Pattern-Normsprache. Durch die Safety-Pattern-Normsprache werden Missverständnisse durch eine genormte Terminologie vermieden. Für die Fachwörter, die für die Spezifikation von Sicherheitsanforderungen erforderlich sind, (auch Themen-

wörter genannt, z. B. „Ereignis“, „Aktion“, „auftreten“, „erreichen“ sowie „erreichbar“) [Ortn97, S. 83 u. 95] wird ein Lexikon erstellt.

Für die Entwicklung der Safety-Pattern-Normsprache wurde zunächst analysiert, welche aus der natürlichen Sprache bekannten Begriffe für eine äquivalente Beschreibung der unterschiedlichen formalen generischen Formulierungen erforderlich sind. Die grundlegenden Begriffe ergeben sich bereits aus der Analyse der Safety-Pattern-Klassifikation. Für die unterschiedlichen Safety-Pattern-Klassen müssen verschiedene adäquate Worte bestimmt werden. Für das Klassifikationskriterium „Art des Gültigkeitsbeginns“ wird beispielsweise unterschieden, ob die Gültigkeit *ab* einem bestimmten Zeitpunkt oder *strikt nach* einem bestimmten Zeitpunkt gelten soll. Während *strikt nach* den Zeitpunkt ausschließt, von dem der Gültigkeitsbeginn abhängt, ist dieser Zeitpunkt bei *ab* inbegriffen.

Gemäß den in Abschnitt 2.3.2 eingeführten Richtlinien werden ungeschickte Sprachregelungen aufgedeckt und bei der Festlegung der genormten Terminologie beseitigt. Die Festlegung der Verwendungsweise und Bedeutung eines Begriffs geschieht durch eine explizite Definition der Bedeutung an sich und durch die Erläuterung der Verwendung in Form von Beziehungsdefinitionen zu anderen Begriffen (Prädikatorenregeln). Häufig verwendete Formen von Prädikatorenregeln sind insbesondere Synonymität (sog. Synonyme) und Konträrität (sog. Antonyme). Definitionsweise:

- Explizite Definition und Nennung der Wortart
- Erläuterung
- Angabe von Beziehungen zu anderen Wörtern
- Beispiele bzw. Gegenbeispiele

Kern der Definition ist die *explizite Definition*. Diese muss für das definierte Wort eingesetzt und es somit direkt ersetzen können. Wesentlich für die Definition dieser Fachwörter ist gemäß [Ortn97, S. 95] eine „größtmögliche Kontextunabhängigkeit“. Wörter werden durch schon bekannte Wörter definiert [Ortn97, S. 82], wobei zwei Wörter niemals für die gegenseitige Definition verwendet werden dürfen.

Beispiel – Definition des Wortes „permanent“:

Explizite Definition: „permanent“ (*Adj.*) =DF ohne Unterbrechung andauernd

Erläuterung: „permanent“ behandelt die andauernde Gültigkeit einer Eigenschaft. Es bezeichnet die Gültigkeit einer Aussage in einem Zeitintervall ohne Unterbrechung. Dabei hat „permanente Gültigkeit“ den Sinn von einer „ununterbrochen andauernden Gültigkeit“.

Beziehungen:
Synonyme sind „immer“ und „ununterbrochen“. Antonyme sind „genau n Mal“, „mindestens n Mal“, „maximal n Mal“ und „beliebig oft“.

Beispiel:

Safety-Pattern Nr. 5: q muss von dem Zeitpunkt an *permanent* gültig sein, zu dem p eintritt.

Die Definition der Safety-Pattern-Normsprache erfolgt rein durch Wörter aus der natürlichen Sprache. Der Nachteil der Verwendung logischer Partikel für die Definition ist, dass die Ausdrucksmöglichkeiten für die Definitionen zu stark eingeschränkt werden. Die Resultate können stark verschachtelte komplizierte Definitionsbeziehungen sein. Prädikatenlogische Partikel im Sinne normsprachlicher Hilfsverben als Bestandteil der Normsprache sind vielmehr als Hindernis für die einfach zu handhabende Anwendung und für das Verständnis der Normsprache zu sehen. Zudem führt dies zu stark künstlich wirkenden Formulierungen. Aus demselben Grund wird auf die Definition spezieller Satzbauregeln verzichtet. Stattdessen werden die der natürlichen Sprache übernommen. Dies ist möglich, da die Normsprache für die Safety-Pattern nicht dazu verwendet werden soll, selbstständig Aussagen zu bilden. Alle möglichen Aussagen werden durch die Safety-Pattern vorgegeben und der Anwender muss die passende Aussage aus dem Safety-Pattern-Katalog wählen. Durch die endliche und überschaubare Menge von Safety-Pattern sind implizit alle Satzbauregeln festgelegt.

Bei der Safety-Pattern-Normsprache geht es nicht um die Ermöglichung der gesamten Formulierung von Sicherheitsanforderungen in Normsprache, sondern nur um Formulierungen bezüglich der Eigenschaften von Safety-Pattern. Mithilfe der entwickelten Normsprache müssen z. B. keine Formulierungen erstellt werden können, die aussagenlogischen Ausdrücken entsprechen, sondern nur entsprechende Formulierungen zu den formalen generischen Formulierungen der Safety-Pattern. In der Formulierung in Normsprache werden also dieselben Systemvariablen bzw. aussagenlogische Verknüpfungen von Systemvariablen verwendet, wie in der formalen Formulierung. Diese erfordern insbesondere Formulierungen bezüglich zeitlich-logischer Aspekte. Dadurch ist der Terminologieumfang dieser Normsprache wenig umfangreich und damit überschaubar und einfach kontrollierbar.

Die generischen Formulierungen der Safety-Pattern in der Safety-Pattern-Normsprache wurde manuell spezifiziert. Dadurch werden künstlich wirkende Formulierungen vermieden und somit die Natürlichkeit der Formulierungen unterstützt. Dies wäre nicht möglich, wenn diese Formulierungen aus den temporallogischen Formulierungen oder aus den grundlegenden generischen Formulierungen automatisch generiert würden. Dann könnte sich leicht ein komplizierter Satzbau mit nicht immer passenden Begriffen ergeben (wenig flexible Begriffswahl). Die Folge wären künstlich wirkende Formulierungen. In Anhang C wird hierfür ein Beispiel gegeben, durch das dies belegt wird.

6.3 Entwicklung einer geeigneten grafischen Notation zur Spezifikation mit Safety-Pattern

In Abschnitt 5.3.4.3 wurde bereits die Zielsetzung und die Anforderungen an eine geeignete grafische Notation festgelegt. Es wurde herausgestellt, dass mit der grafischen Notation zeitliche Relationen zustandsbasiert dargestellt werden müssen. Für alle Safety-Pattern muss eine einfach verständliche Darstellung möglich sein. Bevor eine neue Notation entwickelt wird, muss analysiert werden, ob eine existierende grafische Notation die Anforderungen erfüllt und für das Safety-Pattern-Konzept verwendet werden kann oder ob aus solchen Notationen für die Entwicklung eines neuen Beschreibungsmittels gelernt werden kann. Bereits in Abschnitt 4.2.3 wurden grafische Beschreibungsmittel vorgestellt, die für die Spezifikation von Anforderungen in Temporallogik für die formale Verifikation verwendet werden. Sequence Timing Diagrams (STD) [ScDa93, SJW98] sind sehr gut und intuitiv verständlich, da hier eine Notation verwendet wird, die zumindest in der Hardwareentwicklung bekannt ist und dort verwendet wird. Auch mit Life Sequence Charts (LSC) [DaHa99] sowie MSC_{CTL} [Canv99] wird in Anlehnung an MSC und UML-Sequenzdiagramme eine somit zumindest teilweise vertraute Notation verwendet. Bereits in Abschnitt 4.2.3 wurde jedoch festgestellt, dass es für Automatisierungssysteme erforderliche sicherheitsrelevante temporallogische Ausdrücke gibt, die nicht mit diesen Notationen ausgedrückt werden können. Darstellungen für Zustände, wie von Zustandsdiagrammen bekannt, können in diesen Notationen nicht dargestellt werden. Bei LSC und MSC_{CTL} ist die Darstellungsweise stattdessen nachrichtenorientiert. Somit sind zwei Grundvoraussetzungen für eine Eignung nicht gegeben.

Eine weitere Notation zur Darstellung logischer Ausdrücke sind Constraint Diagrams [GHK99]. Es ist eine grafische Alternative zu OCL und besitzt somit jedoch wie OCL Mängel hinsichtlich der Darstellung zeitlicher Relationen. Es ist auch keine zustandsbasierte Darstellung, sodass aus dieser Darstellungsweise keine nützlichen Aspekte übernommen werden können.

Keine der in Abschnitt 4.2.3 untersuchten grafischen Notationen besitzt eine adäquate Darstellungsweise. Deshalb wurde eine geeignete Notation entwickelt. Da diese Notation speziell für die Spezifikation mit Safety-Pattern angewandt werden soll, wird sie GNSS (**G**rafische **N**otation zur **S**pezifikation mit **S**afety-Pattern bzw. **G**rafical **N**otation for **S**pecification with **S**afety Patterns) genannt.

Bei einer zustandsbasierten Darstellung müssen Zustandsfolgen visualisiert werden können. Für den Ausdruck (6.4) ist in Abbildung 6.14 beispielhaft ein Zustandsdiagramm dargestellt, das Zustandsfolgen enthält, welche die Anforderung des Ausdrucks erfüllen.

$$AG\ (p \rightarrow AX\ AF\ q) \qquad (6.4)$$

(6.4) bedeutet, dass strikt nach jedem Zustand, zu dem p gültig ist, q irgendwann gelten muss.

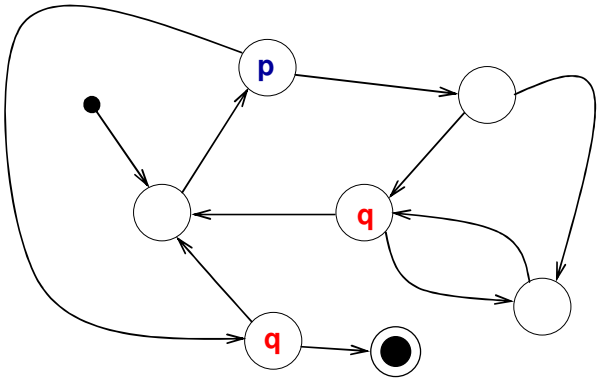


Abbildung 6.14: Beispiel eines Zustandsdiagramms für die Erfüllung des Ausdrucks (6.4)

Möchte man nun dies in grafischer Notation statt in Beispielen allgemein gültig spezifizieren, muss Folgendes gegeben sein:

- Es muss eine Zustandsfolge spezifiziert werden, ohne Zustände darzustellen, die dazwischen liegen können, damit die Anzahl und die Struktur dieser Zustände beliebig ist. Eine zwischen Zuständen mögliche beliebige Zustandsstruktur muss erkennbar sein. Am Beispiel des Ausdrucks (6.4) könnte das, wie in Abbildung 6.15 veranschaulicht, dargestellt werden. Dabei wird durch gestrichelte Linien ausgedrückt, dass hinter solch einem gestrichelten Übergang beliebig viele Zustände mit Zustandsübergängen liegen können.

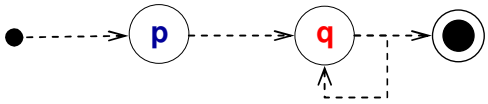


Abbildung 6.15: Darstellung einer Relation zwischen `p` und `q` mit Reduktion von Informationen hinsichtlich irrelevanter Zustandsstrukturen

Bevor `p` eintritt können beliebig viele Zustände liegen. Wichtig ist, dass nach der Gültigkeit von `p` irgendwann `q` folgt. Wann ist jedoch irrelevant. Nach `q` bis zur Terminierung können wiederum beliebig viele Zustandsübergänge sein, und `q` kann beliebig oft gelten.

- In Abbildung 6.15 fehlen die Informationen, ob hier eine Bedingung vorhanden ist und welcher Zustand bzw. welche Variable als Bedingung anzusehen ist. Folglich müssen Bedingungen ausgedrückt werden können. Von Zustandsdiagramm-Notationen wie UML oder von Harel-Statecharts kennt man, dass Bedingungen in rechteckigen Klammern ausgedrückt werden.

- Zudem ist in der Darstellung von Abbildung 6.15 unbekannt, ob `p` oder `q` irgendwann gelten müssen. Es ist also eine Unterscheidung erforderlich, ob eine Aussage

    - möglicherweise gelten könnte,
    - unbedingt gelten muss oder
    - ob keine Aussage darüber gemacht wird (Beliebigkeit).

Hiermit wird die Modalität einer Geltung unterschieden. Für diese Unterscheidung ist keine gebräuchliche grafische Notation bekannt. In Abbildung 6.16 wird für „Möglichkeit der Geltung“ eine Raute und für „Notwendigkeit der Geltung“ ein Dreieck verwendet. Irgendwann kann es in dem Beispiel den Zustand geben, in dem p erfüllt ist. Dann ist p eine Bedingung für das folgende Gelten von q. Die Eigenschaft q muss irgendwann gelten. Wann, ist jedoch unbestimmt.

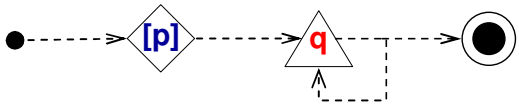


Abbildung 6.16: Darstellung einer Relation zwischen p und q mit Kennzeichnung der Bedingung p und der Modalitäten

- Schließlich fehlt in der Darstellung von Abbildung 6.16 noch die Information, dass die Relation für jedes p gelten soll und nicht nur für das erste p, das eintritt. Dies kann ausgedrückt werden, indem nicht der Startzustand vorangestellt wird, sondern ein beliebiger Zustand, sodass die Relation für jeden Zustand des Systems gilt. Das Ergebnis ist in Abbildung 6.17 dargestellt.

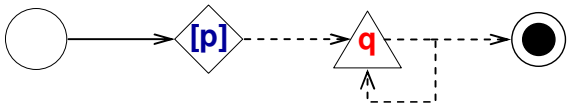


Abbildung 6.17: Grafische Darstellung für den Ausdruck (6.4)

Es muss möglich sein, genau einen Zustandsübergang auszudrücken für die Relationen, in denen eine direkte Zustandsfolge formuliert wird. Dies kann durch ein gewöhnliches Zustandsübergangssymbol ausgedrückt werden. Anhand des Beispiels kann dies jedoch nicht veranschaulicht werden.

In Anhang D erfolgt eine Einführung in die GNSS-Notation.

6.4 Entwicklung von grafischen Darstellungen zur Erläuterung der Safety-Pattern

6.4.1 Verbreitete Darstellungsweisen

Wie in Abschnitt 5.3.7 ausgeführt wurde, ist der Einsatz von Grafiken bei der Erläuterung von Safety-Pattern aufgrund der Veranschaulichung logischer Zusammenhänge von Bedeutung. Im Unterschied zu GNSS-Spezifikationen geht es hier nicht um allgemein gültige generische Spezifikationen, sondern um die Erläuterung der charakteristischen Aspekte durch Beispiele.

Generell kann man zwischen der *statischen Visualisierung in Form von Bildern* und der *dynamischen Visualisierung durch Simulationen* unterscheiden.

Solche Visualisierungen sind dann einfach verständlich, wenn vertraute Darstellungsweisen verwendet werden. Zwei Arten der Visualisierung sind geeignet, welche die Anforderungen aus Abschnitt 5.3.7 erfüllen:

1. **Visualisierung erfüllter Variablen im Ausschnitt eines beliebigen Zustandsraums:** Dies ist eine häufig verwendete Art der Visualisierung, die bei der Einführung in Temporallogik verwendet wird (z. B. in [Béra01], [CGL94], [HuRy00], [McMi97], [DiOl03]). Es ist vorteilhaft, wenn die Darstellungsweise der grafischen Visualisierung die Vorstellung der Entwickler unterstützt. Das Ziel der Verwendung von Safety-Pattern ist, Sicherheitsanforderungen zu spezifizieren, deren Erfüllung in einem automatenbasierten Systemmodell überprüft werden soll. Folglich ist es sinnvoll, die Bedeutung im Zustandsraumausschnitt eines beliebigen automatenbasierten Modells zu zeigen.

2. **Zeitdiagramme:** In einem Koordinatensystem wird über einer Zeitachse (X-Achse) die Werteänderung der Variablen (Y-Achse) aufgetragen. Dies ist eine allgemein bekannte Art der Darstellung. Wird nun der Werteverlauf aller Variablen der generischen Formulierung eines Safety-Pattern gleichzeitig dargestellt, ist dies eine geeignete Darstellung.

In den folgenden beiden Unterabschnitten werden diese beiden Visualisierungsmöglichkeiten genauer untersucht.

6.4.2 Visualisierung im Ausschnitt eines beliebigen Zustandsraums

In Abbildung 6.18 ist ein Beispiel der Darstellungsform gezeigt, wie sie in [McMi97] verwendet wird. Bei dieser Art der Visualisierung wird eine beliebige Zustandsraum-Struktur dargestellt, die nach ein paar Zustandsübergängen abgebrochen wird (gestrichelte Linie in Abbildung 6.18). Die Variable `p` wird in den Zuständen visualisiert, in denen `p` gelten soll. Im Beispiel wird von einem Zustandsraum ausgegangen, der die Anforderung erfüllt, dass die permanente Gültigkeit von `p` von Systembeginn an möglich sein muss. In CTL ist dies durch `EG p` ausgedrückt. Um dies zu veranschaulichen, wird `p` in allen Zuständen eines sog. „Berechnungspfads“ visualisiert. Ein Berechnungspfad (Zustandsfolge) beschreibt ein mögliches Systemverhalten.

Um nicht nur einfache Formulierungen veranschaulichen zu können, ist es wichtig, nicht nur erfüllte Variablen anzuzeigen, sondern auch zu veranschaulichen, in welchen Zuständen Variablen nicht erfüllt sein sollen. Was bei der gebräuchlichen Art der Erläuterung fehlt, ist die Unterscheidung, ob die visualisierte Variable in dem jeweiligen Zustand gelten muss oder ob dies nur eine Möglichkeit der Geltung ist. Deshalb wurde diese Visualisierungsart bei der Verwendung für das Safety-Pattern-Konzept durch die Unterscheidung zwischen möglicher

Geltung und notwendiger Geltung erweitert, indem unterschiedliche Farben[17] verwendet werden, s. Abbildung 6.19. Grün wird gebraucht, um mögliche Werte von Variablen zu kennzeichnen, und schwarz, wenn der visualisierte Wert in dem jeweiligen Zustand gelten muss. Ist eine Variable in einem Zustand nicht visualisiert, bedeutet dies, dass der Wert der Variable hier beliebig ist. In Abbildung 6.19 wird folgender Ausdruck veranschaulicht: `q` muss ununterbrochen gelten, bevor `p` eintritt. In CTL wird dies so formuliert: `A(q W p)`.

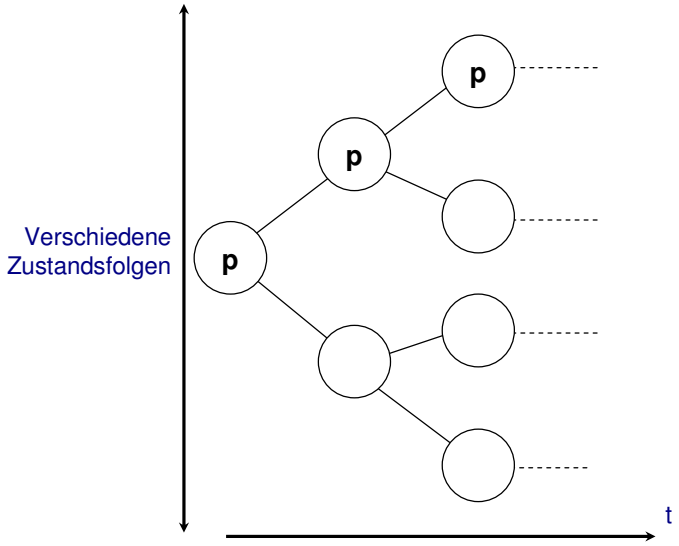


Abbildung 6.18: Veranschaulichung von `EG p` in [McMi97]

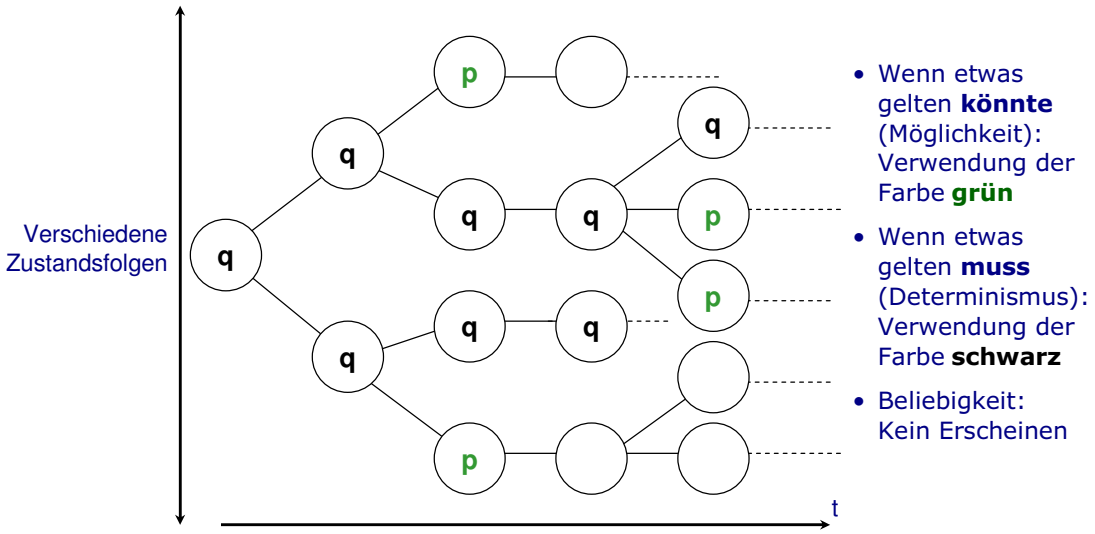


Abbildung 6.19: Erweiterte Veranschaulichungstechnik für das Saftey-Pattern-Konzept

Durch die gleichzeitige Visualisierung mehrerer Zustandsfolgen können verschiedene mögliche Verläufe und Abhängigkeiten der betrachteten Variablen erläutert werden. Eine Simulation ist nicht erforderlich, in der unterschiedliche Verläufe nacheinander dargestellt werden. Deshalb ist diese Art der Darstellung als statische Visualisierung in Bildform geeignet.

[17] Da die grafischen Darstellungen zur Erläuterung für Safety-Pattern dienen und deshalb ausschließlich mit einem Softwarewerkzeug auf einem Bildschirm angezeigt werden, können Farben verwendet werden.

Zur Veranschaulichung von Erreichbarkeit werden Pfeile eingeführt. Diese gehen von den Zuständen aus, von denen die Erreichbarkeit gelten soll, und enden in den Zuständen, zu denen die jeweilige Eigenschaft beispielsweise erreicht wird (vgl. Abbildung 6.20 mit dem Beispiel „ab jedem Zustand zu dem `q` gilt, muss `p` erreichbar sein“, in CTL: `AG [q → EF p]`).

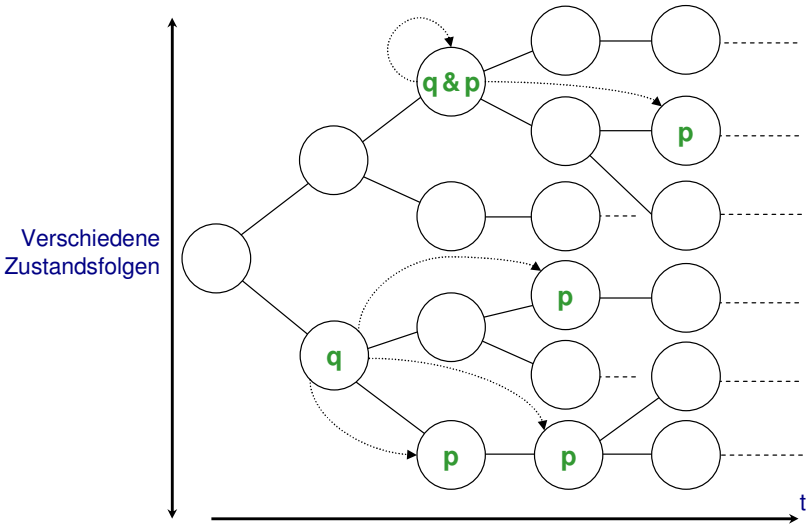


Abbildung 6.20: Veranschaulichung von Erreichbarkeitsrelationen

6.4.3 Erläuterung mithilfe von Zeitdiagrammen

Im Gegensatz zur vorigen Visualisierungsart kann bei Zeitdiagrammen jeweils nur ein Ablauf (eine Zustandsfolge) dargestellt werden. Deshalb ist es hier sinnvoll, Simulationen zu verwenden, bei denen verschiedene Variablenwertefolgen nacheinander simuliert werden können. Um für den betrachteten Zeitraum alle Möglichkeiten zu berücksichtigen, ist eine interaktive dynamische Visualisierung sinnvoll, bei der der Benutzer den Verlauf der Variable vorgibt, durch welche der Verlauf der davon abhängigen Variablen bestimmt wird. Bei jedem Safety-Pattern gibt es eine Variable, die von einer Bedingungsvariablen abhängt. Da die abhängige Variable eine bestimmte Eigenschaft ausdrücken soll, die unter bestimmten Bedingungen gilt, wird sie Eigenschaftsvariable genannt. In der verwendeten Zeitdiagramm-Darstellung wird stets die jeweilige Eigenschaft in Abhängigkeit zur Bedingung dargestellt.

Abbildung 6.21 zeigt ein Beispiel für die Visualisierung in einem Zeitdiagramm. Die Zeitachse ist dabei durch Zustandübergänge unterteilt. Dabei wird folgender Ausdruck veranschaulicht: „Die Aussage `q` darf erst gelten, nachdem die Aussage `p` mindestens einmal aufgetreten ist.“ Der äquivalente Ausdruck in CTL ist: `A(not q W (p and not q))`. Da es in diesem Ausdruck um die Gültigkeit von `q` in Abhängigkeit zu `p` geht, gibt der Benutzer den Wert von `p` in jedem Zustand vor (oberer Graph; Vorgabe z. B. schrittweise mithilfe von Kontroll-Buttons). Im unteren Graphen wird entsprechend der Wert von `q` angezeigt. Dabei wird durch die Linienart unterschieden, wann `q` deterministisch und wann beliebig ist (vgl. Abbildung 6.21).

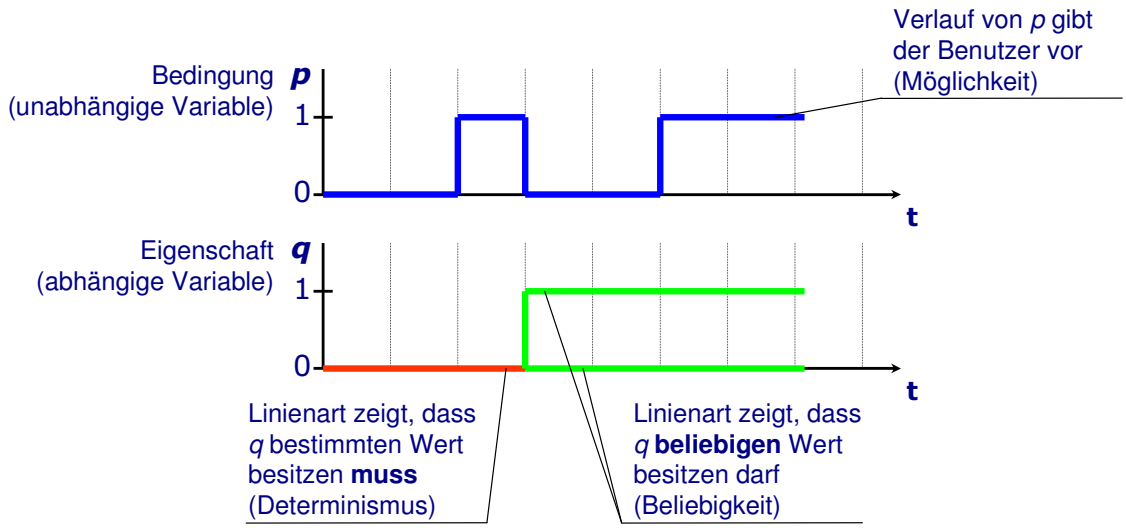


Abbildung 6.21: Veranschaulichung der Bedeutung eines Safety-Pattern durch Simulation

Wie bei der statischen Visualisierung muss ausgedrückt werden können,

- dass eine Variable einen bestimmten Wert besitzen muss (Determinismus),
- dass die Variable beispielsweise den angezeigten besitzt (Möglichkeit) oder
- dass der Wert der Variable irrelevant ist (Beliebigkeit).

Für den Determinismus wird im Simulationskonzept die Farbe rot verwendet. Für das Ausdrücken für Möglichkeiten wird die blaue Farbe verwendet. Im Fall von Beliebigkeit des Variablenwerts wird die grüne Farbe verwendet. Zudem wird in diesem Fall der Variablenwert mit „0“ und „1“ angezeigt.

Damit bei Erreichbarkeitseigenschaften die charakteristischen Merkmale des zu visualisierenden generischen Ausdrucks verständlich sind, wurden hierfür noch spezielle Symbole eingeführt. Statt durch durchgezogene Linien den Wert „1“ oder „0“ anzuzeigen, wird das Symbol Pfeilbeginn eingeführt, um den Ausgangszustand für eine Erreichbarkeit zu visualisieren. Ein solcher Pfeilbeginn ist sinnvollerweise immer rot, da es im Safety-Pattern-Konzept nie vorkommt, dass eine Erreichbarkeit lediglich eventuell möglich ist. Mit einem Pfeilende wird angezeigt, in welchem Zustand der Variablenwert „1“ erreicht wird. Sinnvollerweise ist das Pfeilende immer blau, da bei der Erreichbarkeit der Wert „1“ nicht erreicht werden muss. Um die Relation zwischen Erreichbarkeits-Ausgangspunkt und Punkt des Erreichens anzuzeigen, muss jede Pfeilspitze mit einem Pfeilbeginn verbunden werden (vgl. Beispiel in Abbildung 6.22 – die Pfeile beginnen im Zustand direkt nach `p`). Dabei wird in Abbildung 6.22 der generische Ausdruck `AG (p → AX EF q)` bzw. `q muss vom Zustand direkt nach p aus erreichbar sein` veranschaulicht. Statt den Wert „1“ zu erreichen, gibt es auch den Fall die permanente Gültigkeit von „1“ zu erreichen. Dies wird durch eine gestrichelte waagrechte Linie angedeutet, auf die der Pfeil zeigt. Schließlich gibt es noch den Veranschaulichungsfall, in dem der Variablenwert „1“ in einem Zustand nicht erreicht werden darf. Hierfür wird das

Symbol „x“ verwendet. Anhang E.1 gibt eine Übersicht über die verwendeten Symbole des Simulationskonzepts und deren Bedeutung.

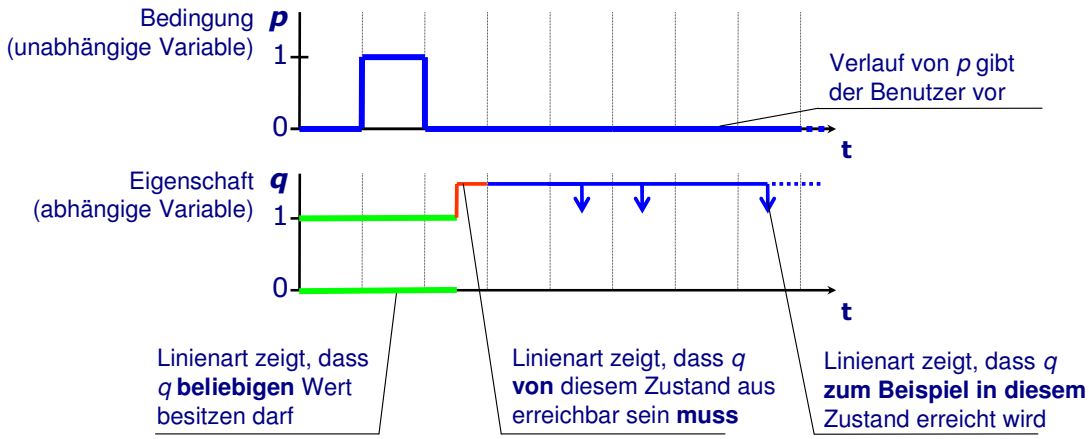


Abbildung 6.22: Veranschaulichung für ein Safety-Pattern mit Erreichbarkeitseigenschaft

Generell ist es auch möglich, die Eigenschaftsvariable vorzugeben und sich dann einen möglichen Verlauf für die Bedingungsvariable anzeigen zu lassen. Darauf wird bei der Realisierung noch genauer eingegangen (vgl. Abschnitt 7.3.2).

Da beim Simulationskonzept die Veranschaulichung dynamisch gebildet wird, eignet es sich nicht nur für die *Interpretation einzelner Safety-Pattern,* sondern auch für die *Veranschaulichung bei der Komposition von Safety-Pattern* und für die *Aufdeckung von Widersprüchen bei der Komposition von Safety-Pattern*. Zum Beispiel sollen die folgenden beiden generischen Formulierungen durch ein logisches UND miteinander verknüpft werden:

Nur wenn `a` eintritt, darf von nun an `c` beliebig oft gültig sein. Sonst darf `c` nie gültig sein.

`UND`

Ab dem Zeitpunkt, zu dem `b` erfüllt ist, muss `c` ununterbrochen gelten.

In beiden Formulierungen kommt die Eigenschaftsvariable `c` vor, die durch die Relation der ersten Formulierung von `a` und durch die Relation der zweiten Formulierung von `b` abhängt. Um die Frage zu beantworten, wie nun die beiden Formulierungen zusammenwirken und ob die beiden Relationen widersprüchlich oder in bestimmten Fällen widersprüchlich sind, ist eine Simulation des Gesamtausdrucks hilfreich.

Hier stellt sich nun noch die Frage, in welchen Fällen eine solche Kompositions-Simulation erforderlich ist. Ein logisches ODER ist uninteressant, da hier nur ein logischer Ausdruck gelten muss. Die getrennte Simulation aller Ausdrücke ist hier ausreichend. Weitere Ausdrücke wie Implikation „→“, die für viele Ingenieure ungewohnt sind und deshalb wohl nur wenig angewandt werden, können ebenso auch mit UND spezifiziert werden. Deshalb sind auch hier keine speziellen Simulationen erforderlich.

Bei der Komposition durch ein logisches UND sind folgende Fälle möglich:

1. Safety-Pattern 1 mit $c(a)$ und Safety-Pattern 2 mit $c(b)$: c hängt von 2 anderen Variablen ab. Deshalb ist eine integrierte Visualisierung sinnvoll, um die gemeinsame Abhängigkeit zu verdeutlichen und widersprüchliche Fälle erkennen zu können.

2. Safety-Pattern 1 mit $c(a)$ und Safety-Pattern 2 mit $c(a)$: c hängt von einer Variablen in verschiedener Weise ab. Wie in Fall 1 ist auch hier eine integrierte Visualisierung erforderlich.

3. Safety-Pattern 1 mit $c(a)$ und Safety-Pattern 2 mit $d(b)$: Dieser Fall ist uninteressant und widerspruchsfrei. Safety-Pattern 1 und Safety-Pattern 2 sind unabhängig voneinander. Dies kann in zwei Fenstern nebeneinander visualisiert werden. Eine integrierte Darstellung ist nicht erforderlich.

4. Safety-Pattern 1 mit $c(a)$ und Safety-Pattern 2 mit $a(b)$: Dieser Fall ist widerspruchsfrei. Die Abhängigkeit c von b kann in zwei Fenstern nebeneinander visualisiert werden. Eine integrierte Darstellung ist nicht erforderlich.

5. Safety-Pattern 1 mit $c(a)$ und Safety-Pattern 2 mit $b(a)$: Dieser Fall ist widerspruchsfrei. c und b hängen beide von a ab. Dies kann in zwei Fenstern nebeneinander visualisiert werden. Eine integrierte Darstellung ist nicht erforderlich.

Interessant ist dabei selbstverständlich nicht nur die Komposition zweier, sondern auch mehrerer Safety-Pattern.

6.5 Erläuterung der Safety-Pattern in natürlicher Sprache

In Abschnitt 5.3.7 wurde bereits aufgezeigt, dass es bei der Erläuterung von Safety-Pattern in natürlicher Sprache darauf ankommt, dass ausführlich und detailliert beschrieben wird, um die die Bedeutung des Safety-Pattern mit seinen Eigenschaften verständlich zu machen.

Im Unterschied zu kurzen prägnanten generischen Formulierungen, die wiederverwendbar sein sollen, kann hier durch die Erläuterung mit redundanten Informationen eine klare Verständlichkeit der Beschreibung erreicht werden. Verschiedene Aspekte werden im Detail und aus unterschiedlichen Perspektiven beschrieben, wodurch Redundanz erzielt wird. Ein Beispiel ist folgendes:

Generische Formulierung in CTL: $\mathrm{AG}\ ((a \text{ and } (t = 0)) \rightarrow \mathrm{AG}\ (((t = T) \text{ and } b) \rightarrow c))$

Generische Formulierung in Normsprache: Wenn b nach der Zeit T nach dem Eintreten von a erfüllt ist, so muss nach der Zeit T nach a c gelten.

Erläuterung in natürlicher Sprache: Im Vordergrund steht hier, dass eine Eigenschaft c unter bestimmten Voraussetzungen gelten muss. Sobald eine Bedingung a erfüllt ist und nach Ablauf

der Zeit T die Bedingung b erfüllt ist, muss im selben Zeitpunkt die Eigenschaft c gültig sein. Dabei wird die Zeit T ab dem Zeitpunkt gemessen, zu dem die Bedingung a erfüllt ist. Ist b nicht zum Zeitpunkt des Ablaufens von T erfüllt, so muss c auch nicht auftreten. Ist die Bedingung a nie erfüllt, so muss c nie gültig sein. Ob b oder c vor Ablauf der Zeit T erfüllt sind, ist irrelevant. Genauso ist für die Geltung der Eigenschaft c unwichtig, ob a während der Zeit T oder nach Ablauf der Zeit T erfüllt ist. Die in der formalen Formulierung verwendete Variable t ist eine Zeitvariable die kontinuierlich heraufgezählt wird.

6.6 Anleitung zur Anwendung durch Musterbeispiele

Bei der Erläuterung durch Musterbeispiele ist die Zielsetzung die klare Verständlichkeit der Safety-Pattern-Bedeutung und -Anwendung. Dadurch soll die Bedeutung der generischen Formulierung im Kontext einer konkreten Anwendung deutlich gemacht werden. Anhand mindestens eines Beispiels wird die Anwendung exemplarisch gezeigt.

Um dies zu verdeutlichen, muss zunächst das Anwendungsbeispiel genannt werden, für das die exemplarische Sicherheitsanforderung spezifiziert werden soll. Die Sicherheitsanforderung wird konventionell beschrieben. Ausgehend davon wird dann die Anwendung der generischen Formulierung durch Instanziierung verdeutlicht.

Im Anhang F ist beispielhaft ein Safety-Pattern abgedruckt, das die im Kapitel 5 hergeleiteten und im Kapitel 6 entwickelten Elemente enthält. Dort findet sich auch das Beispiel einer Erläuterung durch ein Musterbeispiel.

Nach der Herleitung des Safety-Pattern-Ansatzes, wurde in diesem Kapitel gemäß den Vorgaben aus Kapitel 5 das Safety-Pattern-Konzept ausgearbeitet. Die Menge der notwendigen generischen Formulierungen für das Safety-Pattern-Konzept wurde hergeleitet, sowie solche Safety-Pattern, die für die Anwendbarkeit des Verfahrens wichtig sind. Dabei wurde die für die Safety-Pattern-Selektion erforderliche Klassifikation der Safety-Pattern eingeführt. Gemäß der unterschiedlichen Charakteristika der Safety-Pattern und der daraus resultierenden geeigneten generischen Formulierungen in Worten der natürlichen Sprache, wurde die Safety-Pattern-Normsprache eingeführt. Ebenso wurde GNSS als grafische und für Safety-Pattern spezifische Notation eingeführt. Für die Erläuterung von Safety-Pattern wurden geeignete grafische Darstellungen und eine passende Art und Weise zur Beschreibung in natürlicher Sprache und zur Erklärung durch Musterbeispiele entwickelt. Im folgenden Kapitel wird nun auf die werkzeugspezifischen Aspekte des Safety-Pattern-Konzepts (wie z. B. Instanziierung) eingegangen und ein geeignetes Werkzeugkonzept entwickelt.

Inadequate tools are a serious impediment to industrial use of formal methods.

C. Michael Holloway, Ricky W. Butler, Impediments to Industrial Use of Formal Methods. IEEE Computer, 29(4):25-26, April 1996

7 Realisierung eines Web-basierten Unterstützungssystems zur Spezifikation von Sicherheitsanforderungen

Für das in Kapitel 6 eingeführte Safety-Pattern-Konzept wurden adäquate Softwarewerkzeuge entwickelt. Wie bereits in Abschnitt 5.3.9 hergeleitet wurde, ist die Unterstützung durch ein geeignetes Werkzeugkonzept ein unerlässlicher Bestandteil des Safety-Pattern-Konzepts, um der Forderung der einfachen Anwendbarkeit des Verfahrens zur Spezifikation funktionaler Sicherheitsanforderungen nachzukommen. Dieses Werkzeugkonzept wurde ausgearbeitet und prototypisch unter dem Namen SAPIS realisiert (**Sa**fety-**P**attern **I**nstantiation **S**ystem). Es wird in diesem Kapitel vorgestellt. Es umfasst Unterstützungssysteme zur Selektion, Instanziierung und Verwaltung von Sicherheitsanforderungen. Die Interpretation von Safety-Pattern wird durch ein Werkzeug zur grafischen Simulation unterstützt.

7.1 Grundlegende Entwicklungsentscheidungen

Das Unterstützungssystem für das Safety-Pattern-Konzepts muss vier Teilbereiche umfassen: *Datenhaltung für die Safety-Pattern*, *Selektion und Darstellung von Safety-Pattern*, *Instanziierung und Verwaltung von Sicherheitsanforderungen* und *Unterstützung der Interpretation durch grafische Simulation.*

Aufgrund der unterschiedlichen Art der geforderten Funktionalität wurde SAPIS mit einer 3-Tier-Architektur entworfen: Benutzungsoberfläche (GUI – Graphical User Interface), Anwendungslogik und Datenbank. Die Trennung zwischen Kommunikationsschnittstelle mit dem Benutzer, Verarbeitung und Aufbereitung der Daten und Datenhaltung ermöglicht unabhängige Änderungen solange die Schnittstellen zwischen den drei Teilsystemen nicht von den Änderungen betroffen sind.

SAPIS wurde als Webapplikation entwickelt. Dies hat den Vorteil, dass auf den Rechnern der Benutzer keine Installation der Applikation erfolgen muss. Die einzige Voraussetzung ist, dass ein einfacher Web-Browser installiert ist. Somit ist die Anwendung von SAPIS unabhängig vom eingesetzten Betriebssystem. Auch müssen keine Updates von SAPIS bei den Benutzern eingespielt werden. Durch vertraute Browserdarstellungen und die Verwendung bekannter Standard-Bedienelemente des Browsers (z. B. Zurück-Button) wird die einfache Anwendbarkeit von SAPIS unterstützt. Die Realisierung als Webapplikation ist unter der Voraussetzung

möglich, dass Arbeitsplatzrechner in der Industrie heutzutage in der Regel mit dem World Wide Web verbunden sind.

7.2 Speicherung der Safety-Pattern in einer Datenbank

Damit die Safety-Pattern selektiert werden können, müssen sie in einer Datenbasis abgelegt werden. Um die Safety-Pattern dabei gemäß der in Abschnitt 6.1.4 eingeführten Klassifikation systematisch zu ordnen, wurden sie in einer relationalen Datenbank gespeichert, wobei als Datenbanktechnologie MySQL eingesetzt wurde. Das Datenmodell wurde gemäß der Boyce-Codd Normal Form (BCNF) entworfen [Lova03], sodass die Datenbank wenig Redundanz enthält und die Datenpflege einfach ist. BCNF ist die zurzeit am häufigsten verwendete Normalisierungsform [Lova03]. Eine höhere Normalisierungsform ist nicht erforderlich. Sie würde jedoch die Leistungsfähigkeit unnötig einschränken.

Alle nützlichen generischen formalen Formulierungen von Safety-Pattern lassen sich aus den notwendigen ableiten. Deshalb wäre es generell möglich, diese nützlichen Formulierungen nicht manuell in die Datenbank einzutragen, sondern sie durch die Applikationslogik von SAPIS automatisch generieren zu lassen. In Anhang G wird erläutert, dass dies nur für die formalen generischen Formulierungen zweckmäßig und auch hier nur mit Einschränkung sinnvoll ist, da eine Generierung der einfachen Verständlichkeit der generischen Formulierungen entgegenwirkt.

7.3 Beratungssystem für die Selektion und Darstellung von Safety-Pattern

7.3.1 Funktionalität für die Safety-Pattern-Selektion und -Darstellung

Bei der Verwendung des Safety-Pattern-Konzepts zur Spezifikation und Interpretation von Sicherheitsanforderungen müssen die passenden Safety-Pattern selektiert werden. Für die einfache Anwendbarkeit des Verfahrens bei der Selektion ist es wichtig, dass die Werkzeugunterstützung hierfür einfach benutzbar und effizient gestaltet ist. Für die einfache Benutzbarkeit sind Verständlichkeit, Erlernbarkeit und Bedienbarkeit wesentlich. Die Benutzbarkeit wird zudem durch den Aufwand bestimmt, der zur Anwendung erforderlich ist. Die Safety-Pattern-Selektion ist effizient, wenn bei möglichst geringem Zeitaufwand Safety-Pattern ausgewählt und erfasst werden können. Um eine anwendbare Safety-Pattern-Selektion gewährleisten zu können, darf deshalb die Datenbankabfrage nicht manuell durchgeführt werden. Die hierfür erforderlichen Informationen müssen über werkzeuggestützte Dialoge vom Benutzer erfragt werden (Beratungssystem). Die Führung durch Dialoge ist wesentlich für die einfache Erlernbarkeit und Bedienbarkeit der Selektion. Die einfache Anwendbarkeit der Benutzungsoberfläche und der Dialoge wird durch folgende Eigenschaften erreicht (vgl. [Bits03a]):

- *Einheitliche Gestaltung der Benutzungsoberfläche:* Die Benutzungsoberfläche ist die Schnittstelle des Werkzeugs zum Benutzer. Deshalb spielt sie eine sehr wichtige Rolle für die Akzeptanz des Werkzeugs und damit für die Anwendbarkeit des Safety-Pattern-Konzepts. Grundlegend für eine einfach Bedienbarkeit ist eine einheitliche Gestaltung der Benutzungsoberfläche [Lova03]. Zu einem einheitlichen Erscheinungsbild gehören die einheitliche Verwendung von Schriftarten und Farben, die konsistente Verwendung von grafischen Symbolen für Bedienelemente, die Einführung eines werkzeugspezifischen Fenster-Layouts und auch eine einheitliche Terminologie in der Applikation.

- *Integrierte Bedienungsanleitung:* Eine weitere Grundlage für die Anwendbarkeit ist, dass es eine Bedienungsanleitung gibt, in der alle Werkzeug-Funktionen beschrieben sind. Um den Zugriff auf die Anleitung einfach und effizient zu gestalten, wurde die Bedienungsanleitung in die Webapplikation integriert.

- *Führung des Selektionsdialogs durch Fragestellungen:* Der Schritt, die geeigneten Safety-Pattern aufgrund der Charakterisierung der Sicherheitsanforderungen zu wählen, geschieht algorithmisch und muss für eine einfache Anwendbarkeit automatisiert sein. Aufgrund der Charakterisierung muss eine entsprechende Datenbankabfrage werkzeuggestützt erzeugt werden, mit der die Safety-Pattern-Informationen ermittelt und dementsprechend dargestellt werden können. Eine natürliche Benutzerführung wird erreicht, indem es zu jedem Safety-Pattern-Klassifikationskriterium mindestens eine Frage gibt, durch deren Beantwortung die entsprechende Safety-Pattern-Klasse ausgewählt wird. Durch die Frage wird der Benutzer zum Nachdenken über die erforderliche Entscheidung gezielt angeleitet. Durch die Vorgabe alternativer Fragestellungen kann zudem Verständnisproblemen bezüglich der Fragen entgegengewirkt werden. Für die Safety-Pattern-Klasse 2 „Art der Zeitspezifikation“ ist beispielsweise eine geeignete Frage „Soll die Sicherheitsanforderung auch explizite Zeitangaben enthalten?“ Dadurch wird der Benutzer aufgefordert anzugeben, ob die Sicherheitsanforderung ausschließlich implizite Zeitangaben enthält oder auch explizite.

- *Klassifikation durch Auswahl zwischen vorgegebenen Antworten:* Für die einfache Gestaltung der Beantwortung der Frage muss der Benutzer zwischen vorgefertigten Antworten wählen können (Multiple-Choice-Fragen). Dabei steht jede Antwort für eine Safety-Pattern-Klasse. In dem verwendeten Beispiel sind die Antworten:

  A.1„Sicherheitsanforderungen mit ausschließlich impliziten Zeitangaben“

  A.2„Sicherheitsanforderungen mit expliziten Zeitangaben“

- *Strukturierte Safety-Pattern-Darstellung:* Bei der Anzeige eines Safety-Pattern müssen alle Muster-Informationen dargestellt werden (vgl. Anhang H, Abbildung H.1 und Abbildung H.2). Um eine strukturierte Darstellung zu gewährleisten, ist die klare Trennung der verschiedenen Muster-Informationen erforderlich. In SAPIS geschieht diese Aufteilung durch Überschriften.

- *Direkter Zugriff auf Safety-Pattern-Normsprachen-Lexikon:* Für die Interpretation der generischen Formulierungen in der Safety-Pattern-Normsprache, müssen die erforderlichen Fachwörter jederzeit nachgeschlagen werden können. Damit dies parallel zum Selektions- und Instanziierungsprozess geschieht, muss das Lexikon in einem separaten Fenster aufgerufen werden können.

- *Einfache Erlernbarkeit und Effizienz durch benutzerspezifische Dialogformen:* Für die Anwendbarkeit ist jedoch nicht nur entscheidend, dass Ingenieure, die das Safety-Pattern-Konzept bisher kaum verwendet haben, dieses einfach anwenden können (einfache Erlernbarkeit), sondern auch solche effizient arbeiten können, die das Verfahren sehr gut kennen. Deshalb ist ein unterschiedlicher Grad der Führung zu unterscheiden, der sich durch folgende mögliche Dialogarten bestimmt wird:

  1. **Streng geführter Dialog:** Durch diese Dialogform wird die Anwendung von SAPIS zur Safety-Pattern-Selektion auch für Personen ermöglicht, die weder mit dem Safety-Pattern-Konzept noch mit SAPIS vertraut sind [Bits03a]. Bei diesem Dialog findet die Frage-Antwort-Interaktion mit dem Benutzer in einer vorgegebenen Reihenfolge statt [Lova03]. Diese Reihenfolge wird durch einen fest vorgegebenen Klassifikationsbaum vorgegeben. Der Benutzer wird schrittweise geführt und lernt dabei die unterschiedlichen Klassifikationskriterien und Safety-Pattern-Klassen kennen.

  2. **Geführter Dialog:** Im Unterschied zum ersten Dialogmodus werden dem Benutzer statt einer strikten Reihenfolge verschiedene sinnvolle Möglichkeiten gegeben, welches Klassifikationskriterium er als nächstes bearbeiten kann [Bits03a] [Webe04]. Für diese Kriterien ist zu diesem Zweck jeweils die Auswahl aktiviert. Der Dialog wird zum einen durch eine generelle Ordnung bestimmt, welche Klassifikations-Entscheidung vor welcher anderen getroffen werden sollte. Zum anderen wird der Dialog durch das Expertenwissen kontrolliert, welche Safety-Pattern-Klasse welche anderen Klassen ausschließt. Die Klassifikationskriterien, für die bereits eine Entscheidung getroffen wurde, und die Klassen deren Wahl ausgeschlossen ist, werden jeweils ausgeblendet.

  3. **Offener Dialog:** Um eine effiziente Anwendung für erfahrene Benutzer zu ermöglichen, wurde dieser Selektions-Dialog realisiert. Der Benutzer wird nicht gezwungen, eine bestimmte Klassifikations-Reihenfolge einzuhalten. Benutzerfreundlichkeit wird hier in der Weise nachgekommen, dass der Benutzer die Reihenfolge selbst bestimmen kann [Bits03a] [Lova03] [Webe04]. Dadurch ist es ihm möglich, eine Klassifikationsentscheidung zu treffen, durch die implizit auch andere Entscheidungen mitbeschlossen werden. In den anderen Dialogarten wird dies durch die vorgegebene Klassifikations-Reihenfolge verhindert. Deshalb kann bei dieser Dialogart Zeit eingespart werden.

  Alle drei Dialogarten wurden implementiert, um auf die unterschiedlichen Arten von Benutzern adäquat einzugehen und eine adaptive Vermittlung von Expertenwissen zu

erreichen. Bei allen drei Dialogformen wird die Erlernbarkeit und die einfache Verständlichkeit der verschiedenen Klassifikationskriterien und Safety-Pattern-Klassen durch ein Hilfesystem unterstützt. Dies beschreibt nicht nur die Verwendung der Werkzeugfunktionen, sondern ermöglicht auch den direkten Aufruf von Informationen über den Selektionsprozess. So gibt es bei jeder Klassenzuordnung einen Hilfe-Button, über den kontextbezogene Erläuterungen in einem neuen Fenster sofort angezeigt werden können (vgl. Abbildung 7.1).

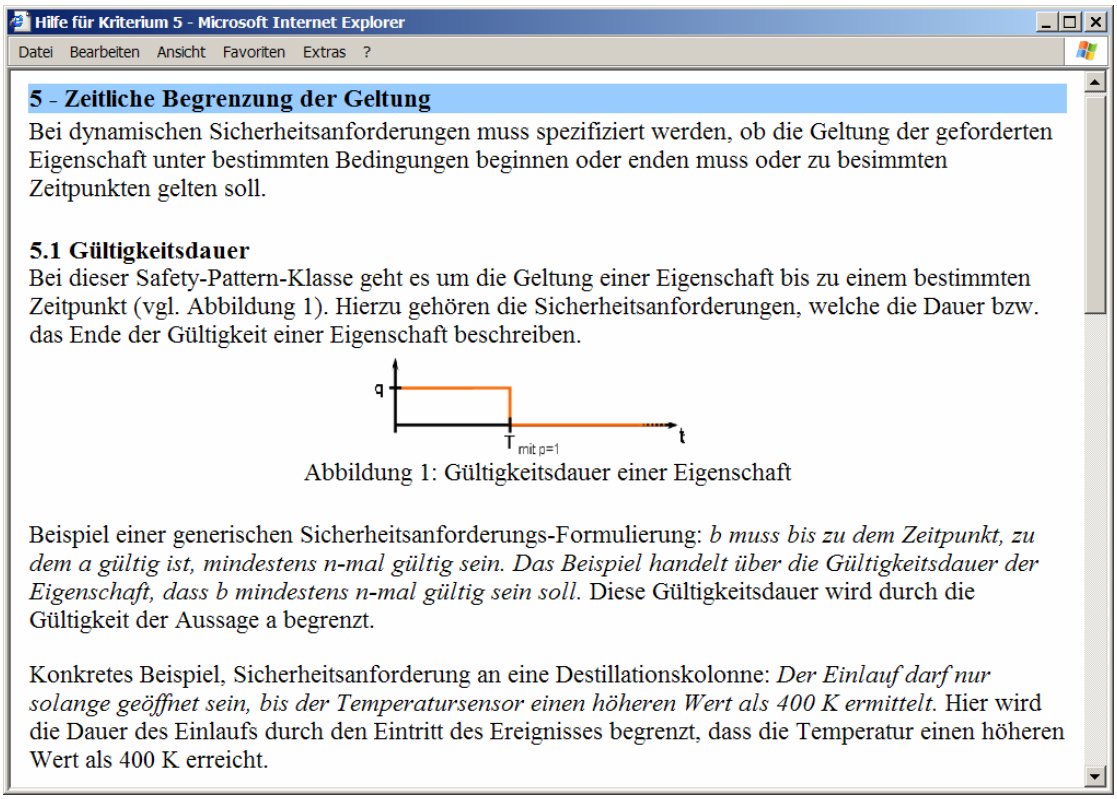


Abbildung 7.1: Direkter Zugriff auf die Erläuterung der Safety-Pattern-Klassifikation

Bei der Auswahl der geeigneten Klasse kann der Benutzer entweder nach dem Auswahlverfahren (vgl. Abbildung 7.2, links) oder dem Ausschlussverfahren vorgehen (vgl. Abbildung 7.2, rechts). Beim Ausschlussverfahren muss der Benutzer jede Klasse ankreuzen, die nicht die passende Klasse ist. Somit muss sich der Benutzer aktiv mit jeder Klasse befassen und wird dabei geschult. Für Benutzer, die mit der Safety-Pattern-Klassifikation vertraut sind, ist es jedoch effizienter, wenn sie lediglich die passende Klasse auswählen.

- *Validierung des Selektionsergebnisses durch Vergleich:* Damit der Benutzer seine Selektions-Entscheidungen validieren kann, müssen Möglichkeiten zur Kontrolle gegeben sein. Dies wird erreicht, indem zu jedem Safety-Pattern Links zu ähnlichen Safety-Pattern angegeben werden (vgl. Bereich „ähnliche Safety-Pattern“ im Anhang H, Abbildung H.2). Dadurch kann sich der Benutzer vergewissern, dass das selektierte Safety-Pattern das gesuchte ist und nicht ein hierzu sehr ähnliches. Zudem können zu einem Safety-Pattern Spezialfälle angegeben werden. Dies sind im Vergleich zur selektierten generischen Formu-

lierung ähnliche Formulierungen. Im Fall des Safety-Pattern von Anhang H, Abbildung H.1 ist ein Spezialfall in Normsprache und allen formalen Sprachen angegeben.

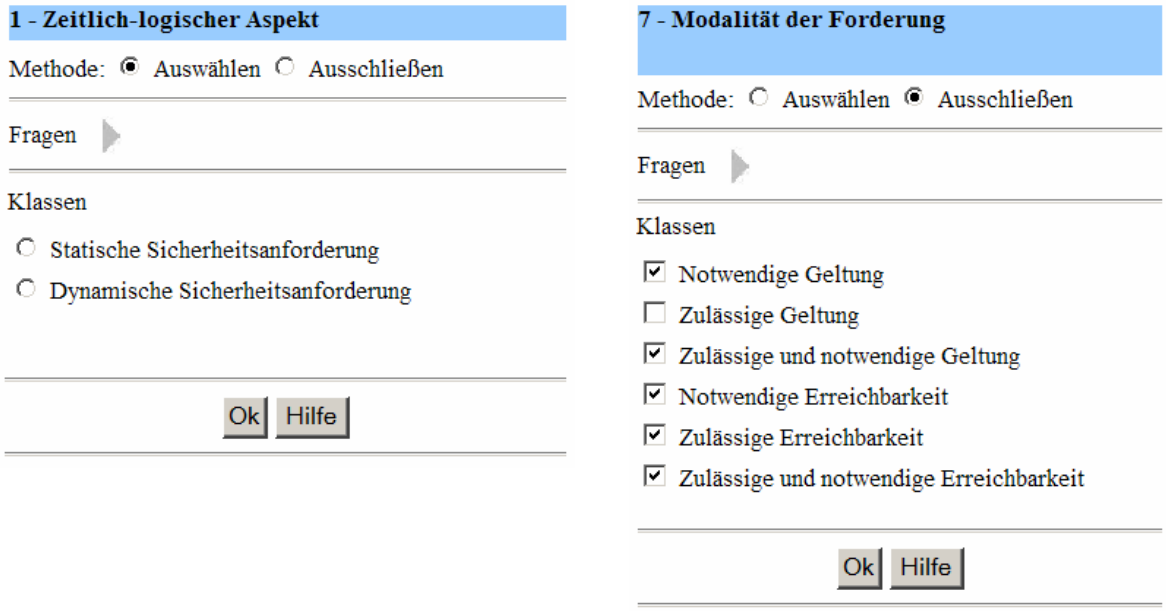


Abbildung 7.2: Klassifikationsdialoge von SAPIS

- *Natürliche Formulierungen durch Formulierungsvarianten:* Um die Natürlichkeit der Formulierung zu unterstützen, ist es erforderlich, verschiedene Formulierungsmöglichkeiten in Normsprache angeben zu können, von denen der Benutzer die auswählen kann, die in dem jeweiligen Fall am geeignetsten ist. In Anhang H, Abbildung H.1 werden verschiedene Formulierungsmöglichkeiten angegeben, die durchnummeriert sind. Für Experten für formale Methoden können auch alternative formale Formulierungen interessant sein. Deshalb ist auch hierfür die Angabe verschiedener Formulierungsmöglichkeiten vorgesehen.

- *Nachschlagen von Safety-Pattern über Suchfunktionen:* Um ein Safety-Pattern nicht nur mithilfe des Selektionsprozesses auswählen und anzeigen zu können, sind zudem Suchfunktionen erforderlich. Diese können genutzt werden, wenn bei einer spezifizierten Sicherheitsanforderung mitangegeben wurde, welches Safety-Pattern für die Spezifikation verwendet wurde. Die Erläuterungen für die verwendete Formulierung können nachgeschlagen werden. Hierfür bietet SAPIS Funktionen, mit denen durch die Angabe des Safety-Pattern-ID-Code (Codierung gemäß der Klassifikation) oder des Safety-Pattern-Namens Safety-Pattern ausgewählt und angezeigt werden können. Dies ist zudem für erfahrene Benutzer des Safety-Pattern-Konzepts wichtig, welche die Klassifikation manuell durchführen wollen, statt den Selektionsprozess werkzeuggestützt auszuführen. Durch die Eingabe des manuell hergeleiteten Safety-Pattern-ID-Code kann das ermittelte Safety-Pattern angezeigt werden.

- *Experten-Suchfunktionen:* Ein anderer Fall, bei dem der schrittweise Selektionsprozess nicht gebraucht wird, ist bei Experten für formale Methoden gegeben, die bestimmte formale Formulierungen bereits im Kopf haben, und sich anzeigen lassen möchten, bei welchen

Safety-Pattern diese Formulierung vorkommt. Dabei ist nicht nur das Safety-Pattern von Interesse, bei dem lediglich diese gesuchte Formulierung vorkommt, sondern auch die, bei welchen die gesuchte eine Teilformulierung ist. Um auf die Kenntnisse der unterschiedlichen Experten einzugehen, ist die Suche in allen verwendeten formalen Sprachen sowie in Normsprache möglich. Abbildung 7.3 zeigt diese Funktionen von SAPIS sowie die zum Nachschlagen von Safety-Pattern.

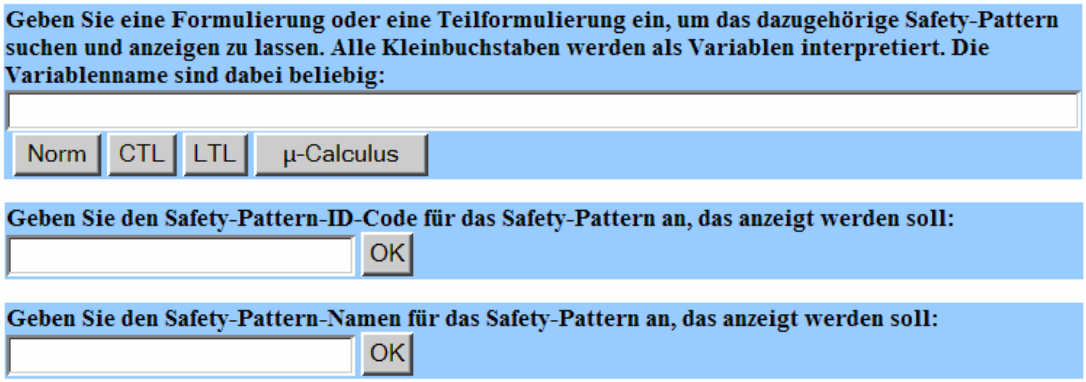


Abbildung 7.3: Funktionen zur Safety-Pattern-Suche

Ein wichtiger Aspekt der Anwendbarkeit ist die Mehrfachverwendung (vgl. Abschnitt 3.2). Hierfür müssen einmal selektierte Safety-Pattern abgespeichert werden können. Auf diese Weise können sie bei Auftreten des gleichen Spezifikationsproblems wiederverwendet werden.

Die Benutzungsoberfläche muss bei SAPIS in der Regel dynamisch gestaltet werden. Beim Selektionsprozess zum Beispiel muss die Darstellung der Benutzungsoberfläche so sein, dass der Benutzer bei der Selektion geführt wird. Zudem muss bei jeder Datenbankabfrage die Benutzungsoberfläche dynamisch aufgebaut werden. Als hierfür geeignete Technologien wurden für den dynamischen Aufbau der Benutzungsoberfläche JSP (**J**ava **S**erver **P**ages) mit HTML (**H**yper**t**ext **M**arkup **L**anguage) und JavaScript verwendet. Für den Datenbankzugriff wurden mit JavaBeans vorgefertigte Softwarekomponenten eingesetzt. Zudem wurde diese Technologie zum Speichern und Auslesen der eingestellten Benutzungsinformationen während eines Selektions- bzw. eines Instanziierungsprozesses verwendet.

7.3.2 Unterstützung der Interpretation von Safety-Pattern durch grafische Simulation

In Abschnitt 6.4.3 wurde bereits ein Simulationskonzept eingeführt, durch das die Interpretation von Safety-Pattern unterstützt wird. Für alle Safety-Pattern wurde die logische Beziehung zwischen der abhängigen von der unabhängigen Safety-Pattern-Variablen bzw. den unabhängigen Safety-Pattern-Variablen implementiert [Jime03]. Dabei wurde das Konzept zur grafischen Darstellung von der Safety-Pattern-Logik getrennt, sodass beides unabhängig voneinander einfach änderbar ist. Für jedes Safety-Pattern wurde eine spezifische Logikrelation implementiert. Alle

Safety-Pattern-Simulationen verwenden jedoch dieselbe Implementierung für die Darstellung. In Anhang E.2 werden weitere Details über die Umsetzung des Simulationskonzepts beschrieben und anhand von Beispielen erläutert.

7.4 Beratungssystem für die Instanziierung

Bei der Instanziierung werden Fehler vermieden, indem der Benutzer zum einen durch eine werkzeuggestützte Führung zu der korrekten Instanziierung angeleitet wird. Zum anderen werden Fehler durch die Überprüfung des spezifizierten Ausdrucks vermieden (vgl. Abschnitt 5.3.5, 5.3.9 und [AGB04]). Zur Unterstützung der unterschiedlichen Personenzielgruppen ist es wichtig, dass dem Benutzer überlassen wird, anhand welcher der generischen Formulierungsarten er die Instanziierung verfolgt. Experten für formale Methoden muss hier die Möglichkeit gegeben werden, die temporallogische Formulierung direkt zu instanziieren. Für Personen, die sich nicht für die temporallogische Formulierung interessieren, ist es sinnvoll, dass sie die verständliche generische Spezifikation in Normsprache oder in grafischer Notation verwenden. Das Werkzeug wurde entsprechend in der Weise realisiert, dass egal welche generische Formulierungsart bei der Instanziierung verwendet wird, die anderen Arten automatisch auch instanziiert werden [Hoss03].

Für die Anleitung zur korrekten Spezifikation sind folgende Werkzeugeigenschaften sinnvoll [AGB04] [Hoss03]:

- Der Benutzer wird nacheinander aufgefordert, sämtliche Safety-Pattern-Variablen durch einzelne System-Variablen oder logische Ausdrücke zu ersetzen.
- Die verschiedenen Instanziierungsmöglichkeiten durch einzelne System-Variablen, aussagenlogische Kombinationen solcher Variablen, Werteänderungen von System-Variablen (vgl. Abschnitt 5.3.5) oder durch andere Ausdrücke, die mithilfe von Safety-Pattern spezifiziert werden, werden dem Benutzer deutlich gemacht.
- Für die Bildung aussagenlogischer Ausdrücke werden dem Benutzer sämtliche mögliche Operatoren aufgelistet, vgl. Abbildung 7.4. Diese können jeweils über Buttons ausgewählt oder für geübte Benutzer über die Tastatur eingegeben werden.
- Über ständig aufrufbare Hilfefunktionen werden die einzelnen Schritte erläutert und Hintergrundinformationen gegeben wie z. B. die Beschreibung der Bedeutung der verwendbaren Operatoren (Hilfesystem).

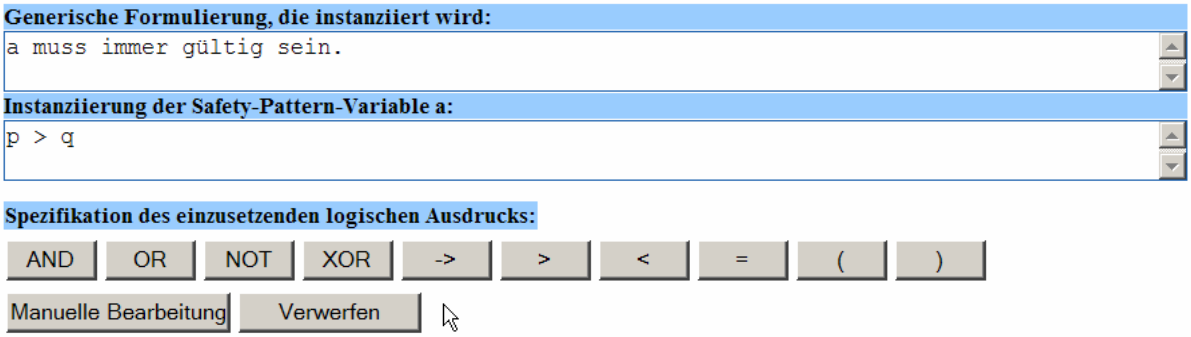


Abbildung 7.4: Anleitung zur Spezifikation aussagenlogischer Ausdrücke in SAPIS

Für die Überprüfung des spezifizierten Ausdrucks sind folgende Werkzeugeigenschaften erforderlich [AGB04] [Hoss03]:

- Die korrekte Verwendung von Operatoren muss kontrolliert werden. Zwei Integer-Variablen können z. B. nicht mit aussagenlogischen Operatoren verknüpft werden, sondern nur durch Vergleichsoperatoren. Je nach Typ der verwendeten Systemvariablen dürfen deshalb nur die Operatoren anwendbar sein, deren Verwendung erlaubt ist.

- Um diese Werkzeugeigenschaft zu ermöglichen, deklariert der Benutzer die Systemvariablen, die bei der Instanziierung verwendet werden sollen. Hierfür muss es eine entsprechende Werkzeugfunktionalität zum Anlegen, Ändern und Löschen von Systemvariablen geben. Deklarierte Variablen erscheinen dann bei den Instanziierungsschritten in einer Liste, aus der die für die Instanziierung verwendbaren Systemvariablen gewählt werden können.

- Auf der Grundlage deklarierter Variablen wird dann die Syntax spezifizierter Ausdrücke durch Vergleich mit einem Syntaxbaum überprüft.

- Für die Komposition mehrerer generischer Formulierungen ist es sinnvoll, dass der Benutzer zum einen geführt wird und zum anderen überprüfen kann, ob durch die Komposition Widersprüche entstehen und ob die Bedeutung des Gesamtausdrucks seinen Vorstellungen entspricht. Diese Überprüfungen sind durch die Simulation des Kompositionsausdrucks möglich. Auf dieses Thema wird im Anhang E.3 genauer eingegangen.

Um einerseits die Verwaltung von Variablen für die Entwicklung in Teams zu unterstützen, können die Variablen auf dem Webserver abgelegt werden. Dadurch können die Teams auch örtlich verteilt arbeiten. Andererseits können die Variablen aber auch entwicklerspezifisch, d. h. lokal auf dem Arbeitsplatzrechner, verwaltet werden.

Als Format für die Speicherung wurde ein XML[18]-Format erstellt [Hoss03], sodass mithilfe von Transformationsvorschriften in XSLT[19] ein einfacher Datenaustausch mit Modellierungswerk-

[18] XML (**E**xtensible **M**arkup **L**anguage) ist eine Metasprache, die es erlaubt eigene Dokumentmarkierungselemente zu schaffen und somit entsprechend frei konfigurierbar ist [EcCa02].

zeugen und Model-Checking-Werkzeugen möglich ist. Hierfür wurden die XML-Tags[20] `Systemvariable` und `Var` mit den Attributen `Name`, `Constant` und `Value` eingeführt. Beispiel:

```
<sytemvariable>
    <var  name="train.maximumvelocity" type="integer"
          initialvalue="50" constant="yes" />
</sytemvariable>
```

In dem Beispiel wird die Systemvariable `maximumvelocity`, die dem Teilsystem `train` zugeordnet und vom Typ `integer` ist, als Konstante mit dem Wert `50` definiert.

7.5 Verwaltung von instanziierten Sicherheitsanforderungen

Das Ergebnis der Instanziierung muss so festgehalten werden, dass die Sicherheitsanforderung zu einem späteren Zeitpunkt wieder schnell auffindbar ist. Orientiert sich die Speicherung der Sicherheitsanforderungen an dem System, für das sie gelten, geschieht der Aufruf gespeicherter Sicherheitsanforderungen intuitiv und damit einfach und benutzerfreundlich. Dabei können Sicherheitsanforderungen nicht nur bestimmten Automatisierungssystemen, sondern auch speziellen Teilsystemen und Systemkomponenten hierarchisch zugeordnet werden. SAPIS umfasst deshalb Funktionen [Hoss03] zur Definition von Automatisierungssystemen mit dazugehörigen Teilsystemen. Diesen wiederum können rekursiv Systemkomponenten zugeordnet werden.

Als Format für die Speicherung wurde ein XML-Format erstellt [Hoss03] und verwendet. Dadurch können mithilfe von Transformationsvorschriften in XSLT Sicherheitsanforderungen einfach in für Model-Checking-Werkzeuge spezifische Datenformate transformiert werden. Mit JavaBeans wurden für die Dateizugriffe vorgefertigte Softwarekomponenten eingesetzt.

7.6 Integration der Teilsysteme von SAPIS

Entsprechend dem in diesem Kapitel hergeleiteten Werkzeugkonzept zeigt Abbildung 7.5 zusammenfassend die Systemarchitektur von SAPIS, die alle Teilsysteme integriert. Für die Darstellung wurde die Subsystem-Sichtweise in der Klassendiagramm-Notation der UML 2.0 gewählt, vgl. [Doug04, S. 485 ff.]. Mit dem verwendeten Stereotyp «`subsystem`» werden dabei System-Untereinheiten und mit dem Stereotyp «`system`» das Gesamtsystem gekennzeichnet.

---

[19] In XSLT (**Ex**tensible **S**tylesheet **L**anguage **T**ransformation) werden Transformationsvorschriften definiert. XSLT-Prozessoren erlauben auf der Grundlage von XSLT-Spezifikationen eine einfache und standardisierte Transformation von XML Dokumenten in andere Formate.

[20] „Tags“ sind eine definierte Folge von Zeichen, die den Beginn und das Ende eines logischen Abschnitts oder eines logischen Elements in einem XML-Dokument repräsentieren.

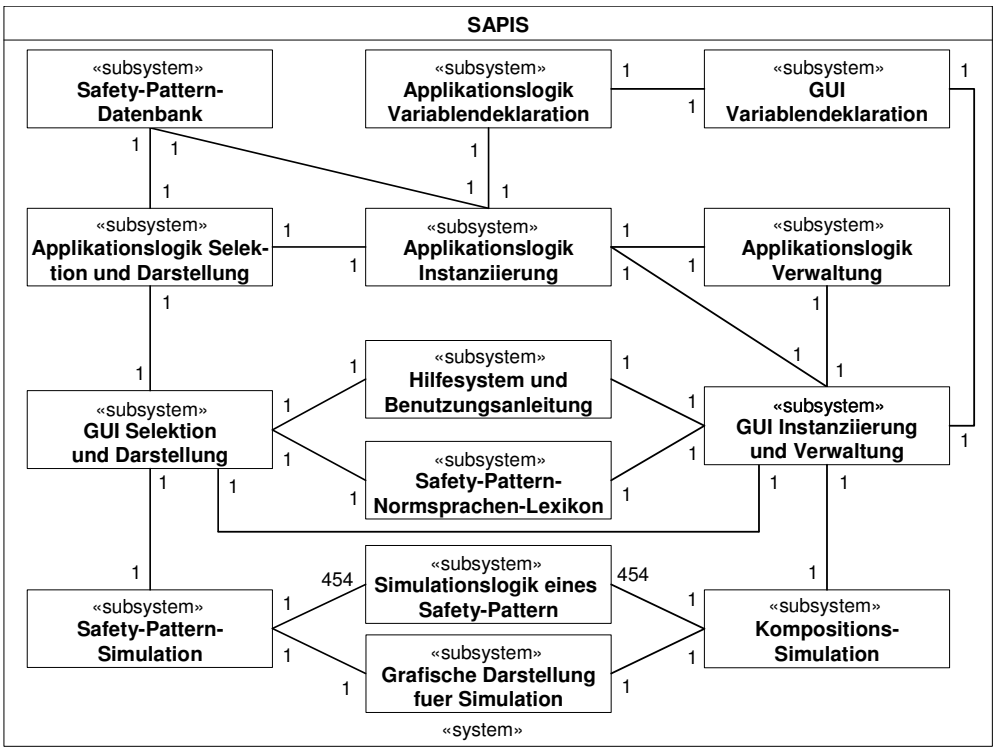


Abbildung 7.5: Systemarchitektur von SAPIS

SAPIS startet mit dem Teilsystem *GUI Selektion und Darstellung*. Im Programmfenster befindet sich immer das Programm-Menü, über das die gewünschten Hauptfunktionalitäten jederzeit direkt angewählt werden können. Darüber hinaus wird der Benutzer für den Aufruf weiterer Funktionalitäten des Werkzeugs kontextabhängig geführt, d. h., die Teilsysteme sind untereinander verlinkt. Zum Beispiel wird dem Benutzer nach dem Abschluss der Selektion eines Safety-Pattern im Hauptfenster die Option angeboten, dieses Safety-Pattern zu instanziieren. Wählt der Benutzer diese Option, so wird das entsprechende Teilsystem aufgerufen.

In diesem Kapitel wurde das Werkzeugkonzept für das Safety-Pattern-Verfahren hergeleitet und beschrieben. Durch Werkzeugunterstützung und die Automatisierung algorithmischer Vorgänge wird die Effizienz der Anwendung des Safety-Pattern-Konzepts entscheidend erhöht. Durch die genaue Anleitung für die Anwendung des Verfahrens wird eine einfache Erlernbarkeit erreicht. Die auf den Kenntnisstand unterschiedlicher Personengruppen ausgelegte Werkzeugführung ist ein wesentlicher Faktor für die Anwenderakzeptanz. SAPIS bietet für die Unterstützung des Verständnisses von Safety-Pattern multimediale Elemente (textuelle und grafische Beschreibungen sowie interaktive Darstellungen durch dynamisch gestaltete Benutzungsoberflächen und Simulation). Durch das Speichern und Verwalten selektierter Safety-Pattern und spezifizierter Sicherheitsanforderungen wird Mehrfachverwendung gewährleistet. Durch die werkzeuggestützte interaktive Führung und Kontrolle wird die Handhabbarkeit des Verfahrens erhöht. Die Kopplung von Werkzeugunterstützung und Dokumentation ermöglicht eine selektive Dokumentations-Präsentation, die den Gebrauch individuell gemäß der Kenntnis des Safety-Pattern-Konzepts ermöglicht. Somit entfällt aufwendiges Suchen in Handbüchern, wodurch der Spezifikationsaufwand reduziert und die Anwendbarkeit unterstützt wird.

Lang ist der Weg durch Lehren (der Theorie), kurz und erfolgreich durch Beispiele (der Praxis).

Seneca, ep., 6, 5

8 Anwendung des Safety-Pattern-Konzepts für einen eingleisigen Bahnübergang im Funk-Fahr-Betrieb

Die Verwendung des Safety-Pattern-Konzepts wird in diesem Kapitel anhand eines sicherheitskritischen Anwendungsbeispiels aus der Verkehrsleittechnik demonstriert. Als Fallstudie dient ein eingleisiger Bahnübergang im Funk-Fahr-Betrieb. Die Sicherheitsanforderungen wurden formal spezifiziert. Gegenüber dieser Spezifikation wurde das betriebliche Konzept der Anwendung formal verifiziert. Die Ergebnisse und der Nutzen des Safety-Pattern-Konzepts werden anhand eines Modellprozesses demonstriert.

Bei der Anwendung des Safety-Pattern-Konzepts sollen im Wesentlichen zwei Aspekte verdeutlicht werden. Zum einen wurden die Sicherheitsanforderungen der Fallstudie mithilfe von Safety-Pattern im Kontext des Steuerungsmodells der Fallstudie spezifiziert. Dadurch soll die Formalisierbarkeit der Sicherheitsanforderungen mithilfe des Safety-Pattern-Konzepts demonstriert werden. Zum anderen wurde die Korrektheit der Formalisierung anhand der Prüfung des sicheren Verhaltens des Modellprozesses der Fallstudie erprobt.

Abschließend werden in diesem Kapitel die Ergebnisse dargestellt und bewertet, die mit dem Safety-Pattern-Konzept erreicht werden.

8.1 Anforderung an eine geeignete Anwendung zur Erprobung und Demonstration

Um den Nutzen des Safety-Pattern-Konzepts demonstrieren zu können, ist eine automatisierungstechnische Anwendung erforderlich, die hierfür verschiedenen Anforderungen gerecht wird. Die Anwendung soll repräsentativ für die industrielle Praxis sein. Deshalb soll sie auf der Spezifikation eines realen Automatisierungssystems mit Sicherheitsverantwortung basieren. An die Anwendung müssen deshalb funktionale Sicherheitsanforderungen gestellt werden, deren Korrektheit im Steuerungsmodell der Anwendung erfüllt sein muss.

Es sollte eine verteilte Anwendung sein, für welche Sicherheitsanforderungen über mehrere Teilsystemen formuliert werden müssen. Dadurch können komplexe Instanziierungsprobleme veranschaulicht werden. Um den Nutzen des Safety-Pattern-Konzepts adäquat demonstrieren zu

können, muss die Anwendung dynamische Sicherheitsanforderungen[21] besitzen und nicht nur solche, bei denen die geforderte Eigenschaft statisch in allen Systemzuständen gilt. Für die Anwendung müssen zudem komplexe Sicherheitsanforderungen erforderlich sein, anhand derer die Komposition von Safety-Pattern demonstriert werden kann.

Die Anwendung „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb (FFB)“ erfüllt diese Anforderungen. Die Bahnübergangsproblematik ist ein von Informatikern häufig verwendetes Fallbeispiel [Jans99a], das jedoch durch Abstraktionen häufig stark vereinfacht wird. Durch die Betrachtung eines Bahnübergangs, in dem von der Deutschen Bahn AG konzipierten Betriebsverfahren „Funk-Fahr-Betrieb“, wurde eine Spezifikationsaufgabe von realistischer Komplexität bei überschaubarem Umfang gewählt. Am Institut für Verkehrssicherheit und Automatisierungstechnik der TU Braunschweig wurde die Fallstudie „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb“ im Rahmen des DFG-Schwerpunktprogramms „Integration von Techniken der Softwarespezifikation für ingenieurwissenschaftliche Anwendungen“ entwickelt, s. [JaSc00], [Jans99a], [Jans99b], [Jans99c], [Gaye99a], [Gaye99b], [Gaye99c], [HPSS04]. Diese dient für die Demonstration und Erprobung des Safety-Pattern-Konzepts als Vorbild. Die Fallstudie wurde auf der Grundlage der Lastenhefte der Deutschen Bahn AG zum Funk-Fahr-Betrieb [Adtr97] erstellt.

8.2 Modellprozess „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb“

An Bahnübergängen kreuzen sich Verkehrswege des Schienen- und des Straßenverkehrs höhengleich. Zur Vermeidung von Unfällen ist die technische Sicherung von Bahnübergängen erforderlich. Diese besteht im Wesentlichen in der Sperrung des Bahnübergangs für den Straßenverkehr bei der Befahrung durch Schienenfahrzeuge. Erfolgt keine vollständige Sperrung, darf der Bahnübergang nicht durch Schienenfahrzeuge befahren werden.

Im Funk-Fahr-Betrieb werden Fahrwegelemente wie Bahnübergänge nicht von einem Stellwerk sondern vom Eisenbahnfahrzeug, welches den Bahnübergang befahren soll, direkt per Funk angesteuert. Die Kontrolle der sicheren Funktionalität der Fahrwegelemente erfolgt vom Eisenbahnfahrzeug aus per Funk. Die Fahrwegelemente nehmen ihre Steuerung und Überwachung selbst vor (Selbstdiagnose), indem sie sich auf Anforderung hin sichern und ihren Status melden.

Am IAS wurde ein Modellprozess „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb“ entwickelt (vgl. [Herm04], [Stec03], [Zimm02]). Anhand dieses Modellprozesses wird die Anwendung des Safety-Pattern-Konzepts demonstriert.

---

[21] Safety-Pattern-Klasse, zu der solche Sicherheitsanforderungen gehören, bei denen es um die Geltung einer Eigenschaft in bestimmten Zuständen des Zustandsraums des Systemmodells geht. Die geforderte Eigenschaft gilt also nicht statisch in allen Systemzuständen, sondern besitzt eine dynamische Geltung.

In der ausgewählten Fallstudie wird ein eingleisiger Bahnübergang behandelt, der im Funk-Fahr-Betrieb von einem Eisenbahnfahrzeug befahren werden soll. Die Fallstudie enthält somit die drei Teilsysteme Bahnübergang, Eisenbahnfahrzeug und FFB-Zentrale (s. Abbildung 8.1). Das Eisenbahnfahrzeug darf nur dann den Fahrweg befahren, auf der sich der Bahnübergang befindet, wenn es von der FFB-Zentrale über Funk eine Freigabe erhält (Fahrwegzuweisung). Beim Befahren des Fahrwegs muss zur Gewährleistung der Sicherheit der Bahnübergang zur geforderten Zeit vollständig gesichert sein. Ist dies nicht der Fall, muss das Eisenbahnfahrzeug vor dem Bahnübergang zum Halt kommen. Alle Systemkomponenten des FFB kommunizieren miteinander und es ergibt sich die in Abbildung 8.1 gezeigte Kommunikationsstruktur.

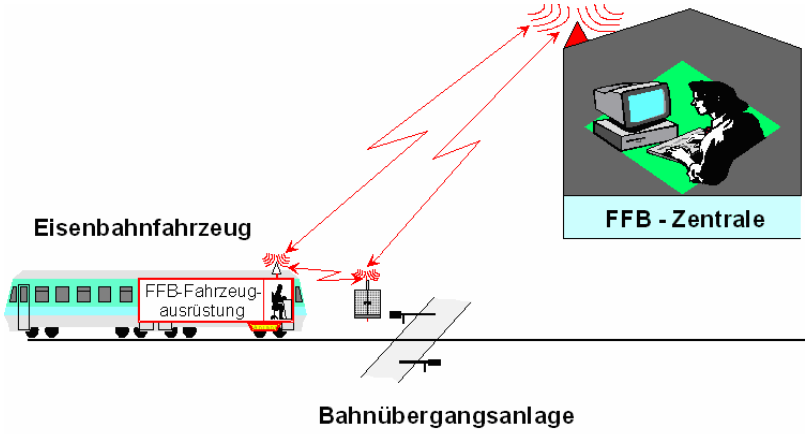


Abbildung 8.1: Systemkomponenten der Fallstudie

8.2.1 Beschreibung des betrieblichen Konzepts

Regulärer betrieblicher Ablauf

Zu Beginn eines betrieblichen Ablaufs sendet das Eisenbahnfahrzeug eine Fahrweganforderung an die FFB-Zentrale. Der Fahrwegrechner der FFB-Zentrale überprüft, ob der angeforderte Fahrweg frei von Eisenbahnfahrzeugen und ob sich der Bahnübergang in einem störungsfreien Zustand befindet. Je nach Ergebnis wird der Fahrweg dem anfordernden Eisenbahnfahrzeug zugewiesen. Der Ort unmittelbar vor dem Bahnübergang des zugewiesenen Fahrwegs gilt nun als potenzieller Gefahrenpunkt. Dieser ist ein bedingter Geschwindigkeits-Zielpunkt mit der Geschwindigkeit v = 0 des Eisenbahnfahrzeugs. Dieser Geschwindigkeitszielpunkt besteht nur solange, wie keine entsprechende Statusmeldung darüber vorliegt, dass der Bahnübergang gesichert ist. Rechtzeitig vor Erreichen dieses Punkts muss das Eisenbahnfahrzeug per Funk im Bahnübergang den Sicherungsvorgang initiieren (Einschaltbefehl). Der Erhalt der Funkmeldung wird vom Teilsystem Bahnübergang quittiert. Den Auftrag zur Sicherung erledigt das Teilsystem Bahnübergang selbstständig. Nach der für die vollständige Sicherung des Bahnübergangs erforderlichen Zeit fordert das Eisenbahnfahrzeug per Funk die Statusmeldung an (Statusabfrage). Wurde der Sicherungsvorgang des Bahnübergangs erfolgreich durchgeführt,

erhält das Eisenbahnfahrzeug daraufhin die Sicherungsmeldung. Das Eisenbahnfahrzeug darf den Bahnübergang passieren. Nach Befahren des Fahrwegelements wird dieses nach Aktivierung der Ausschaltsensorik im Gleis anschließend wieder freigegeben.

Beim Sicherungsvorgang des Bahnübergangs wird nach Erhalt des Einschaltbefehls die Lichtzeichenanlage auf gelb geschaltet, um dem Straßenverkehr die Sicherung des Bahnübergangs anzukündigen. Vor dem Schließen der Schranken wird die Lichtzeichenanlage auf Rot geschaltet. Dadurch soll das Passieren des Bahnübergangs während des Schließens der Schranken vermieden werden. Nach Ablauf der Rotzeit werden die Schranken geschlossen. Der Bahnübergang gilt als gesichert, sobald die Schranken vollständig geschlossen sind.

Betriebliche Abläufe mit Fehlfunktionen

Das beschriebene Szenario bezieht sich auf den regulären, störungsfreien Betriebsfall. Sollte eines der beschriebenen Funktelegramme den Empfänger nicht erreichen oder sollten Komponenten der Bahnübergangssicherungsanlage defekt sein, wird der Bahnübergang manuell gesichert. Die Eisenbahnfahrzeugsteuerung bringt das Eisenbahnfahrzeug vor dem Bahnübergang zum Halt, sodass der Triebfahrzeugführer die manuelle Sicherung des Bahnübergangs vornehmen kann. Erhält ein Eisenbahnfahrzeug die Rückmeldung über die erfolgte Sicherung des Bahnübergangs nicht, bevor es sich auf Bremswegabstand dem Bahnübergang genähert hat, muss eine Zwangsbremsung eingeleitet werden. Ausfälle in der Bahnübergangssicherung müssen an die FFB-Zentrale gemeldet werden. Bis zur Behebung der Ausfälle darf der Fahrweg, auf dem sich der Bahnübergang befindet, nicht mehr befahren werden.

Fällt das Rotlicht aus, bevor die Rotzeit abgelaufen ist, wird der Sicherungsvorgang abgebrochen und der Bahnübergang befindet sich in einem Störungszustand. Bei Ausfall des Gelblichts wird hingegen der Sicherungsvorgang fortgesetzt und die Rotzeit wird entsprechend verlängert. Bei jedem Ausfall muss jedoch eine Störungsmeldung an die FFB-Zentrale versandt werden.

Ist der Bahnübergang zu lange geschlossen, besteht die Gefahr, dass der Bahnübergang trotz geschlossener Schranken von Verkehrsteilnehmern wie z. B. Fußgängern überquert wird. Deshalb wird nach dem Ablauf der sog. Sperrzeit eine Funkmeldung an die FFB-Zentrale gesandt, dass die zulässige Sperrzeit überschritten wurde. In der FFB-Zentrale muss der Fahrdienstleiter ggf. eine Aufhebung der Sicherung des Bahnübergangs per Funk anweisen.

8.2.2 Aufbau des Modellprozesses

Das beschriebene Automatisierungssystem wurde als Modellprozess umgesetzt, um die korrekte Formalisierung der Sicherheitsanforderungen zu veranschaulichen. Erfüllt das Steuerungsmodell die korrekt formalisierten Sicherheitsanforderungen, müssen im Betrieb des Modellprozesses die Sicherheitsanforderungen immer eingehalten werden. Dabei wird vorausgesetzt, dass die Steuerungsmodelle für die einzelnen Teilsysteme korrekt umgesetzt werden.

Der Modellprozess wurde als Gleisoval der Spurbreite G der Firma LGB (Lehmann Groß Bahn) und mit Modellen eines Eisenbahnfahrzeugs und eines Bahnübergangs aufgebaut. Um das Verhalten des Modellprozesses in den sicherheitsrelevanten Situationen erproben zu können, wurden bei der Entwicklung des Modellprozesses Experimentierfunktionen bereitgestellt, durch die sicherheitsbezogene Ausfälle verursacht werden können. Für die drei Komponenten des technischen Prozesses *FFB-Zentrale*, *Bahnübergang* und *Eisenbahnfahrzeug* wurde jeweils ein Steuerungs-PC installiert. Dabei werden für die FFB-Zentrale ein herkömmlicher Desktop-PC und für die beiden anderen Komponenten Embedded-PCs verwendet. An diese PCs wurden I/O-Schnittstellenkarten angeschlossen, um über die Eingänge Sensorwerte der Anlage zu erfassen und über die Ausgänge Einfluss auf die Anlage zu nehmen (s. Abbildung 8.2). Die Ansteuerung des Bahnübergangs kann zudem alternativ über einen Mikrocontroller erfolgen. Jede der drei Systemkomponenten besitzt ihr eigenes Funkmodul zum Empfangen und Senden von Steuerungsinformationen im betrieblichen Ablauf. Technische Details der Realisierung sind in [Zimm02], [Steck03], [Faud04] und [Herr04] beschrieben.

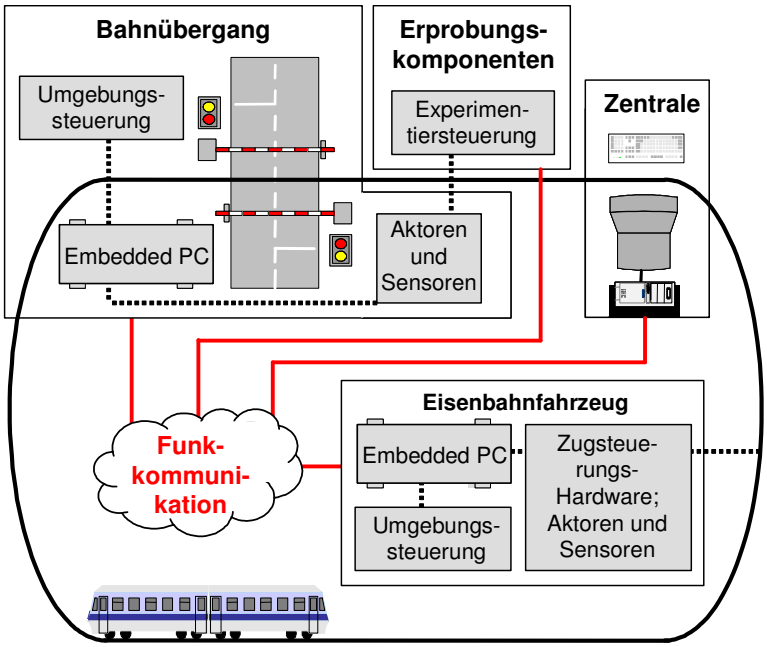


Abbildung 8.2: Systemübersicht des Modellprozesses

Der Modellprozess besitzt 2 Arten von Benutzungsschnittstellen:

1. **Umgebungssteuerung:** Die Umgebungssteuerung beinhaltet Mensch-Maschine-Interaktionen im betrieblichen Ablauf. Dieses sind Interaktionen des Fahrdienstleiters (Teilsystem FFB-Zentrale) und des Triebfahrzeugführers (Teilsystem Eisenbahnfahrzeug).

2. **Experimentiersteuerung:** Mithilfe der Experimentiersteuerung lassen sich sicherheitskritische Situationen herbeiführen, welche durch einzelne Fehlfunktionen vorgegeben werden. Durch die Vorgabe von Störungen wie Lichtzeichenstörungen, Schrankenstörungen,

Kommunikationsstörungen werden sicherheitskritische Situationen initiiert, unter denen das Prüfen der Steuerungssoftware stattfindet. Die Experimentierfunktionen werden durch das Umlegen dafür vorgesehener Schalter aktiviert.

Für den Zugriff auf die Sensoren und Aktoren des technischen Prozesses wurden in [Zimm02], [Steck03] und [Herr04] entsprechende Softwarebibliotheken entwickelt. Zudem wurden in [Zimm02] und [Teuf03] eine Visualisierungsfunktionalität bereitgestellt, die eine Fernvisualisierung ermöglicht. Insgesamt ergeben sich die in Abbildung 8.3 gezeigten Systemschnittstellen im Modellprozess.

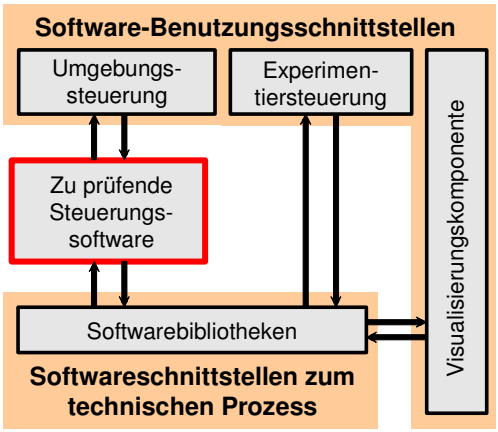


Abbildung 8.3: Schnittstellen des Modellprozesses

8.3 Steuerungsmodell

Auf der Grundlage des UML-Systemmodells, das im DFG-Projekt SafeRail für die Fallstudie „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb“ maßgeblich am Institut für Eisenbahnwesen und Verkehrssicherung der TU Braunschweig erstellt wurde (vgl. [ABG04], [ArGa00]), ist ein Steuerungsmodell für den IAS-Modellprozess mit UML-Klassen- und Zustandsdiagrammen entwickelt worden [Faud04] [Link03]. Das Modell wurde mit dem UML-Werkzeug *Software through Pictures* (StP) der Firma Aonix spezifiziert, das für die Modellierung Editoren für alle Diagrammtypen der UML zur Verfügung stellt. Auf diese Weise war eine Verarbeitung des Modells zur Codegenerierung und formalen Verifikation möglich, wie es in den folgenden Abschnitten noch beschrieben wird.

8.4 Spezifikation der Sicherheitsanforderungen

Insgesamt wurden 51 modellspezifische Sicherheitsanforderungen für den Modellprozess spezifiziert [Faud04]. Diese wurden mithilfe von Safety-Pattern formalisiert. Im Folgenden wird exemplarisch die Spezifikation durch Safety-Pattern-Selektion und –Instanziierung gezeigt.

Konventionelle globale Formulierung der Sicherheitsanforderung [GaPa99]**:** *Bei einer Zeitüberschreitung kann die Sicherung eines Bahnübergangs nur von der FFB-Zentrale aufgehoben werden.*

Wird diese Sicherheitsanforderung im Kontext des konkreten Systems formuliert, folgt daraus die nachstehende systemspezifische Sicherheitsanforderung, vgl. Abschnitt 2.2.4.

Konventionelle systemspezifische Formulierung der Sicherheitsanforderung: *Nach dem Auftreten einer Zeitüberschreitung*[22] *dürfen die Schrankenanlage und Lichtzeichenanlage eines Bahnübergangs jeweils bis zum Eintreten des Ausschaltbefehls von der FFB-Zentrale nicht ausgeschaltet werden.*

Im Kontext des UML-Modells der Systemfunktionen folgt die nachstehende modellspezifische Sicherheitsanforderung, vgl. Abschnitt 2.2.4.

Konventionelle modellspezifische Formulierung der Sicherheitsanforderung: *Nach dem Auftreten von* `Bahnuebergang.sendeUeberschreitungsmeldung` *müssen* `(Schrankenanlage.SchA_ausschalten = 0)` *und* `(Lichtzeichenanlage.LzA_ausschalten = 0)` *jeweils solange gelten, bis* `Bahnuebergang.getAusschaltbefehl` *eintritt.*

Die Eigenschaft, dass in der Systemkomponente *Schrankenanlage* `SchA_ausschalten = 0` und in der Systemkomponente *Lichtzeichenanlage* `LzA_ausschalten = 0` ist, muss unter bestimmten Umständen gelten. Zum einen muss die Eigenschaft ab dem Zeitpunkt gültig sein, zu dem eine Zeitüberschreitungsmeldung vom Teilsystem *Bahnübergang* erkannt wurde (`sendeUeberschreitungsmeldung`). Zum anderen ist dann die Gültigkeit dieser Eigenschaft bis zu dem Zeitpunkt erforderlich, zu dem das Teilsystem *Bahnübergang* per Funk den Befehl zur Ausschaltung, d. h. Aufhebung der Sicherung (`getAusschaltbefehl`), erhält.

Die Eigenschaft besteht aus zwei Teilen und kann einfach mit einem logischen „und“ formalisiert werden: `(Schrankenanlage.SchA_ausschalten = 0) and (Lichtzeichenanlage.LzA_ausschalten = 0)`. Um die Gültigkeit der Eigenschaft in Abhängigkeit von der beschriebenen Bedingung auszudrücken, muss das geeignete Safety-Pattern und damit die geeignete formale Spezifikation in CTL ermittelt werden. Hierfür muss die Sicherheitsanforderung mithilfe von SAPIS den passenden Safety-Pattern-Klassen zugewiesen werden. Im Unterschied zur Benutzerführung in SAPIS werden hier aus Platzgründen nicht alle Entscheidungsmöglichkeiten, sondern nur die jeweils relevante beschrieben. Zudem werden exemplarisch nur zwei der insgesamt acht Selektionsschritte gezeigt. Die vollständige Beschreibung der Selektionsschritte ist in Anhang I zu finden.

---

[22] Zeitüberschreitung ist die Zeit $t_{ZÜ}$, „nach der ein gesicherter Bahnübergang durch Inakzeptanz der Straßenverkehrsteilnehmer wegen langer Schließzeit als nicht gesichert gilt“ [Gaye99b].

3. Selektionsschritt:

Kriterium: **Art der Zeitspezifikation**

Klassifikationsfrage: Soll die Sicherheitsanforderung auch explizite Zeitangaben enthalten?

Diskussion: Nein, es soll keine zeitbehaftete Aussage in Abhängigkeit zu einer Systemuhr gemacht werden. Infolgedessen handelt es sich um eine *Sicherheitsanforderung mit ausschließlich impliziten Zeitangaben*.

4. Selektionsschritt:

Kriterium: **Zeitliche Begrenzung der Geltung**

Klassifikationsfrage: Welche zeitliche Begrenzung der Geltung einer Eigenschaft enthält die Sicherheitsanforderung?

Diskussion: Es soll ausgedrückt werden, ab welcher Zeit die geforderte Eigenschaft `((SchA_ausschalten = 0) and (LzA_ausschalten = 0))` gelten soll und bis wann sie gelten muss. Folglich muss die Sicherheitsanforderung der Safety-Pattern-Klasse *Gültigkeitsbeginn und -dauer* zugewiesen werden.

Auf der Grundlage der Selektion wird von SAPIS das folgende Safety-Pattern mit den dazugehörigen Erläuterungen identifiziert:

Generische Formulierung in der Safety-Pattern-Normsprache:

Wenn *a* gültig ist, so muss von nun an *b* permanent gelten, bevor *c* gültig ist.

Generische Formulierung in formaler Sprache (CTL):

$AG\ (a \rightarrow A(b\ W\ c))$

Spezifikation der Sicherheitsanforderung in der Safety-Pattern-Normsprache:

Wenn `Bahnuebergang.sendeUeberschreitungsmeldung` gültig ist, so muss von nun an `((Schrankenanlage.SchA_ausschalten = 0 and Lichtzeichenanlage.LzA_ausschalten = 0)` permanent gelten, bevor `Bahnuebergang.Status_Ausschaltbefehl` gültig ist.

Spezifikation der Sicherheitsanforderung in formaler Sprache (CTL):

```
AG (Bahnuebergang.sendeUeberschreitungsmeldung →
(A((Schrankenanlage.SchA_ausschalten = 0 and
Lichtzeichenanlage.LzA_ausschalten = 0) W
Bahnuebergang.Status_Ausschaltbefehl)
```

In [Bits03a] wird beispielhaft die Spezifikation einer Sicherheitsanforderung auf der Grundlage der Komposition von Safety-Pattern gezeigt. Um nur dann Kompositionen zu verwenden, wenn es erforderlich ist, gilt folgende Spezifikationsregel: Es muss untersucht werden, wie viele zeitlich-logische Relationen zwischen Systemvariablen in der betreffenden Sicherheitsanforderung vorhanden sind. Sind es mehr als zwei Relationen (Relationen über drei Aussagen) ist eine Komposition aus mehreren Safety-Pattern erforderlich. Das in diesem Abschnitt verwendete

Beispiel umfasst zwei zeitlich-logische Relationen über drei Aussagen. Die Safety-Pattern-Menge und das Safety-Pattern-Werkzeug wurden so konzipiert, dass für weniger als drei Relationen ein Safety-Pattern vorhanden ist oder das Werkzeug im Selektionsprozess vorgibt, aus welchen Safety-Pattern-Klassen Safety-Pattern für die Komposition selektiert werden müssen.

8.5 Erprobung des Systemverhaltens in sicherheitskritischen Situationen

Um die Korrektheit der formalen Spezifikation der Sicherheitsanforderungen anhand des Modellprozesses zu demonstrieren, ist eine Steuerung erforderlich, welche gegenüber den Sicherheitsanforderungen korrekt ist. Gemäß dem Modellprozess-Verhalten in den von den Sicherheitsanforderungen beschriebenen sicherheitsrelevanten Situationen können Rückschlüsse auf die Korrektheit der Steuerung und auf die Korrektheit der Formalisierung der Sicherheitsanforderungen geschlossen werden. Würde eine Sicherheitsanforderung fehlerhaft formalisiert werden, sodass die ausgedrückte Relation die sichere Eigenschaft nicht mehr ausdrückt, wird bei der formalen Verifikation diese Sicherheitsanforderung nicht überprüft. Dadurch können in der Steuerung sicherheitskritische Fehler vorhanden sein.

In Abbildung 8.4 wird das Prinzip für die Erprobung der korrekten Formalisierung von Sicherheitsanforderungen veranschaulicht. Über die Experimentierfunktionen des Modellprozesses können Störungsvorgaben getätigt werden, die sicherheitskritische Situationen verursachen, wie z. B. ein Ausfall des Schrankenantriebs. Dann wird beobachtet, ob das Systemverhalten trotz der Ausfälle sicher ist und die Sicherheitsanforderungen eingehalten werden. Aufgrund dessen können Rückschlüsse hinsichtlich der fehlerfreien Formalisierung der Sicherheitsanforderungen gezogen werden. Ein unkorrektes Verhalten lässt jedoch nicht zwingend auf Formalisierungsfehler schließen. Es können auch andere Fehlerursachen vorliegen, insbesondere bei der Transformation des UML-Modells in Programmcode.

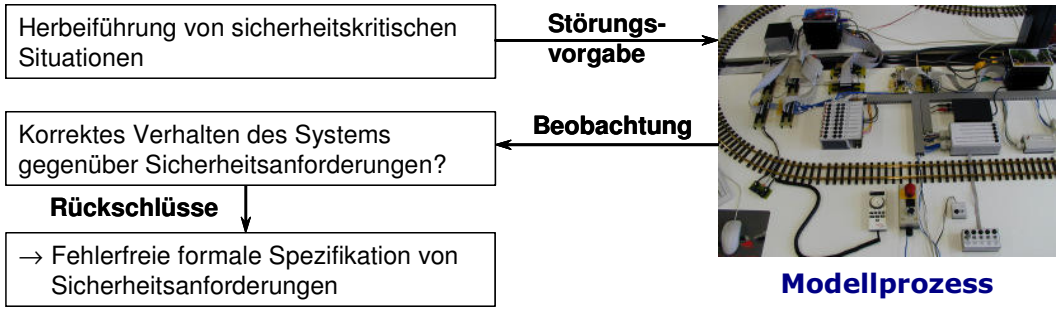


Abbildung 8.4: Prinzip für die Erprobung

Um Flüchtigkeitsfehler bei der Codierung auszuschließen, wurde für die Umsetzung des Modells in C++-Programmcode eine Codegenerierung entwickelt und angewandt [AGB04] [Faud04] [Link03]. Hierfür wurden die Codegenerierungsmöglichkeiten des UML-Werkzeugs

Software through Pictures (StP) der Firma Aonix genutzt. Der Codegenerator erlaubt die Vorgabe von Transformationsregeln in so genannten *Architecture Component Development* (*ACD*) Templates (vgl. [Aoni02]). Sie werden in einer werkzeugspezifischen Sprache – der *Transformation Description Language* (TDL) [Aoni02] [HuSe00] verfasst (vgl. Abbildung 8.5).

Im Lieferumfang von StP sind bereits Vorgaben für Templates zur Umsetzung nach C++ enthalten. Diese wurden in [Faud04] in Kooperation mit der Firma Aonix erweitert, um Klassen- und Zustandsdiagramme in der erforderlichen Weise in C++-Code umzusetzen.

Wie aus Abbildung 8.5 ersichtlich ist, wird der Codegenerator von StP zudem genutzt, um das Steuerungsmodell für die formale Korrektheitsprüfung in die Eingangssprache des Model-Checker SMV zu transformieren [Faud04] [Gont03]. Die Umsetzung erfolgte gemäß der in [Faud04] definierten Umsetzungsregeln. Das so erzeugte Modell in der Eingangssprache von SMV war die Basis für die formale Verifikation. Die 51 Sicherheitsanforderungen wurden mithilfe von 17 Safety-Pattern formalisiert. Alle Sicherheitsanforderungen wurden durch Model-Checking verifiziert. Zunächst waren zwei davon nicht erfüllt. Zum einen war folgende Sicherheitsanforderung verletzt [Bits03b]: *Der Bahnübergang darf nicht abgeschaltet werden, während eine Anforderung vom Eisenbahnfahrzeug anliegt.*

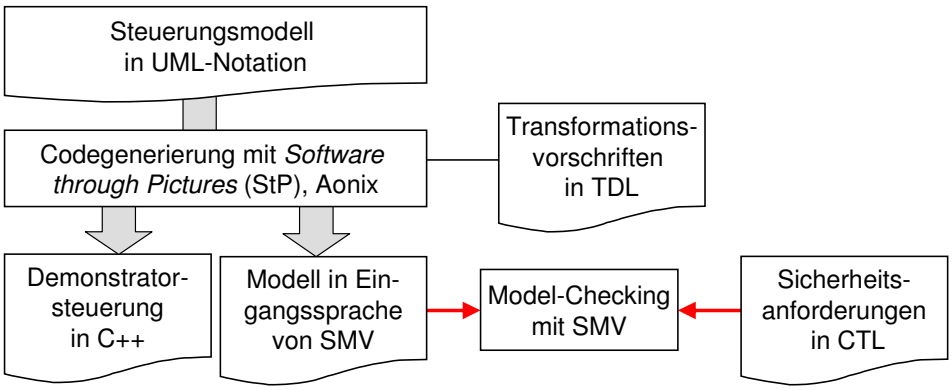


Abbildung 8.5: Transformation des UML-Modells der Systemfunktionen in C++-Code und in die Eingangssprache des Model-Checker SMV

Die Ursache für die Verletzung war, dass nach der Anforderung des Bahnübergangs durch das Eisenbahnfahrzeug der Bahnübergang im selben Zustand ist, der auch durch das Ereignis „außer Betrieb nehmen“ (d. Abschaltung) verlasst werden kann. Als Maßnahme für die Einhaltung der Sicherheitsanforderung musste in das Modell ein Zustandswechsel nach einem Einschaltbefehl eingefügt werden.

Zum anderen war folgende Sicherheitsanforderung verletzt: *Bei einer Zeitüberschreitung kann die Sicherung eines Bahnübergangs nur von der FFB-Zentrale aufgehoben werden.*

Als Maßnahme musste im Zustandsdiagramm der Klasse *Bahnuebergang* die unzulässigen Abschaltbefehle *LzA_ausschalten* in durch die Zeitüberschreitung ausgelösten Transitionen entfernt werden [Faud04].

Wie oben bereits beschrieben, ist die aus dem verifizierten UML-Modell generierte Steuerungssoftware die Grundlage für Prüfungen mit dem Modellprozess. Ein Beispiel für Prüfung des Systemverhaltens in identifizierten sicherheitskritischen Situationen ist folgendes:

Sicherheitsanforderung:
Wenn im Teilsystem Bahnübergang die untere Schrankenendlage nicht erreicht ist, darf der Status des Bahnübergangs nicht als gesichert gemeldet werden.

Mögliche Störungsvorgabe zur Herbeiführung der sicherheitskritischen Situation:
Antrieb für Schrankenschließung des Teilsystems Bahnübergang wird deaktiviert.

Folgen der Störungsvorgabe:
- Der Bahnübergang ist ungesichert.

Demonstration – beobachtetes Verhalten:
- Es erfolgt keine Statusmeldung.
- Das Eisenbahnfahrzeug kommt vor dem Bahnübergang zum Halt.

Das beobachtete Verhalten des Modellprozesses ist sicher und entspricht der Sicherheitsanforderung.

Auf diese Weise wurde die Einhaltung der Sicherheitsanforderungen beim Betrieb des Modellprozesses beobachtet. Das Safety-Pattern-Konzept hat sich für die Formalisierung der Sicherheitsanforderungen bewährt (vgl. [AGB04]). Durch die formale Verifikation wurden die sicherheitsrelevanten systematischen Fehler ausgeschlossen. Bei der Verwendung des Modellprozesses wurden auch nicht-sicherheitskritische Modellierungsfehler aufgedeckt. Ein erkannter nicht-sicherheitskritischer Modellierungsfehler, der durch die Erprobungsarbeiten mit dem Modellprozess identifiziert wurde, war folgender [Bits03b]:

Beobachtung: Das Eisenbahnfahrzeug hält immer vor dem Bahnübergang an.

Grund: Die Bedingung für die Bremsung war immer erfüllt.

Ursache: Die maximale Zeit, in der auf das Erreichen des Bahnübergangs gewartet wird, ist zu gering.

Maßnahme: Diese Zeit muss entsprechend der Zeit gewählt werden, die der Bahnübergang maximal geschlossen sein darf.

8.6 Erfahrungen und Erkenntnisse

Neben der Anwendung „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb“ dienten insbesondere die Automatisierungssysteme „Elektrisch ortsbediente Weiche“ [CGM97], Zugtürsteuerung (in einer Kooperation mit dem TÜV Nord, Hamburg), Bremssteuermodul für Nutz-

fahrzeuge [Bits99] [Büch00], Brake-by-Wire-System [Gunz03] [Näge98] sowie Ampelsteuerung [Moik02] als Beispiele für den Einsatz des Safety-Pattern-Konzepts zur Formulierung von Sicherheitsanforderungen. Bereits während der Entwicklung des Verfahrens wurden in studentischen Arbeiten Erfahrungen mit der Verwendung des Safety-Pattern-Konzepts gewonnen [Lova03] [Jime03] [Karu04] [Touc04] [Teuf04] [Webe04]. Ausgehend von den gesammelten Erfahrungen wird im Folgenden die Erfüllung der in Kapitel 3 beschriebenen Konzept-Anforderungen durch das Safety-Pattern-Konzept bewertet.

/KA1/ Präzise Spezifikation

Das Safety-Pattern-Konzept ermöglicht die präzise Spezifikation funktionaler Sicherheitsanforderungen. Durch die Safety-Pattern werden präzise generische Formulierungen vorgegeben, die bei der Anwendung des Konzepts wiederverwendet werden. Dabei erfolgt die Spezifikation in der Safety-Pattern-Normsprache, der grafischen Notation GNSS sowie in Temporallogik. Durch detaillierte und ausführliche Erläuterungen unterschiedlicher Art und Weise sowie die systematische Klassifikation ist die präzise Bedeutung der generischen Formulierungen festgelegt.

/KA2/ Formal verifizierbare Spezifikation

Mithilfe der formalen generischen Formulierungen in Temporallogik wird eine formal verifizierbare Spezifikation funktionaler Sicherheitsanforderungen ermöglicht. Durch die Angabe der generischen Formulierungen in mehreren temporallogischen Sprachen werden verschiedene formale Verifikationswerkzeuge unterstützt.

/KA3/ Vollständige Spezifizierbarkeit

Für die Spezifikation von Anforderungen bietet das Verfahren für alle sicherheitsrelevanten funktionalen Eigenschaften von Automatisierungssystemen geeignete Formulierungsmöglichkeiten in Temporallogik. Der Safety-Pattern-Katalog ist eine vollständige Basis, um durch die Verwendung einzelner generischer Formulierungen sowie durch deren Komposition alle Arten funktionaler Sicherheitsanforderungen formal ausdrücken zu können.

/KA4-5/ Anwendbarkeit

Fehlervermeidung durch das Verfahren: Safety-Pattern vermitteln Expertenwissen für die korrekte Spezifikation und Interpretation formaler Sicherheitsanforderungen. Durch die Wiederverwendung vorgefertigter Formulierungen werden Spezifikations- und Interpretationsfehler vermieden, die beim selbstständigen Gebrauch von Temporallogik leicht auftreten können. Dabei sind für die Verwendung des Safety-Pattern-Konzepts keinerlei Kenntnisse in Temporallogik erforderlich. Somit werden Fehler, die in der mangelhaften Beherrschung der formalen Spezifikationssprache begründet sind, vollständig vermieden. Fehler können jedoch auch bei der Anwendung des Verfahrens entstehen. Wesentlich für die Vermeidung solcher Fehler ist, dass die Safety-Pattern-Klassen klar verstanden werden. Ebenso müssen die Erläuterungen der

Safety-Pattern korrekt erfasst werden. Um sich des korrekten Verständnisses gewiss zu sein, sind dabei die diversitären Erläuterungen der Safety-Pattern sehr hilfreich. Dabei trägt die Verwendung multimedialer Elemente in den Erläuterungen (textuelle und grafische Beschreibungen und interaktive Simulation) zur einfachen Verständlichkeit bei. Für die Vermeidung von Fehlern bei der Anwendung des Verfahrens ist zudem die werkzeuggestützte interaktive Führung essenziell.

Schwierigkeiten des Verfahrens: Die Erstellung komplexer Sicherheitsanforderungen durch Komposition von Formulierungen kann insbesondere für Systementwickler ohne Kenntnisse in höherer Logik mit Schwierigkeiten verbunden sein. Durch die werkzeuggestützte Führung und Beratung bei der Komposition wird jedoch diesen Schwierigkeiten entgegengewirkt. Zudem wird durch die Simulationsmöglichkeit für aussagenlogische Verknüpfungen sowie die Veranschaulichung mit GNSS für ineinander eingesetzte Formulierungen die Komposition unterstützt. Generell werden Schwierigkeiten mit dem Verfahren insgesamt jedoch dadurch gering gehalten, indem wenige Voraussetzungen für die Anwendung erforderlich sind.

Erlernbarkeit des Verfahrens: Voraussetzung für die Anwendung des Verfahrens sind Kenntnisse in Aussagenlogik. Zudem ist das Erlernen der GNSS-Notation für den Gebrauch des Verfahrens vorteilhaft, wobei sich die verwendeten Symbole an bekannten Beschreibungsmitteln orientieren. Die Notationen für die Simulation sowie für die grafischen Erläuterungen bauen auf gebräuchlichen Darstellungsarten auf, sodass auch hier der Lernaufwand gering gehalten wird. Ohne Lernaufwand können die Erläuterungen der Safety-Pattern in natürlicher Sprache verwendet werden. Ebenso einfach verständlich sind die Safety-Pattern-Klassifikation sowie die Erläuterung der Klassifikation in natürlicher Sprache. Die Safety-Pattern-Normsprache muss generell nicht erlernt werden, da die verwendete Terminologie intuitiv verständlich ist und die Bedeutung der Fachwörter jederzeit werkzeuggestützt nachgeschlagen werden kann. Für die effiziente Anwendung ist jedoch die Kenntnis der Fachwörter-Definitionen hilfreich. Diese wird durch Erfahrungen mit dem Gebrauch der Normsprache gewonnen. Personen mit guten Kenntnissen in Temporallogik können die generischen formalen Formulierungen der Safety-Pattern direkt interpretieren. Sie müssen für die Anwendung des Verfahrens lediglich SAPIS verwenden und die Klassifikation verstehen.

Für die intuitive Erlernbarkeit ist die Werkzeugführung von entscheidender Bedeutung. Dabei spielt die selektive Dokumentations-Präsentation für die einfache und effiziente Erlernbarkeit des Verfahrens eine wichtige Rolle.

Aufwand und Effizienz des Verfahrens: Der durch die Safety-Pattern-Klassifikation bestimmte Selektionsprozess, das Durcharbeiten der Safety-Pattern-Erläuterungen, das Nachschlagen im Normsprachen-Lexikon, die schrittweise Führung durch das Safety-Pattern-Werkzeug sowie gegebenenfalls die Komposition von Formulierungen bringt einen hohen Zeitaufwand mit sich. Da es bei Automatisierungssystemen mit Sicherheitsverantwortung um die Sicherung von Leib

und Leben und der Vermeidung hoher Sachschäden geht, ist generell ein hoher Entwicklungsaufwand vertretbar. Darüber hinaus ist ein hoher Spezifikationsaufwand deshalb akzeptabel, weil das Safety-Pattern-Konzept die Anwendung und Nutzung von formalen Sicherheitsnachweisen ermöglicht. Das Verfahren umfasst jedoch verschiedene Prinzipien, durch die ein zu hoher Aufwand vermieden wird.

Durch die Werkzeugunterstützung und die Automatisierung algorithmischer Vorgänge (z. B. Datenbankabfrage) wird die Effizienz bei der Verwendung des Safety-Pattern-Konzepts entscheidend erhöht. Sind Safety-Pattern aufgrund früherer Anwendungen des Verfahrens bereits bekannt, müssen die umfassenden Erläuterungen nicht mehr gelesen werden. Stattdessen sind in diesem Fall die Beschreibungen in der Safety-Pattern-Normsprache oder in GNSS ausreichend. Diese geben die Safety-Pattern-Bedeutung prägnant und deshalb schnell erfassbar wieder. Die Erfahrungen mit dem Safety-Pattern-Konzept zeigen, dass lediglich eine überschaubar kleine Anzahl von Safety-Pattern (< 20) häufig zur Anwendung kommt. Sobald man Erfahrungen in der Verwendung von Safety-Pattern gesammelt hat, ist deshalb das Durcharbeiten der Erläuterungen nur in vereinzelten Fällen erforderlich. Da das Verfahren Mehrfachverwendung unterstützt, muss somit nur in wenigen Fällen der Selektionsprozess durchlaufen werden. Einmal selektierte Safety-Pattern können gespeichert und für andere Sicherheitsanforderungen oder gar in anderen Projekten ohne Durchlaufen des Selektionsprozesses direkt für die Instanziierung verwendet werden. Deshalb ist der Mehrfachverwendung eine entscheidende Bedeutung für die effiziente Anwendung des Safety-Pattern-Konzepts beizumessen.

Beim Selektionsprozess ist der Aufwand geringer, je mehr dem Benutzer des Verfahrens die Safety-Pattern-Klassifikation vertraut ist. Je nach Erfahrungsstand können unterschiedliche Dialogarten zur Selektion verwendet werden. Die auf den Kenntnisstand unterschiedlicher Personen ausgelegte Werkzeugführung ist ein wesentlicher Faktor für die Anwenderakzeptanz.

In diesem Kapitel wurde anhand des Modellprozesses „Eingleisiger Bahnübergang im Funk-Fahr-Betrieb“ die Anwendung des Safety-Pattern-Konzepts dargestellt. Dabei wurde die Vorgehensweise zur Demonstration der Konzeptanwendung und des Konzeptnutzens erläutert. Exemplarisch wurde die Spezifikation der Sicherheitsanforderungen für die Beispielanwendung dargestellt. Die Korrektheit des Steuerungsmodells gegenüber den Sicherheitsanforderungen wurde formal verifiziert. Die Einhaltung der Sicherheitsanforderungen wurde im Betrieb des Modellprozesses erprobt. Somit wurde insgesamt gezeigt, wie das Safety-Pattern-Konzept in der Praxis eingesetzt werden kann. Unter Berücksichtigung der gesammelten Erfahrungen wurde die Erfüllung der Konzept-Anforderungen erörtert. Im nächsten Kapitel folgt eine Zusammenfassung der Ergebnisse und der Bewertung des Safety-Pattern-Konzepts. Grenzen des Verfahrens werden aufgezeigt, und es wird ein Ausblick auf mögliche Fortsetzungsarbeiten gegeben.

9 Schlussbetrachtungen und Ausblick

9.1 Zusammenfassung der Ergebnisse und Bewertung

In der vorliegenden Arbeit wurde das Safety-Pattern-Konzept als Verfahren zur anwendbaren Spezifikation funktionaler Sicherheitsanforderungen für Automatisierungssysteme entwickelt. Damit wurde ein systematisches ingenieurgerechtes Verfahren konzipiert, das die korrekte formale Spezifikation und Interpretation von Sicherheitsanforderungen in formalen Sprachen unterstützt. Das Verfahren nutzt Expertenwissen für die Spezifikation von Sicherheitsanforderungen, sodass für die Verwendung keinerlei Kenntnisse in höherer Logik erforderlich sind. Somit ist das Safety-Pattern-Konzept eine Grundlage für die Anwendung formaler Methoden zur Durchführung von Sicherheitsnachweisen bei der Entwicklung von Automatisierungssystemen.

Das Verfahren ersetzt den selbstständigen fehleranfälligen Gebrauch formaler Spezifikationssprachen zur Formulierung von Sicherheitsanforderungen durch die Führung bei der Transformation von Ingenieur-Vorstellungen über Sicherheitsanforderungen in eine formal verifizierbare Darstellung. Durch die systematische Unterstützung dieses Transformationsprozesses behandelt das Safety-Pattern-Konzept die Kernproblematik formaler Spezifikation, vgl. [PaSc99].

Der Safety-Pattern-Katalog wurde so entwickelt, dass er durch die Verwendung einzelner generischer Formulierungen sowie durch deren Komposition eine vollständige Grundlage zur Spezifikation aller Arten funktionaler Sicherheitsanforderungen für Automatisierungssysteme bietet. Darüber hinaus enthält der Katalog generische Formulierungen, durch welche die Handhabbarkeit des Verfahrens gewährleistet ist.

Das Safety-Pattern-Konzept geht durch die alternativen Möglichkeiten bei der Safety-Pattern-Selektion sowie bei der Analyse selektierter Safety-Pattern auf die unterschiedlichen Vorkenntnisse verschiedener Benutzer ein. Es ist sowohl für Sicherheitsingenieure verwendbar, die keinerlei Kenntnisse in höherer Logik besitzen als auch für Experten für formale Methoden, welche die Ergebnisse ihrer manuellen Spezifikation überprüfen möchten.

Ein wesentlicher Aspekt für die Anwendung des Safety-Pattern-Konzepts in der industriellen Praxis ist, dass es dabei hilft, Empfehlungen und Forderungen der aktuell gültigen internationalen, europäischen und deutschen Normen für funktionale Sicherheit nachzukommen. Das Verfahren unterstützt die von den Normen empfohlene formale Spezifikation der Sicherheitsanforderungen. Ebenso wird die Beschreibung formaler Spezifikationen in Worten der natürlichen Sprache unterstützt. Darüber hinaus behebt das Verfahren Schwachstellen in den Normen. Das Verfahren umfasst ein methodisches Vorgehen für den korrekten Gebrauch der formalen Sprachen bei der Spezifikation und Interpretation. Durch die Safety-Pattern-Norm-

sprache und die Verwendung vorformulierter Spezifikationen wird ein systematischer Weg zum Gebrauch natürlicher Sprache zur Beschreibung der formalen Spezifikation aufgezeigt, sodass Mehrdeutigkeiten vermieden werden. Zudem wird die Äquivalenz der formalen Spezifikation und der Spezifikation in der Safety-Pattern-Normsprache gewährleistet, da durch die Safety-Pattern die Äquivalenzen fest definiert sind.

Voraussetzung für die Verwendbarkeit des Safety-Pattern-Konzepts sind Kenntnisse in Aussagenlogik. Durch die Verwendung des Verfahrens kann nicht ausgeschlossen werden, dass bei der Spezifikation von Sicherheitsanforderungen Fehler passieren. Allerdings kann durch die Verwendung des methodischen Vorgehens mit vorgefertigten generischen Spezifikationen das Risiko für Spezifikationsfehler wesentlich reduziert werden. Entscheidend für die Nutzung des Verfahrens ist das korrekte Verständnis der Safety-Pattern-Klassen und der Klassifikationskriterien.

Für Systementwickler ohne Kenntnisse in höherer Logik kann die Erstellung von komplexen Sicherheitsanforderungen durch Komposition mit Aufwand und Erschwernissen verbunden sein. Durch die werkzeuggestützte Führung und Kontrolle durch ein Beratungssystem wird jedoch auch in diesem Fall das effiziente und korrekte Spezifizieren unterstützt. Insbesondere für Systementwickler, die mit dem Safety-Pattern-Konzept nicht vertraut sind, kann die Verwendung des Verfahrens am Anfang aufwendig sein. Da es jedoch um die Entwicklung sicherer Automatisierungssysteme geht und durch das Safety-Pattern-Konzept die Anwendung und Nutzung von formalen Sicherheitsnachweisen ermöglicht wird, ist der Aufwand gerechtfertigt und zweckmäßig. Zudem wird durch die umfassende Werkzeugunterstützung für alle Spezifikationsschritte, die Kopplung von Dokumentation und Werkzeugunterstützung und der Unterstützung von Mehrfachverwendung einem zu hohen Aufwand entgegengewirkt und eine einfache und effiziente Handhabbarkeit für die Spezifikation und Interpretation funktionaler Sicherheitsanforderungen für Automatisierungssystem erreicht.

Mit dem Safety-Pattern-Konzept wurde ein Verfahren geschaffen, dessen Kern das Prinzip der Wiederverwendung bildet. Dabei integriert es durch die Kombination verschiedener generischer Beschreibungs- und Erläuterungsarten und durch die selektive werkzeuggestützte Benutzerführung die Ansprüche unterschiedlicher Personengruppen des Entwicklungs- und Prüfungsprozesses für sichere Automatisierungssysteme. Verschiedene Betrachtungsweisen werden unterstützt. Unterschiedlichen Benutzern wird Expertenwissen für die Spezifikation funktionaler Sicherheitsanforderungen adaptiv vermittelt. Dadurch wird einerseits die einfache Anwendbarkeit und Erlernbarkeit sowie die Vermeidung von Fehlern unterstützt und andererseits einem zu hohen Aufwand entgegengewirkt. Ein solches Wiederverwendungskonzept wie es mit dem Safety-Pattern-Konzept für die Spezifikation von Sicherheitsanforderungen entwickelt wurde, bietet überall dort, wo sich ein selbstständiges Vorgehen als schwierig und fehleranfällig erweist und Fehler schwerwiegende Folgen nach sich ziehen können, Lösungsansätze.

9.2 Ausblick

Das Safety-Pattern-Konzept unterstützt die Formalisierung funktionaler Sicherheitsanforderungen. Darüber hinaus ist auch eine Vereinfachung der Spezifikation des quantitativen Anteils in Sicherheitsanforderungen (vgl. Abschnitt 2.2.3) denkbar. Die Spezifikation zeitlich-stochastischer Informationen in Sicherheitsanforderungen, um quantitative Aussagen über das Versagen von Sicherheitsfunktionen aufgrund von physikalischen Fehlern auszudrücken, kann ebenso durch ein Wiederverwendungsprinzip vereinfacht werden. Hierfür sind Formulierungen in einer stochastischen Temporallogik wie zum Beispiel CSL (vgl. Absatz 6.1.2.1) erforderlich. Mit entsprechenden formalen Verifikationswerkzeugen (z. B. [CJMS01] und [HKKS03]) kann die Einhaltung solcher Sicherheitsanforderungen durch Systemmodelle, in denen stochastisches Ausfallverhalten modelliert ist, überprüft werden.

Ähnlich zur Musterbildung für Sicherheitsanforderungen können auch Muster für bewährte sicherheitsrelevante Verhaltensbeschreibungen in gängigen Notationen wie UML-Zustandsdiagrammen, Petri-Netzen sowie SFC gebildet werden. Aufgrund der bereits in Abschnitt 5.3.7.2 genannten Unterschiede zwischen Modellen für Systemfunktionen, die für sichere Automatisierungssysteme geeignet sind, und Sicherheitsanforderungen, ist jedoch eine andere Klassifizierung als die des Safety-Pattern-Ansatzes sinnvoll. Entsprechend ist auch die Entwicklung eines Software-Komponenten-Katalogs für sichere Automatisierungssysteme zweckmäßig. Sowohl bei den einzelnen Komponenten als auch bei den Verhaltensmustern ist dabei ein Verweis auf die Safety-Pattern mit den generischen Sicherheitsanforderungen sinnvoll, welche durch das generische Entwicklungselement erfüllt werden.

Es ist zu untersuchen, ob die Safety-Pattern-Klassifikation auch bei der Entwicklung von Automatisierungssystemen mit Sicherheitsverantwortung zur Unterstützung der Identifikation von Sicherheitsanforderungen verwendet werden kann. Dabei würden die verschiedenen Safety-Pattern-Klassen Hinweise geben, welche Sicherheitsanforderungsarten es generell gibt und damit zum Nachdenken anregen, welche der allgemein denkbaren Fälle für das zu entwickelnde Automatisierungssystem sinnvoll und notwendig sind. Gegebenenfalls müsste speziell für diesen Zweck eine geeignete Liste mit Schlüsselbegriffen für Sicherheitsanforderungsarten erstellt werden.

Literaturverzeichnis

[ABG04] S. Arabestani, F. Bitsch, J.-T. Gayen: *Precise Definition of the Single-track Level Crossing in Radio-based Operation in UML Notation and Specification of Safety Requirements.* In: Integration of Software Specification Techniques for Applications in Engineering, H. Ehrig et al. (eds.). LNCS 3147, Springer Verlag, Berlin, Heidelberg, New York, 2004, pp. 119-144.

[ABG02] S. Arabestani, F. Bitsch, J.-T. Gayen: *2. Zwischenbericht im DFG-Projekt SafeRail „Integration von Methoden zur Spezifikation und Verifikation von Sicherungseinrichtungen im spurgeführten Verkehr“.* Projekt im DFG Schwerpunktprogramm „Integration von Techniken der Softwarespezifikation für ingenieurwissenschaftliche Anwendungen“, März 2002.

[Adtr97] Adtranz: *Lastenhefte zum Funkfahrbetrieb(FFB).* Deutsche Bahn AG, 1997.

[AFSC00] W. Ahrens, M. Felleisen, E. Schnieder, M. Chouikha*: Formale Prozessbeschreibungen – gestern, heute und morgen.* In: atp – Automatisierungstechnische Praxis 42 (2000) H. 9, S. 24-32.

[AHMS99] S. Autexier, D. Hutter, H. Mantel, A. Schairer: *System Description: INKA 5.0 – A Logic Voyager*. In: Proceedings of CADE-16, Trento, 1999.

[Alex80] C. Alexander: *The Timeless Way of Building*. Oxford University Press, New York, 1980.

[AlHe92] R. Alur, T. Henzinger: *Logics and Models of Real Time: A Survey.* In: Real Time: Theory in Practice, J.W. de Bakker, K. Huizing, W.-P. de Roever, G. Rozenberg (eds.). LNCS 600, Springer Verlag, 1992.

[ABB+01] T. Amnell, G. Behrmann, J. Bengtsson, P. R. D'Argenio, A. David, A. Fehnker, Th. Hune, B. Jeannet, K.G. Larsen, M. O. Möller, P. Pettersson, C. Weise, W. Yi: *Uppaal – Now, Next, and Future*. In: Proceedings of Modelling and Verification of Parallel Processes (MOVEP'2k), F. Cassez, C. Jard, B. Rozoy, and M. Ryan (eds.). LNCS Tutorial 2067. Springer Verlag, 2001, pp. 100-125.

[Aoni02] Aonix GmbH: *ACD-Programming Guide für Software through Pictures.* April 2002.

[Arab05] S. Arabestani: *Formal verifizierbare objektorientierte Systemspezifikationen mit UML für Eisenbahnsicherungssysteme*. Diss. IfEV, TU Braunschweig, 2005. IfEV-Schriftenreihe, J. Pachl (Hrsg.), Heft 67, 2005.

[Arab03] S. Arabestani: *Umschreibung von Zustandsdiagrammen der UML für das Model-Checking*. In: Entwurfsmethodik, Modellbildung, Werkzeuge und Anwendungen, 8. Fachtagung Entwurf komplexer Automatisierungssysteme (EKA), E. Schnieder (Hrsg.). Braunschweig, 2003, S. 63-79.

[ArGa00] S. Arabestani, J.-T. Gayen: *Objektorientierte Analyse zur Modellierung im Eisenbahnwesen*. In: Signal + Draht, 92(2000)1+2, S. 20-27.

[ASSB00] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton: *Model-checking continuous time Markov chains.* In: ACM Transactions on Computational Logic (TOCL) Volume 1 (2000) Issue 1, pp. 162-70.

[Balz96] H. Balzert: *Lehrbuch Software-Technik – Software-Entwicklung*. 1. Auflage, Spektrum Akademischer Verlag, Heidelberg, Berlin, Oxford, 1996.

[Balz99] H. Balzert: *Lehrbuch der Objektmodellierung – Analyse und Entwurf.* Spektrum Akademischer Verlag, Heidelberg, Berlin, 1999.

[Bare93] R. Bareiß: *Verfahren zur Propagierung von Zeitbedingungen und zur formalen Verifikation des zeitlichen Verhaltens beim Entwurf von Automatisierungssystemen*. Diss. IAS, Universität Stuttgart, 1993.

[BaPi04] K. J. Barltrop, P. J. Pingree: *Model Checking Investigations for Fault Protection System Validation (cont.).* MathWorks / NASA, 2004. http://www.mathworks.com/company/newsletters/aero_digest/feb04/ modcheck2.html

[BBB+04] R. Buschermöhle, M. Brörkens, I. Brückner, W. Damm, W. Hasselbring, B. Josko, C. Schulte, Th. Wolf: *Model Checking – Grundlagen und Praxiserfahrungen*. Informatik Spektrum, Vol. 27 (2004) Issue 2, S. 146-158.

[BCL91] J. R. Burch, E. M. Clarke, D. E. Long: *Representing circuits more efficiently in symbolic model checking*. In: Proceedings of the 28th Design Automation Conference. IEEE Computer Society Press, 1991, pp. 403-407.

[BCM+90] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, J. Hwang: *Symbolic Model Checking: 10E20 states and beyond.* In: LICS, 1990. http://www-2.cs.cmu.edu/~modelcheck/ed-papers/smctsab.pdf

[BCM00] F. Bitsch, E. Canver, A. Moik: *Strukturierte Erstellung von Sicherheits-spezifikationen in UML mit Hilfe der FMEA-Methode.* In: FORMS 99 – Formale Techniken für die Eisenbahnsicherung, E. Schnieder (Hrsg.). Fortschritt-Berichte VDI, Reihe 12, Nr.436. VDI Verlag GmbH, Düsseldorf, 2000, S. 225-245.

[Bels97] R. Belschner: *Ein Verfahren zur simulationsbasierten Analyse von verteilten Prozeßautomatisierungssystemen in der Spezifikationsphase*. Diss. IAS, Universität Stuttgart, 1997.

[Béra01] B. Bérard et al.: *Systems and Software Verification – Model-Checking Techniques and Tools.* Springer Verlag, Berlin, Heidelberg, New York, 2001.

[BDI+99] Bundesamt für Sicherheit in der Informationstechnik, DFKI GmbH, Institut für Automatisierungs- und Softwaretechnik (Universität Stuttgart), Innovative Software Technologie, Abt. Programmiermethodik und Compilerbau (Universität Ulm): *VSE-II CD-ROM*. 1999.

[Beck92] Th. Beck: *Verfahren zur Behandlung zeitlicher Beziehungen in Echtzeitexpertensystemen und zur Integration wissensbasierter Komponenten in Automatisierungssysteme*. Diss. IAS, Universität Stuttgart, 1992.

[BHLE04] N. Bauer, R. Huuck, B. Lukoschus, S. Engell: *A Unifying Semantics for Sequential Function Charts.* In: Integration of Software Specification Techniques for Applications in Engineering, H. Ehrig et al. (eds.). LNCS 3147, Springer Verlag, Berlin, Heidelberg, New York, 2004, pp. 400-418.

[Bier97] A. Biere: *Effiziente Modellprüfung des μ-Kalküls mit binären Entscheidungsdiagrammen*. Dissertation, Universität Karlsruhe, 1997.

[BiGö02] F. Bitsch, P. Göhner: *Spezifikation von Sicherheitsanforderungen mit Safety-Patterns.* In: Veröffentlichungsband von Software Engineering in der industriellen Praxis, VDI-Bericht-Nr. 1666. VDI Verlag GmbH, Düsseldorf, 2002, S. 29-40.

[BiGu00] F. Bitsch, M. Gunzert: *Formale Verifikation von Softwarespezifikationen in ASCET-SD und Matlab®.* In: Begleitband zur Fachtagung „Verteilte Automatisierung – Modelle und Methoden für Entwurf, Verifikation, Engineering und Instrumentierung“, Ch. Döschner (Hrsg.). Otto-von-Guericke-Universität Magdeburg, 2000, S. 302-310.

[Bits03a] F. Bitsch: *A Way for Applicable Formal Specification of Safety Requirements by Tool-Support.* In: Proceedings of “FORMS 2003 – Symposium on Formal Methods for Railway Operation and Control Systems”, E. Schnieder, G. Tarnai (eds.). L’Harmattan, Budapest, Hungary, 2003, pp. 175-185.

[Bits03b] F. Bitsch: *Erprobung von Konzepten zur korrekten Spezifikation von Sicherungseinrichtungen anhand des Demonstrators „Eingleisiger Bahnübergang im FFB*“. In: 6. Kolloquium im DFG-Schwerpunktprogramm „Integration von Techniken der Softwarespezifikation für ingenieurwissenschaftliche Anwendungen”, Stuttgart, 20.-21.11.2003.

[Bits02a] F. Bitsch: *Process Model for the Development of System Requirements Specifications for Railway Systems*. In: Workshop on Software specification of safety relevant transportation control tasks, E. Schnieder (ed.). VDI-Fortschrittbericht, Reihe12 Verkehrstechnik / Fahrzeugtechnik, Nr. 535. VDI Verlag GmbH, Düsseldorf, 2002, pp. 75-90.

[Bits02b] F. Bitsch: *Requirements on Methods and Techniques in Perspective to Approval Process for Railway Systems.* In: Proceedings of 2nd International Workshop on Integration of Specification Techniques for Applications in Engineering – Satelite Event of ETAPS 2002, H. Ehrig, M. Große-Rhode (eds.). Grenoble, 2002, http://tfs.cs.tu-berlin.de/~mgr/int02/papers/bitsch.pdf

[Bits01] F. Bitsch: *Safety Pattern – The Key to Formal Specification of Safety Requirements*. In: Proceedings of 20th International Conference SAFECOMP 2001 – Computer Safety Reliability and Security, U. Voges (ed.). LNCS 2187. Springer Verlag, Berlin, Heidelberg, New York, 2001, S. 176-189.

[Bits00] F. Bitsch: *Classification of Safety Requirements for Formal Verification of Software Models of Industrial Automation Systems*. In: Proceedings Volume 5 of 13th International Conference Software and Systems Engineering and their Applications (ICSSEA), Paris, 2000.

[Bits99] F. Bitsch: *Entwicklung eines Bremssteuermoduls für Nutzfahrzeuge.* Diplomarbeit, SADA-Nr. 1690, IAS, Universität Stuttgart, 1999.

[Brab02] J. Braband: *Lessons Learned from Risk Assessment Studies in Railway Signalling.* In: Moderne Sicherheitsanalysen – Theorie und Praxis, J. Pachl (Hrsg.). IfEV Schriftenreihe, TU Braunschweig, Heft 64, 2001, S. 3-16.

[Bray96] R. Brayton et. Al: *VIS: A system for Verification and Synthesis*. In: Proceedings of the 8th International Conference on Computer Aided Verification, R. Alur and T. Henzinger (eds.). New Brunswick, NJ, Springer Verlag, LNCS 1102, 1996, pp. 428-432.

[BrSt90] J. Bradfield, C. Stirling: *Modal logics and mu-calculi: an introduction*, Laboratory for Foundations of Computer Science, University of Edinburgh EDH9 3JZ, United Kingdom, 1990.

[BRSS99] M. Balser, W. Reif, G. Schellhorn, K. Stenzel: *KIV 3.0 for Provably Correct Systems*. In: *Current Trends in Applied Formal Methods*, Springer Verlag, LNCS 1641, Boppard, 1999.

[Büch00] H. Büchele: *Entwicklung der Steuerungssoftware eines Bremssteuermoduls für Nutzfahrzeuge,* Studienarbeit, SADA-Nr. 1775, IAS, Universität Stuttgart, 2000.

[Canv99] E. Canver: *Einsatz von Model-Checking zur Analyse von MSCs über Statecharts*, Ulmer Informatik Berichte, Nr. 99-04, Universität Ulm, 05/1999.

[CGL94] E. M. Clarke, O. Grumberg, D. E. Long: *Model Checking and Abstraction*. In: ACM Transactions on Programming Languages and Systems, vol. 16 (1994) no 5, pp. 1512-1542.

[CGM97] E. Canver, J.-T. Gayen, A. Moik: *Formale Spezifikation von Steuerungs-software am Beispiel einer Weiche.* In: atp – Automatisierungstechnische Praxis 39 (1997) H. 5, S. 57-64.

[CJS98] M. Chouikha, A. Janhsen, E. Schnieder: *Klassifikation und Bewertung von Beschreibungsmitteln für die Automatisierungstechnik.* In: at – Automatisierungstechnik 46 (1998) H. 12, S. 582-591.

[CJMS01] G. Ciardo, R. L. Jones, A. S. Miner, R. Siminiceanu: *SMART: Stochastic Model Analyzer for Reliability and Timing.* In: Proceedings of International Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems, Aachen, Germany, Sept. 2001, pp. 29-34.

[CMZ02] A. Cau, B. Moszkowski and H. Zedan: *ITL – Interval Temporal Logic.* Software Technology Research Laboratory, 2002. http://www.cse.dmu.ac.uk/~cau/papers/itlhomepage.pdf

[DAC99] M. B. Dwyer, G. S. Avrunin, J. C. Corbett: *Patterns in Property Specification for Finite-state Verification*. In: Proceedings of the 21st International Conference on Software Engineering (ICSE99), Los Angeles, CA. ACM Press, New York, 1999, pp. 411-420.

[DAC98] M. B. Dwyer, G. S. Avrunin, J. C. Corbett: *Property Specification Patterns for Finite-state Verification.* In: Proceedings of the 2nd Workshop on Formal Methods in Software Practice, M. Ardis (ed.). ACM Press, New York, 1998, pp. 7-15.

[DaHa99] W. Damm, D. Harel: *LSCs: Breathing Life into Message Sequence Charts.* In: FMOODS'99 IFIP TC6/WG6.1 Third International Conference on Formal Methods for Open Object-Based Distributed Systems, 1999.

[DaWe03] W. Damm, B. Westphal: *Live and Let Die: LSC-based Verification of UML-Models.* In: Proceedings of the First International Symposium on Formal Methods for Components and Objects (FMCO), LNCS, Springer Verlag, Berlin, Heidelberg, New York, 2003.

[DDH+92] D. L. Dill, A. J. Drexler, A. J. Hu, C. H. Yang: *Protocol Verification as a Hardware Design Aid.* In: IEEE International Conference on Computer Design: VLSI in Computers and Processors, 1992, pp. 522-525.

[DIN66201] DIN 66201: *Begriffe bei Prozessrechensystemen*. Deutsche Industrie Norm, Berlin, 1981.

[DINEN 61508] DIN EN 61508: *Funktionale Sicherheit elektrischer/elektronischer/programmierbar elektronischer sicherheitsbezogener Systeme.* Deutsche Industrie Norm, Berlin, 2002.

[DiOl03] H. Dierks, E.-R. Olderog: *Temporale Spezifikationslogiken*. at – Automatisierungstechnik, 51(2), 2003, S. A1-A4.

[DKE03] Mitteilung des DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE: *Rechtliche Stellung der Normen der Reihe DIN EN 61508 (VDE 0803).* Frankfurt, 2003.

[Dose90] M. Dose et al.: *DUDEN – Fremdwörterbuch*. Der Duden Bd. 5, 5. Auflage, Bibliographisches Institut & F.A. Brockhaus AG, Mannheim, 1990.

[Doug04] B. P. Douglass: *Real-Time UML Third Edition – Advances in the UML for Real-Time Systems.* Addison Wesley, Boston, San Francisco, New York, 2004.

[DSW+03] W. Damm, C. Schulte, H. Wittke, M. Segelken (OFFIS); U. Higgen, M. Eckrich (BMW AG): *Formale Verifikation von ASCET Modellen im Rahmen der Entwicklung der Aktivlenkung.* GI Workshop Automotive Software Engineering, 2003.

[DSSZ] *DSSZ Model Checking Tools*. Chair of Data Structures and Software Dependability of the Computer Science Department of the University of Technology Cottbus. http://www-dssz.informatik.tu-cottbus.de/~wwwdssz/

[Dujm02] S. Dujmović: *Anwendungsentwicklung mit Komponenten-Frameworks in der Automatisierungstechnik.* Diss. IAS, Universität Stuttgart, 2002. IAS-Forschungsberichte, P. Göhner (Hrsg.). Shaker Verlag, Aachen, Band 1/2002.

[EcCa02] R. Eckstein, M.Casabianca: *XML – kurz & gut.* 2. Auflage, O'Reilly Verlag GmbH & Co. KG Beijing, Cambridge, Köln, Paris, Tokyo, 2002.

[ECS93] H. Engesser, V. Claus, A. Schwill: *Duden – Informatik*. Dudenverlag, Mannheim, Leipzig, Wien, Zürich, 1993.

[Ehri04] H. Ehrig et al. (eds.): *Integration of Software Specification Techniques for Applications in Engineering*. LNCS 3147, Springer Verlag, 2004.

[EN50126] EN 50126: *Railway Applications. The specification and demonstration of reliability, Availability, maintainability and safety (RAMS).* CENELEC, 1999.

[EN50128] EN 50128: *Railway Applications. Communications, signalling and processing systems. Software for railway control and protection systems.* CENELEC, 2001.

[EN50129] EN 50129: *Railway applications. Communication, signalling and processing systems. Safety related electronic systems for signalling.* CENELEC, 2003.

[Espa04] J. Esparza et al.: *The Model Checking Kit.* Universität Stuttgart, 2003-2004. http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db/themodelcheckingkit.html

[Espa03] J. Esparza: *Model Checking*. Vorlesung, Universität Stuttgart, 2003.

[Espa98] J. Esparza et al.: *Hinweise zu Petri-Netz-basierten Model-Checkern*. Praktikum „Automatische Verifikation reaktiver Systeme“, München, 1998.

[EsSc03] J. Esparza, S. Schwoom: *Aktuelles Schlagwort: „Model Checking“*. Newsletter, Informatik Verbund Stuttgart (IVS) der Universität Stuttgart, 7/2003.

[Faud04] J. Faude: *Entwicklung einer sicheren Steuerungssoftware für einen Bahnübergang im Funk-Fahr-Betrieb.* Diplomarbeit, SADA-Nr. 1962, IAS, Universität Stuttgart, 2004.

[FlMu02] S. Flake, W. Mueller: *Spezifikation von Echtzeiteigenschaften mit OCL für eine MFERT-basierte Fallstudie.* In: DFG-Workshop Ablaufmodellierungen in ingenieurwissenschaftlichen Anwendungen, Berlin, 2001.

[FlMu01] S. Flake, W. Mueller: *An OCL Extension for Real-Time Constraints.* In: Advances in Object Modelling with the OCL, T. Clark and J. Warmer, (eds.). Heidelberg: Springer Verlag. 2001.

[FMR00a] S. Flake, W. Müller, J. Ruf: *Structured English for Model Checking Specification*. In: Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, GI/ITG/GMM Workshop, Frankfurt, 2000, pp. 91-100.

[FMR00b] S. Flake, W. Müller, J. Ruf: *Mapping of Structured English Sentences to CCTL Formulae.* Technical Report, C-LAB Report 04/2000, Januar 2000.

[FNT04] W. Fenner, P. Naumann, J. Trinckauf: *Bahnsicherungstechnik*. Wiley-VCH Verlag, Weinheim, 2004.

[Gamm94] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Massachusetts. 1994.

[GaPa99] J.-T. Gayen, J. Pachl: *Sicherheitsanforderungen/-eigenschaften im FFB*. 1999. http://rzv113.rz.tu-bs.de/Forschung/projekte/saferail/sicherheitsanforderungen/index.htm

[Gaye99a] J.-T. Gayen: *Grundlagenbetrachtungen zum Bahnübergang im FunkFahrBetrieb (FFB)*. 1999. http://rzv113.rz.tu-bs.de/Forschung/projekte/saferail/bue-grundlagen/index.htm

[Gaye99b] J.-T. Gayen: *Glossar für FFB-Referenzfallstudie.* 1999. http://rzv113.rz.tu-bs.de/Forschung/projekte/saferail/ffb-glossar/index.htm

[Gaye99c] J.-T. Gayen: *Zugfahrten im FunkFahrBetrieb (FFB).* 1999. http://rzv113.rz.tu-bs.de/Forschung/projekte/saferail/zugfahrt/index.htm

[GHK99] J. Gil, J. Howse, S. Kent: *Constraint Diagrams: A Step Beyond UML.* In: Proceedings of TOOLS USA'99, IEEE Computer Society Press, 12/1999.

[GHKM00] C. Grover, A. Holt, E. Klein, M. Moens: *Designing a controlled language for interactive model checking*. In: Proceedings of the Third International Workshop on Controlled Language Applications – CLAW 2000, pp. 90-104.

[Gors86] J. Gorski: *Design for Safety Using Temporal Logic.* In: Proceedings of Safecomp '86, W.J. Quirk (ed.). Pergamon Press, Oxford, 1986.

[Göhn05a] P. Göhner: *Skript zur Vorlesung „Softwaretechnik II"*. IAS, Universität Stuttgart, SS 2005.

[Göhn05b] P. Göhner: *Skript zur Vorlesung „Prozessautomatisierung I"*. IAS, Universität Stuttgart, SS 2005.

[Göhn04] P. Göhner: *Skript zur Vorlesung „Einführung in die Informatik III"*. IAS, Universität Stuttgart, WS 2003/04.

[Göhn98] P. Göhner: *Komponentenbasierte Entwicklung von Automatisierungssystemen.* In: GMA-Kongress 1998, VDI-Berichte 1397. VDI Verlag GmbH, Düsseldorf, 1998, S. 513-521.

[Göhn95] P. Göhner: *Spezifikation und Verifikation von sicheren Softwaresystemen.* In: atp – Automatisierungstechnische Praxis 37 (1995) H. 4, S. 24-31.

[Gont03] M. Gontcharova: *Sicherheitsnachweis für das betriebliche Konzept eines Bahnübergangs auf der Basis eines UML-Systemmodells.* Studienarbeit, SADA-Nr. 1881, IAS, Universität Stuttgart, 2003.

[Grav95] B. Graves: *Erläuterungen zu Esparza's L1-Model-Checker*. In: PEP-Workshop. Berlin, 1994.

[Gunz03] M. Gunzert: *Komponentenbasierte Softwareentwicklung für sicherheitskritische eingebettete Systeme*. Diss. IAS, Universität Stuttgart, 2002. IAS-Forschungsberichte, P. Göhner (Hrsg.). Shaker Verlag, Aachen, Band 4/2003.

[HaJo94] H. Hansson, B. Jonsson: *A logic for reasoning about time and probability*. In: Formal Aspects of Computing, 6 (1994).

[HeHe98] M. P. E. Heimdahl, C. L. Heitmeyer: *Formal Methods for Developing High Assurance Computer Systems: Working Group Report.* In: Proceedings of the Second IEEE Workshop on Industrial-Strength Formal Techniques (WIFT'98). Boca Raton, FL, Oct. 19, 1998.

[Heit98a] C. L. Heitmeyer: *On the Need for 'Practical' Formal Methods*. In: Proceedings of Formal Techniques in Real-Time and Real-Time Fault-Tolerant Systems, 5th Intern. Symposium (FTRTFT'98), Lyngby, Denmark, LICS 1486. 1998, pp. 18-26.

[Heit98b] C. L. Heitmeyer: *SCR: A Practical Method for Requirements Specification*. In: Proceedings of 17th AIAA/IEEE/SAE Digital Avionics System Conference (DASC), Bellevue, WA, 1998.

[Herr04] M. Herrmann: *Softwareentwicklung mit UML für einen Mikrocontroller zur Steuerung eines Bahnübergangs im Funk-Fahr-Betrieb.* Studienarbeit, SADA-Nr. 1996, IAS, Universität Stuttgart, 2004.

[HKKS03] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, M. Siegle: *A tool for model-checking Markov chains*. In: Int. Journal on Software Tools for Technology Transfer, vol. 4 (2003) no. 2, pp. 153-172.

[HKL+98] C. Heitmeyer, J. Kirby, Jr., B. Labaw, M. Archer, R. Bharadwaj: *Using Abstraction and Model Checking to Detect Safety Violations in Requirements Specifications*. In: IEEE Transactions on Software Engineering, vol. 24 (1998) no. 11, pp. 927-948.

[HLB03] R. Huuck, B. Lukoschus, N. Bauer: *A Model-Checking Approach to Safe SFCs*. In: Proceedings of IMACS-IEEE CESA'03 (Computational Engineering in Systems Applications), Lille (France), 2003.

[HMD01] M. Heiner, Th. Mertke, P. Deussen: *A Safety-oriented Technical Language for the Requirement Specification in Control Engineering.* Computer Science Reports 09/01. Brandenburg University of Technology at Cottbus, 2001.

[HoKl99] A. Holt, E. Klein: *A semantically-derived subset of English for hardware verification*. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics. 1999, pp. 451-456.

[Holt99] A. Holt: *Formal verification with natural language specifications: guidelines, experiments and lessons so far*. South African Computer Journal, No 24, November 1999, pp. 253-257.

[Holz03] G. J. Holzmann: *The Spin Model Checker: Primer and Reference Manual.* Addison-Wesley, 2003.

[Hoss03] R. Hossain: *Development of an Instantiation Tool for Safety Requirements Specification.* Master Thesis, SADA-Nr. 1912, IAS, Universität Stuttgart, 2003.

[HPSS04] F. Hänsel, J. Poliak, R. Slovák, E. Schnieder: *Reference case study for comparison and validation of formal specifications using a model demonstrator.* In: Integration of Software Specification Techniques for Applications in Engineering, H. Ehrig et al. (eds.). LNCS 3147, Springer Verlag, Berlin, Heidelberg, New York, 2004, pp. 96-118.

[HuRy00] M. Huth, M. Ryan: *Logic in Computer Science – Modelling and reasoning about systems.* Cambridge University press, Cambridge. 2000.

[HuSe00] Huber, S.; Seidl, H. G.: *Architecture Centric Development – generating more code from your UML models*. White Paper, Aonix Europe Ltd., 2000.

[IEC61508] International Standard IEC 61508: *Functional Safety of Electrical/Electronic/ Programmable Electronic Safety Related Systems*. International Electrotechnical Commission, Geneva, Switzerland, 2000.

[IEC61511] International Standard IEC 61511: *Functional Safety – Safety Instrumented Systems for the Process Industry Sector*. International Electrotechnical Commission, Geneva, Switzerland, 2004.

[INA] *INA – Integrated Net Analyzer*. Lehrstuhl für Automaten und Systemtheorie, Institut für Informatik, Humboldt-Universität zu Berlin. http://www.informatik.hu-berlin.de/~starke/ina.html

[ITU93] *ITU-T Recommendation Z.120 : Message Sequence Charts (MSC).* ITU-T, Geneva, 1993.

[Jans99a] L. Jansen: *Referenzfallstudie Bahnübergang – Referenzfallstudie im Bereich Verkehrsleittechnik des DFG-SPP Softwarespezifikation.* Version 1.1, Institut für Regelungs- und Automatisierungstechnik, Technische Universität Braunschweig, 1999.

[Jans99b] L. Jansen: *Protokoll des Arbeitstreffens zur Abstimmung der Modellierung im Themenbereich „Referenzfallstudie Verkehrsleittechnik“ am 15.07.1999.* Institut für Regelungs- und Automatisierungstechnik, Technische Universität Braunschweig, 1999.

[Jans99c] L. Jansen: *Protokoll des Arbeitstreffens zur Abstimmung der Modellierung im Themenbereich „Referenzfallstudie Verkehrsleittechnik“ am 15.07.1999.. (Ergänzung).* Institut für Regelungs- und Automatisierungstechnik, Technische Universität Braunschweig, 1999.

[JaSc00] L. Jansen, E. Schnieder: *Traffic Control Systems Case Study: Problem Description and a Note on Domain-based Software Specification.* In: INT 2000 Integration of Specification Techniques with Applications in Engineering, H. Ehrig, M. Große-Rhode, F. Orejas (Hrsg.). TU Berlin, 2000, S. 41-47.

[Jime03] B. Jimenez: *Graphical Simulation of Generic Safety Requirements.* Master Thesis, SADA-Nr. 1905, IAS, Universität Stuttgart, 2003.

[Kaiv97] R. Kaivola: *Using Automata to Characterise Fixed Point Temporal Logics*. Doctor of Philosophy, LFCS report ECS-LFCS-97-356, University of Edinburgh, 1997.

[Karu04] S. Karuturi: *Development of a Software Tool for Safety Requirements Instantiation in Graphical Notation.* Master Thesis, SADA-Nr. 1978, IAS, Universität Stuttgart, 2004.

[KBW00] P. Kunzmann, F.-P. Burkard, F. Wiedmann: *dtv-Atlas Philosophie*. 9. Auflage, Deutscher Taschenbuch Verlag, München, 2000.

[KlMø01] N. Klarlund, A. Møller*: MONA Version 1.4 – User Manual*. BRICS – Basic Research in Computer Science, Department of Computer Science, University of Aarhus, Denmark, 2001.

[KnMe02] A. Knapp, S. Merz: *Model Checking and Code Generation for UML State Machines and Collaborations.* In: Proceedings of 5th Workshop Tools for System Design and Verification*,* D. Haneberg, G. Schellhorn and W. Reif (eds.). Technical Report 2002-11, Institut für Informatik, Universität Augsburg, 2002, pp. 59-64.

[KNP01] M. Kwiatkowska, G. Norman and D. Parker: *PRISM: Probabilistic Symbolic Model Checker.* In: Proceedings of PAPM/PROBMIV'01, Tools Session. Technical Report 760/2001, University of Dortmund, 2001.

[Kona03] R. Konakovsky: *Skript zur Vorlesung „Zuverlässigkeit und Sicherheit von Automatisierungssystemen"*. IAS, Universität Stuttgart, 2003.

[Lako70] G. Lakoff: *Linguistik und natürliche Logik*. Athenäum Verlag, Frankfurt, 1970.

[LaGö99a] R. Lauber, P. Göhner: *Prozessautomatisierung I*. 3. Auflage, Springer Verlag, Berlin, Heidelberg, New York, 1999.

[LaGö99b] R. Lauber, P. Göhner: *Prozessautomatisierung II*. 1. Auflage, Springer Verlag, Berlin, Heidelberg, New York, 1999.

[Lamp94] L. Lamport: *Introduction to TLA*. HP Labs Technical Reports, SRC-TN-1994-001, 1994.
http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-TN-1994-001.pdf

[Leve95] N. Leveson: *Safeware – System, Safety and Computers*. Addison-Wesley Publishing Company, Massachusetts, Bonn, 1995.

[Lieb04] R. Liebi: *Herkunft und Entwicklung der Sprachen*. 2. Auflage, Hänssler Verlag, Holzgerlingen, 2004.

[Link03] S. Link: *UML-Entwurf und Codegenerierung der Steuerungssoftware eines Bahnübergangs im Funk-Fahr-Betrieb.* Wahlstudienarbeit, SADA-Nr. 1879, IAS, Universität Stuttgart, 2003.

[LoLA] *Low Level Petri net Analyzer.* Institut für Informatik Humboldt-Universität zu Berlin.
http://www.informatik.hu-berlin.de/top/lola/lola.html

[Lova03] C. Lovasi: *Development of a Dialog System for the Correct Specification of Safety Requirements.* Master Thesis, SADA-Nr. 1857, IAS, Universität Stuttgart, 2003.

[MaPn95] Z. Manna, A. Pnueli: *Temporal Verification of Reactive Systems – Safety*. Springer Verlag, NewYork, 1995.

[Maria] *Maria – The Modular Reachability Analyzer.* Laboratory for Theoretical Computer Science, Helsinki University of Technology.
http://www.tcs.hut.fi/Software/maria/

[MCB] *CSML and MCB.* Model Checking Group, Specification and Verification Center, Carnegie Mellon University, Pittsburgh. http://www-2.cs.cmu.edu/~modelcheck/csml.html

[McMi97] K. L. McMillan*: Lecture notes for NATO summer school on verification of digital and hybrid systems.* 1997. http://www-cad.eecs.berkeley.edu/~kenmcmil/tutorial/toc.html

[McMi92] K. L. McMillan: *Symbolic model checking – an approach to the state explosion problem*. PhD thesis, SCS, Carnegie Mellon University, 1992.

[MDH01] Th. Mertke, P. Deussen, M. Heiner: *Eine anwenderorientierte Sicherheitsfachsprache zur Verifikation von Steuerungsprogrammen.* In: Engineering komplexer Automatisierungssysteme (EKA), E. Schnieder (Hrsg.). 2001, S. 297-309.

[Moik02] A. Moik: *Ingenieurgerechte formale Methoden für die Entwicklung von sicheren Automatisierungssystemen*. Diss. IAS, Universität Stuttgart, 2002. IAS-Forschungsberichte, P. Göhner (Hrsg.). Shaker Verlag, Aachen, Band 3/2002.

[Mucke] *Mucke*. Institute for Formal Models and Verification, Johannes Kepler Universität Linz. http://fmv.jku.at/mucke/

[Murphi] *Murphi description language and verifier*. Formal Verification Group, Stanford University. http://sprout.stanford.edu/dill/murphi.html

[Näge98] A. Nägele: *Entwicklung und Verifikation synchroner Software für einen TTP-basierten Brake-by-Wire Manager.* Diplomarbeit, SADA-Nr. 1663, IAS, Universität Stuttgart, 1998.

[NPW02] T. Nipkow, L. C. Paulson, M. Wenzel: *Isabelle / HOL – A Proof Assistant for Higher Order Logic.* Tutorial, LNCS 2283, Springer Verlag, 2002.

[ORSS98] S. Owre, J. Rushby, N. Shankar, D. Stringer-Calvert: *PVS: An Experience Report.* In: Proceedings of Applied Formal Methods - FM-Trends 98, D. Hutter, W. Stephan, P. Traverso, and M. Ullman (eds.), LNCS 1641, Springer Verlag, 1998, pp. 338-345.

[Ortn97] E. Ortner: *Methodenneutraler Fachentwurf – Zu den Grundlagen einer anwendungsorientierten Informatik.*B. G. Teubner Verlagsgesellschaft, Stuttgart, Leipzig ,1997.

[Ostr97] J. Ostroff: *A Visual Toolset fort he Design of Real-Time Discrete-Event Systems.* IEEE Trans. Control Systems Technology. Vol. 5 (1997) no. 3, pp. 320-337.

[PaSc99] S. Parthasarathy, E. Schnieder: *The explication problem: Achille's heel of formal methods.* In: Entwicklung und Betrieb komplexer Automatisierungssysteme (EKA), E. Schnieder (Hrsg.). Braunschweig, 1999, S. 93-103.

[PEP] *PEP (Programming Environment based on Petri Nets) Homepage*. Abteilung Theoretischen Informatik, Universität Oldenburg. http://theoretica.informatik.uni-oldenburg.de/~pep/

[Pere95] C. E. Pereira: *Verfahren zur Beschreibung und Überprüfung von Zeitbedingungen bei der objektorientierten Entwicklung von Automatisierungssystemen.* Diss. IAS, Universität Stuttgart, 1995.

[Ping02] R. Pinger: *Kompositionale Verifikation nebenläufiger Softwaremodelle durch Model-Checking*. Diss. Institut für Software-Abteilung Informationssysteme, Technischen Universität Braunschweig, 2002.

[Pnue77] A. Pnueli: *The Temporal Logic of Programs.* In: Proceedings of the 18th IEEE Symposium Foundations of Computer Science (FOCS 1977), pp. 46-57.

[PoHa02] A. Porzig, J. Hartmann: *Folgenanalyse am Beispiel der Risikoanalyse Funk-Fahr-Betrieb.* In: Moderne Sicherheitsanalysen – Theorie und Praxis, J. Pachl (Hrsg.). IfEV-Schriftenreihe, TU Braunschweig, Heft 64, 2001, S. 33-50.

[Prio62] A. Prior: *Tense-logic and the continuity of Time*. Studia Logica, vol. 13 (1962), pp. 133-148.

[PROD] *PROD Homepage*. Formal Methods Group of the Laboratory for Theoretical Computer Science at the Helsinki University of Technology. http://www.tcs.hut.fi/Software/prod/

[Raven] *Raven – Real-Time Analysis and Verification Environment*. Formal Methods Group, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen. http://www-ti.informatik.uni-tuebingen.de/~fmg/raven/raven.html

[RBH+01] W.-P. de Roever, F. de Boer, U. Hannemann, J. Hooman, J. Lakhnech, M. Poel, J. Zwiers: *Concurrency Verification – Introduction to Compositional and Noncompositional Methods.* Cambridge University Press, Cambridge, New York, 2001.

[Renp02] F. Renpenning: *Gefährdungsanalyse am Beispiel des FunkFahrBetriebs.* In: Moderne Sicherheitsanalysen – Theorie und Praxis, J. Pachl (Hrsg.). IfEV-Schriftenreihe, TU Braunschweig, Heft 64, 2001, S. 51-80.

[Reff00] F. Reffel: *Übersetzung von CTL*-Formeln in den μ-Kalkül.* Diplomarbeit, Institut für Logik, Komplexität und Deduktionssysteme, Lehrstuhl Prof. Dr. P. Deussen, Universität Karlsruhe, 1996.

[RFP00] J. Ruf, S. Flake, U. Pape: *Modellprüfung für den Entwurf von Produktionssteuerungssystemen.* In: 3. Gesamtworkshop des DFG SPP „Integration von Techniken der Softwarespezifikation für ingenieur-wissenschaftliche Anwendungen“ Universität Ulm, 14. November 2000.

[RiGo98] M. Richters, M. Gogolla: *On Formalizing the UML Object Constraint Language OCL.* In: Proceedings of 17th International Conference Conceptual Modeling (ER’98), Tok Wang Ling (ed.). LNCS 1507, Springer Verlag, 1998, pp. 449-464.

[RuKr97] J. Ruf, T. Kropf: *Symbolic Model Checking for a Discrete Clocked Temporal Logic with Intervals*. In: Conference on Correct Hardware Design and Verification Methods (CHARME), E. Cerny, D. Probst (eds.). IFIP WG 10.5, Chapman and Hall, 1997, pp. 146-166.

[Rush00] J. Rushby: *Disappearing Formal Methods*. Invited paper from the Fifth IEEE International Symposium on High Assurance Systems Engineering (HASE), Albuquerque NM, November 2000, pp. 95-96.

[Rush95] J. Rushby: *Formal Methods and their Role in the Certification of Critical Systems.* Technical Report CSL-95-1. SRI International, 1995.

[ScDa93] R. Schlör, W. Damm: *Specification and verification of system-level hardware designs using timing diagrams*, In: Proceedings of the European Conference on Design Automation. IEEE Computer Society Press, 1993, pp. 518-524.

[ScEi99] E. Schnieder, S. Einer: *Formale Techniken für die Eisenbahnsicherungstechnik - Anforderungskatalog.* In: FORMS 99 – Formale Techniken für die Eisenbahnsicherung, E. Schnieder (Hrsg.). Fortschritt-Berichte VDI, Reihe 12, Nr.436. VDI Verlag GmbH, Düsseldorf, 2000, S. 1-23.

[Schi97] E. Schienmann: *Objektorientierter Fachentwurf.* B. G. Teubner Verlagsgesellschaft, Stuttgart, Leipzig, 1997.

[Schn99] E. Schnieder: *Methoden der Automatisierung*. Vieweg & Sohn Verlagsgesellschaft, Braunschweig/Wiesbaden, 1999.

[SJW98] Schlör, R., Josko, B., Werth D.: *Using a visual formalism for design verification in industrial environments.* In: VISUAL'98, LNCS 1385, Springer Verlag, Berlin, Heidelberg, 1998, pp. 208-221.

[Stec03] T. Stecker: Entwicklung der Funkkommunikation für den Demonstrator eines Bahnübergangs im Funk-Fahr-Betrieb. Wahlstudienarbeit, SADA-Nr. 1880, IAS, Universität Stuttgart, 2003.

[Stei97] P. Steiner: *Semiotik, Semantik, Pragmatik – eine Einführung*. Proseminar „Semantik und Pragmatik". Arbeitsbereich Linguistik, Westfälische Wilhelms-Universität Münster, 1997. http://santana.uni-muenster.de/Linguistik/user/steiner/semindex/intro.html

[Step00] S. Stephen: *VIS*. Vortrag in der Vorlesung „Model-Checking Tools", ETH Zürich, 2000.

[Stor96] N. Storey: *Safety Critical Computer Systems*. Prentice Hall, Harlow, London, New York, 1996.

[Szyp97] C. Szyperski: *Component Software*. Addison-Wesley, 1997.

[Tche03] K. Tchekine: *Specification of Safety Pattern using the μ-Calculus.* Master Thesis im International Masters Programme in Computational Logic, Dresden University of Technology Computer Science Department, 2003.

[Teuf04] B. Teufel: *Unterstützung der Spezifikation von Sicherheitsanforderungen durch grafische Simulationen.* Studienarbeit, SADA-Nr. 1954, IAS, Universität Stuttgart, 2004.

[Teuf03] B. Teufel: *Fernvisualisierung für das verteilte System eingleisiger Bahnübergang im Funk-Fahr-Betrieb.* Wahlstudienarbeit, SADA-Nr. 1920, IAS, Universität Stuttgart, 2003.

[Tidh00] D. Tidhar: *ALVEY to CTL translation - a preparatory study for a finite-state verification Natural Language interface.* MSc Dissertation, Department of Linguistics, University of Edinburgh, 2000.

[Touc04] J.-L. Touchevier: *Extension of a Web-based Software Tool for the Generation of Generic Safety Requirements.* Wahlstudienarbeit, SADA-Nr.1990, IAS, Universität Stuttgart, 2004.

[VDA96] *Sicherung der Qualität vor Serieneinsatz – FMEA.* VDA-Band 4, Teil 2, Verband der Automobilindustrie e.V., Frankfurt, 1996.

[VSS] T. Villa, G. Swamy, T. Shiple. *VIS User's Manual*. The VIS Group, University of California, Berkeley.

[Warl04] J. Warlitz: *Built-in Safety: Europäische Eisenban-Normen im Wandel*. Alcatel Telecom Rundschau – Transportation Automation Solutions, 2004, S. 141-147.

[WaKl99] Warmer, J.B., Kleppe, A.G.: *The Object Constraint Language*. Addison-Wesley, 1999.

[Webe04] L. Weber: *Extension of a Web-Based Software Tool for Safety Patterns Selection.* Wahlstudienarbeit, SADA-Nr. 1991, IAS, Universität Stuttgart, 2004.

[WiMe01] M. Wirsing, S. Merz: *Skript zur Vorlesung "Grundlagen der Systementwicklung (PG)"*. Institut für Informatik, Lehr- und Forschungseinheit für Programmierung und Softwaretechnik, Universität München, 2001.

[WWS02] J. Warlitz, A. Wozniak, K.-H. Suwe: *Ist das Konzept der Prüfleitstellen auf Europa übertragbar?* In: Signal + Draht 94(2002)3, S. 6-9.

[Yovi02] S. Yovine: *Kronos: A Verification Tool for Real-Time Systems. User's guide*. Version 2.5i, VERIMAG, France, 2002.

[Zimm02] E. Zimmer: *Entwicklung eines Demonstrators zur Prüfung von Steuerungs-modellen für einen Bahnübergang im Funk-Fahr-Betrieb.* Diplomarbeit, SADA-Nr. 1863, IAS, Universität Stuttgart, 2002.

Anhang

A Syntax von CTL und Umformungsregeln

Syntax von CTL (mit p für aussagenlogischen Ausdruck und ϕ sowie ψ für CTL-Ausdruck):

$$
\begin{aligned}
\phi ::= \; & p \;\mid \\
& (\neg\phi) \;\mid\; (\phi \vee \psi) \;\mid\; (\phi \wedge \psi) \;\mid\; (\phi \rightarrow \psi) \;\mid \\
& \mathbf{AF}\ \phi \mid \mathbf{EF}\ \phi \mid \mathbf{AG}\ \phi \mid \mathbf{EG}\ \phi \mid \mathbf{AX}\ \phi \mid \mathbf{EX}\ \phi \mid \\
& \mathbf{A}[\phi\ \mathbf{U}\ \psi] \mid \mathbf{E}[\phi\ \mathbf{U}\ \psi] \mid \mathbf{A}[\phi\ \mathbf{W}\ \psi] \mid \mathbf{E}[\phi\ \mathbf{W}\ \psi]
\end{aligned}
$$

Neben allen Äquivalenzen der Aussagenlogik gelten zur äquivalenten Umformung von Ausdrücken die folgenden Regeln. Dabei ist T eine Konstante mit dem Wert „wahr“.

- $\neg\mathbf{AF}\ \phi \equiv \mathbf{EG}\ \neg\phi$
- $\neg\mathbf{EF}\ \phi \equiv \mathbf{AG}\ \neg\phi$
- $\neg\mathbf{AX}\ \phi \equiv \mathbf{EX}\ \neg\phi$
- $\mathbf{AF}\ \phi \equiv \mathbf{A}[\mathrm{T}\ \mathbf{U}\ \phi]$
- $\mathbf{EF}\ \phi \equiv \mathbf{E}[\mathrm{T}\ \mathbf{U}\ \phi]$
- $\mathbf{A}[\phi\ \mathbf{U}\ \psi] \equiv \neg(\mathbf{E}[\neg\psi\ \mathbf{U}\ (\neg\phi \wedge \neg\psi)] \vee \mathbf{EG}\ \neg\psi)$
- $\mathbf{E}[\phi\ \mathbf{W}\ \psi] \equiv \mathbf{E}[\phi\ \mathbf{U}\ \psi] \vee \mathbf{EG}\ \phi$
- $\mathbf{A}[\phi\ \mathbf{W}\ \psi] \equiv \neg\mathbf{E}[\neg\phi\ \mathbf{U}\ \neg(\phi \wedge \psi)]$

Fixpunkt-Charakterisierung der temporalen CTL-Operatoren:

- $\mathbf{AG}\ \phi \equiv \phi \wedge \mathbf{AX}\ \mathbf{AG}\ \phi$
- $\mathbf{EG}\ \phi \equiv \phi \wedge \mathbf{EX}\ \mathbf{EG}\ \phi$
- $\mathbf{AF}\ \phi \equiv \phi \wedge \mathbf{AX}\ \mathbf{AF}\ \phi$
- $\mathbf{EF}\ \phi \equiv \phi \wedge \mathbf{EX}\ \mathbf{EF}\ \phi$
- $\mathbf{A}[\phi\ \mathbf{U}\ \psi] \equiv \psi \vee (\phi \wedge \mathbf{AX}\ \mathbf{A}\ [\phi\ \mathbf{U}\ \psi])$
- $\mathbf{E}[\phi\ \mathbf{U}\ \psi] \equiv \psi \vee (\phi \wedge \mathbf{EX}\ \mathbf{E}\ [\phi\ \mathbf{U}\ \psi])$

B Ausschnitt aus der Safety-Pattern-Klassifikation

Auf Basis der Menge der notwendigen generischen Formulierungen und der in Abschnitt 6.1.3 beschriebenen Anforderungen wurde die Safety-Pattern-Klassifikation bestimmt. Im Folgenden wird beispielhaft ein Ausschnitt aus der Klassifikation gezeigt, wie sie im Hilfesystem von SAPIS (vgl. Abschnitt 7.3) erläutert wird. Eine Auswahl von Klassifikationskriterien mit den dazugehörigen Klassen wird dargelegt. Hierfür werden jeweils Erklärungen für das Klassifikationskriterium und anschließend für die dazugehörigen Klassen gegeben. Bei jeder Klasse wird die Erläuterung zum einen anhand des Beispiels einer typischen generischen Formulierung für diese Klasse verdeutlicht. Zum anderen wird ein konkretes Anwendungsbeispiel einer Sicherheitsanforderung gegeben, die dieser Klasse zugeordnet ist.

Klassifikationskriterium 1: ***Vorhandensein eines zeitlich-logischen Aspekts in der Sicherheitsanforderung***

In Sicherheitsanforderungen können Aussagen über Eigenschaften, die ohne Bedingung jederzeit erfüllt sein müssen (statische Geltung), oder über Eigenschaften, die unter bestimmten Bedingungen gelten müssen (dynamische Geltung), ausgedrückt werden.

Klasse 1.1: ***Statische Sicherheitsanforderungen***

Diese Klasse umfasst die Sicherheitsanforderungen, bei denen es um die Geltung einer Eigenschaft in allen Zuständen des Zustandsraums des Modells der Systemfunktionen geht. Diese Sicherheitsanforderungen besitzen weder zeitliche Beziehungen noch Implikationen (dies sind nicht-zeitliche „wenn-dann“-Beziehungen, vgl. Abschnitt 2.4.1.1, wie z. B. „wenn die Aussage `p` erfüllt ist, dann muss auch die Aussage `q` erfüllt sein“). Sie besitzen somit ausschließlich Aussagen ohne Bedingungen. Das bedeutet, dass zu jedem Zeitpunkt eine Eigenschaft `p` erfüllt ist, ohne dass dafür irgendeine Bedingung erfüllt sein muss.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss während der gesamten Systemlaufzeit erfüllt sein. In dem Beispiel muss `b` in allen Zuständen des Zustandsraums gelten, ohne dass hierfür eine bestimmte Bedingung erfüllt sein muss.

Konkretes Beispiel, Sicherheitsanforderung an eine Verkehrsampel: *In allen Systemsituationen muss gelten, dass die Lichtzeichen der Ampeln der Haupt- und der Nebenstraße nicht gleichzeitig grün sind.* In diesem Beispiel muss in allen Zuständen des Zustandsraums die Eigenschaft erfüllt sein, dass die Lichtzeichen der Ampel der Hauptstraße und die der Nebenstraße nicht gleichzeitig grün sind.

Klasse 1.2: ***Dynamische Sicherheitsanforderungen***

Dieser Klasse sind die Sicherheitsanforderungen zugeordnet, bei denen es um die Geltung einer Eigenschaft in bestimmten Zuständen des Zustandsraums des Modells der Systemfunktionen geht. Die geforderte Eigenschaft gilt also nicht statisch in allen Systemzuständen, sondern besitzt eine dynamische Geltung.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` darf nur ab dem Zeitpunkt, zu dem `a` erfüllt ist, gültig sein. In dem Beispiel darf `b` nur zu bestimmten Zuständen erfüllt sein, nämlich ausschließlich in solchen, die auf einen Systemzustand folgen, in dem `a` erfüllt ist.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb: *Ab dem Zeitpunkt, ab dem der Bahnübergang sich im Defektzustand befindet, darf er nicht die Meldung „gesichert“ an das Eisenbahnfahrzeug senden, bis der Bahnübergang repariert worden ist.* In dieser Sicherheitsanforderung darf die Meldung „gesichert“ nicht in allen, sondern nur in bestimmten Systemzuständen vom Bahnübergang an das Eisenbahnfahrzeug gesendet werden.

Klassifikationskriterium 2: ***Art der Zeitspezifikation***

Eine zeitliche Aussage kann formuliert werden, indem explizite Zeiten genannt werden oder indem Aussagen über die zeitliche Relation zwischen Aussagen gemacht werden. Im ersten Fall werden quantitative Zeitaussagen und im zweiten Fall qualitative Zeitaussagen formuliert.

Klasse 2.1: ***Sicherheitsanforderungen ausschließlich mit impliziten Zeitangaben***

In den Sicherheitsanforderungen dieser Klasse gibt es keine Aussage in Abhängigkeit zu einer Systemuhr. Stattdessen sind der Beginn und das zeitliche Ende von Aussagen von der Gültigkeit anderer Aussagen abhängig.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss bis zu dem Zeitpunkt, zu dem `a` gültig ist, mindestens `n`-mal gültig sein. In dem Beispiel wird keine quantitative Zeitangabe gemacht, sondern eine qualitative Zeitangabe, die durch die temporale Relation zwischen `a` und `b` ausgedrückt wird.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb: *Erhält das Eisenbahnfahrzeug vom Bahnübergang die Meldung „Bahnübergang_nicht_gesichert“, so muss das Eisenbahnfahrzeug vor dem Bahnübergang zum Halten kommen.* In dieser Sicherheitsanforderung besteht eine zeitliche Relation zwischen dem Erhalt der Meldung „*Bahnübergang_nicht_gesichert*“ einerseits und dem Halt des Eisenbahnfahrzeugs vor dem Bahnübergang andererseits. Eine explizite Zeitangabe gibt es nicht.

Klasse 2.2: ***Sicherheitsanforderungen mit expliziten Zeitangaben***

In den Sicherheitsanforderungen dieser Klasse gibt es mindestens eine Aussage in Abhängigkeit zu einer Systemuhr.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: Wenn nach der Zeit `T` mit `T1 < T < T2`, die ab dem Zeitpunkt, zu dem `b` gilt, gemessen wird, `c` erfüllt ist, so muss gleichzeitig zu `c` auch `a` gelten. Für die temporale Relation zwischen `b` einerseits sowie `a` und `c` andererseits wird mit `T` eine quantitative Zeitangabe gemacht.

Konkretes Beispiel, Sicherheitsanforderung an eine S-Bahn-Fahrzeugtür: *Tritt das Signal S_1 ein, so muss der Warnsignalgeber für eine Vorwarnzeit von 5 Sekunden eingeschaltet sein.* In dieser Sicherheitsanforderung wird mit der Vorwarnzeit eine explizite Zeitdauer ausgedrückt, die den quantitativen Wert von 5 Sekunden umfassen soll.

Klassifikationskriterium 5 : Zeitliche Begrenzung der Geltung

Bei dynamischen Sicherheitsanforderungen muss spezifiziert werden, ob die Geltung der geforderten Eigenschaft unter bestimmten Bedingungen beginnen oder enden muss oder zu besimmten Zeitpunkten gelten soll.

Klasse 5.1: ***Gültigkeitsdauer***

Bei dieser Klasse geht es um die Geltung einer Eigenschaft bis zu einem bestimmten Zeitpunkt, vgl. Abbildung B.1.

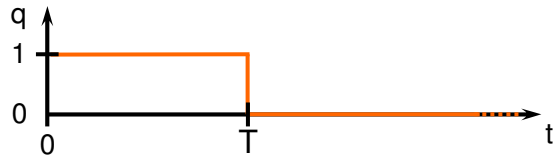


Abbildung B.1: Gültigkeitsdauer einer Eigenschaft `q` bis zum Zeitpunkt `t = T`

Zu dieser Klasse gehören die Sicherheitsanforderungen, welche die Dauer beziehungsweise das Ende der Gültigkeit einer Eigenschaft beschreiben.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss bis zu dem Zeitpunkt, zu dem `a` gültig ist, mindestens `n`-mal gültig sein.

Konkretes Beispiel, Sicherheitsanforderung an eine Destillationskolonne: *Der Einlauf darf nur solange geöffnet sein, bis der Temperatursensor einen höheren Wert als 400 K ermittelt.*

Klasse 5.2: ***Gültigkeitsbeginn oder Gültigkeitszeitpunkte***

Der Gültigkeitsbeginn beschreibt die Geltung einer Eigenschaft nach einem bestimmten Zeitpunkt, vgl. Abbildung B.2.

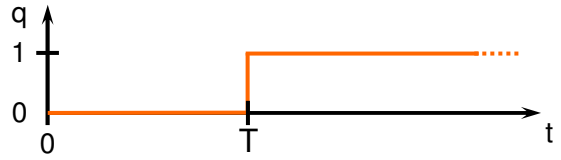


Abbildung B.2: Gültigkeitsbeginn einer Eigenschaft `q` nach dem Zeitpunkt `t = T`

Zu dieser Klasse gehören die Sicherheitsanforderungen, die den Beginn der Gültigkeit einer Eigenschaft beschreiben oder Zeitpunkte, zu denen eine Eigenschaft gelten soll.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss strikt nach dem Zeitpunkt, zu dem `a` gültig ist, mindestens `n`-mal gültig sein.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang: *Wenn ein Eisenbahnfahrzeug einen Bahnübergang befahren hat, darf die Sicherung des Bahnübergangs nur dann aufgehoben werden, strikt nachdem der Bahnübergang vollständig geräumt wurde.* Der Beginn der Aufhebung der Sicherung hängt zeitlich von der zuvor abgeschlossenen kompletten Räumung des Bahnübergangs ab.

Klasse 5.3: ***Gültigkeitsbeginn und -dauer***

Diese Klasse beinhaltet solche Safety-Pattern, mit denen die Geltung einer Eigenschaft zwischen zwei Zeitpunkten [T1, T2] spezifiziert werden kann, vgl. Abbildung B.3.

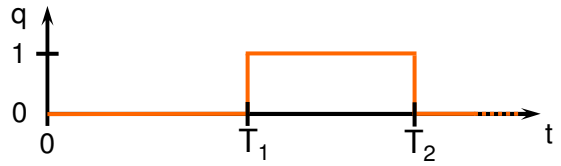


Abbildung B.3: Gültigkeit von `q` nach dem Zeitpunkt `t = T`$_1$ und bis zum Zeitpunkt `t = T`$_2$

Zu dieser Klasse gehören die Sicherheitsanforderungen, die sowohl den Beginn der Gültigkeit einer Eigenschaft als auch die Gültigkeitsdauer beschreiben.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss strikt nach dem Zeitpunkt, zu dem `a` gültig ist, permanent gültig sein, bis `c` eintritt.

Konkretes Beispiel, Sicherheitsanforderung an eine Destillationskolonne: *Nur nachdem der Temperatursensor eine Temperatur von 350 K ermittelt hat, darf von dem Zeitpunkt an der Zufluss solange geöffnet sein, bis der Tank den minimalen Füllstand erreicht.*

Klassifikationskriterium 7: ***Modalität der Forderung***

Sicherheitsanforderungen unterscheiden sich in der Modalität ihrer Forderung, d. h. in der Art und Weise der Forderung. Zum einen muss entweder eine Eigenschaft unter gewissen Bedingungen notwendigerweise gelten oder darf ausschließlich unter bestimmten Konditionen gültig sein. Zum anderen kann die Gültigkeit oder die Erreichbarkeit einer Eigenschaft gefordert sein.

Klasse 7.1: ***Notwendigkeit der Geltung***

In diesen Sicherheitsanforderungen geht es um die notwendige Geltung einer Eigenschaft unter bestimmten Bedingungen.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss strikt nach dem Zeitpunkt, zu dem `a` gültig ist, mindestens `n`-mal gültig sein.

Konkretes Beispiel, Sicherheitsanforderung für ein Bremssteuermodul: *Bei der Vorgabe eines Bremswunsches mit dem Druck p muss das Einlassventil während der Zeit T mit genau n Druckluft-Impulsen bestromt werden.*

Klasse 7.2: ***Zulässige Geltung***

In diesen Sicherheitsanforderungen wird beschrieben, unter welchen Voraussetzungen eine Eigenschaft gültig sein darf (Erlaubnis) oder nicht gültig sein darf (Verbot).

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` darf strikt nach dem Zeitpunkt, zu dem `a` gültig ist, maximal `n`-mal gültig sein.

Konkretes Beispiel, Sicherheitsanforderung für ein Bremssteuermodul: *Das Schnellentlüften darf nicht während einer Bremsung erfolgen.*

Klasse 7.3: ***Zulässige und notwendige Geltung***

In diesen Sicherheitsanforderungen wird zum einen ein notwendiges Verhalten gefordert und zum anderen beschrieben, unter welchen Voraussetzungen die Eigenschaft gültig sein darf (Erlaubnis) oder nicht gültig sein darf (Verbot).

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss strikt nach dem Zeitpunkt, zu dem `a` gültig ist, genau `n`-mal und darf nur genau `n`-mal gültig sein.

Konkretes Beispiel, Sicherheitsanforderung für ein Bremssteuermodul: *Bei der Vorgabe eines Bremswunsches mit dem Druck p_{max} muss das Einlassventil während der Zeit T bestromt werden und darf während dieser Zeit nur mit genau n Druckluft-Impulsen bestromt werden.*

Klasse 7.4: ***Notwendige Erreichbarkeit***

In diesen Sicherheitsanforderungen wird nicht die notwendige Geltung einer Eigenschaft betrachtet, sondern die erforderliche Erreichbarkeit einer bestimmten Eigenschaft. Unter bestimmten zeitlichen Bedingungen soll die Erreichbarkeit zugesichert werden.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss strikt nach dem Zeitpunkt, zu dem `a` gültig ist, mindestens `n`-mal erreichbar sein.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang: *Von allen Systemsituationen aus, in denen die Schranken geschlossen sind, muss eine Systemsituation erreicht werden können, in der die Schranken geöffnet sind.*

Klasse 7.5: ***Zulässige Erreichbarkeit***

In den Sicherheitsanforderungen dieser Klasse wird beschrieben, von welchen Zuständen aus eine bestimmte Eigenschaft erreicht werden darf und wann die Eigenschaft nicht erreicht werden darf.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` darf strikt nach dem Zeitpunkt, zu dem `a` gültig ist, maximal `n`-mal erreichbar sein.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb: *Nur wenn der Bahnübergang nicht reserviert ist, darf er außer Betrieb genommen werden können.*

Klasse 7.6: ***Zulässige und notwendige Erreichbarkeit***

Diese Klasse ist eine Kombination der Klassen 7.4 und 7.5. In den Sicherheitsanforderungen dieser Klasse wird also beschrieben, von welchen Zuständen aus eine bestimmte Eigenschaft erreicht werden muss und wann diese Eigenschaft erreicht werden darf.

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss strikt nach dem Zeitpunkt, zu dem `a` gültig ist, genau `n`-mal und darf nur genau `n`-mal erreichbar sein.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb: *Wenn keine Zugnummer registriert ist, muss der Bahnübergang außer Betrieb genommen werden können, und nur in diesem Fall darf er außer Betrieb genommen werden können.*

Klassifikationskriterium 9: ***Art des Gültigkeits-/Erreichbarkeitsbeginns***

Für Sicherheitsanforderungen über den Gültigkeits- bzw. Erreichbarkeitsbeginn einer Eigenschaft muss spezifiziert werden, ab wann genau die Geltung bzw. Erreichbarkeit der Eigenschaft beginnen muss: *ab*, *strikt nach*, *exakt nach* oder *zu* einem bestimmten Zeitpunkt.

Klasse 9.1: ***Gültigkeit / Erreichbarkeit ab bestimmtem Zeitpunkt***

Diese Art von Sicherheitsanforderungen beschreiben eine Aussage q, die von dem Zeitpunkt an (Zustand) gültig ist, zu dem eine Aussage p gültig ist (vgl. Abbildung B.4).

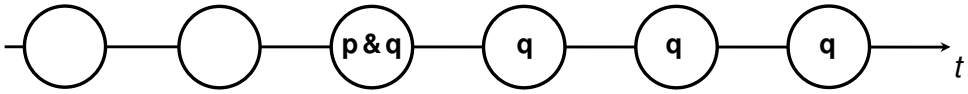


Abbildung B.4: Gültigkeit von q ab dem Zeitpunkt, zu dem p eintritt

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: b muss ab dem Zeitpunkt, zu dem a gültig ist, mindestens n-mal gültig sein. Zu dem Zeitpunkt, zu dem a gilt, und strikt danach darf b gültig sein.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb: *Ab dem Zeitpunkt, ab dem der Bahnübergang sich im Defektzustand befindet, darf er nicht die Meldung „gesichert“ an das Eisenbahnfahrzeug senden, bis der Bahnübergang repariert worden ist.* Zu dem Zeitpunkt, zu dem der Bahnübergang in den Defektzustand eintritt, und strikt danach darf die Meldung „gesichert“ nicht versandt werden.

Klasse 9.2: ***Gültigkeit / Erreichbarkeit strikt nach bestimmtem Zeitpunkt***

Diese Art von Sicherheitsanforderungen beschreiben eine Aussage q, die strikt nach einer ersten Aussage p gültig ist. *Strikt nach* bedeutet, dass q nicht zusammen mit p jedoch nach p gültig ist. Das Wort *strikt* wird zur Unterscheidung von *exakt nach* (Klasse 9.3) verwendet. Im Falle von *strikt nach* liegt die Betonung darauf, dass es nicht um ein Gelten der Aussage q zum Zeitpunkt des Eintritts der Bedingung p geht, sondern um die Gültigkeit von q im Folgenden (vgl. Abbildung B.5).

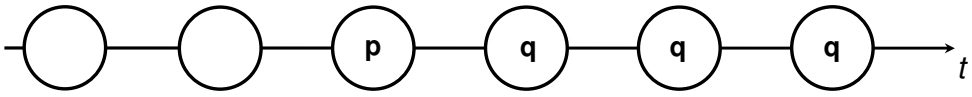


Abbildung B.5: Gültigkeit von q ab einem Zeitpunkt strikt nach p

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: b muss strikt nach dem Zeitpunkt, zu dem a gültig ist, mindestens n-mal gültig sein. Ist b im Zeitpunkt, zu dem a gilt, erfüllt, so wird dies nicht zu den *n* Zuständen hinzugerechnet, zu denen b mindestens gültig sein muss. *n* wird erst ab dem Zeitpunkt unmittelbar nach a gezählt.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb: *Strikt nach dem Zeitpunkt, zu dem die FFB-Zentrale die Meldung erhält, dass der Bahnübergang defekt ist, darf solange keine Fahrerlaubnis erteilt werden, bis der Bahnübergang repariert worden ist.* Ab dem Zeitpunkt, unmittelbar nachdem die beschriebene Meldung bei der FFB-Zentrale eingetroffen ist, darf keine Fahrerlaubnis erteilt werden.

Klasse 9.3: ***Gültigkeit / Erreichbarkeit exakt nach bestimmtem Zeitpunkt***

Diese Art von Sicherheitsanforderungen besitzen eine Aussage q, die direkt nach einer ersten Aussage p gültig ist. Das bedeutet einerseits, dass q nicht zusammen mit p gültig ist, und andererseits, dass q unmittelbar nach p gültig ist (vgl. Abbildung B.6). Für zeitgetriggerte Systeme ist q im nächsten Zeittakt nach p gültig. Im Falle eines ereignis-getriggerten Systems bedeutet dies, dass es zwischen p und q nicht erlaubt ist, dass irgendeine andere definierte Systemvariable ihren Wert ändert. Die Worte *exakt* oder *direkt* werden zur Unterscheidung von *strikt nach* verwendet (vgl. Klasse 9.2).

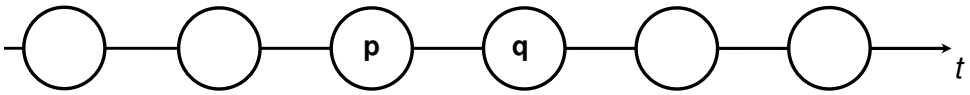


Abbildung B.6: Gültigkeit von q im genau nächsten Zustand nach p

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: b muss direkt nach dem Zeitpunkt, zu dem a erfüllt ist, gültig sein. Wenn b in einem Zustand gilt, so darf strikt nach diesem Zustand sich keine andere Systemvariable ändern, bevor a gültig ist.

Konkretes Beispiel, Sicherheitsanforderung an ein Bremssteuermodul: *Die Schnellentlüftung muss abgebrochen werden, unmittelbar nachdem der Fahrer einen Bremswunsch gestellt hat.* Die Schnellentlüftung muss unmittelbar nach einem Bremswunsch des Fahrers abgebrochen werden.

Klasse 9.4: ***Systeminitialisierungseigenschaft / Eigenschaften mit nicht-temporalen Bedingungen***

In dieser Klasse geht es einerseits um Sicherheitsanforderungen, bei denen eine Eigenschaft im Systembeginn-Zustand erfüllt sein muss. Zudem geht es andererseits um nicht-zeitliche Eigenschaften, die sich auf einzelne Zustände beziehen. Diese Eigenschaften sind gültig, wenn im selben Zustand eine bestimmte Bedingung erfüllt ist (vgl. Abbildung B.7).

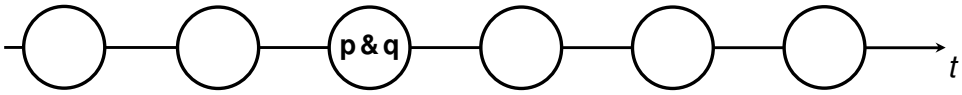


Abbildung B.7: Gültigkeit von q im selben Systemzustand wie p

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: Immer wenn a erfüllt ist, muss b auch gelten. Wenn a gilt, muss immer im selben Zustand auch b gelten. Es wird jedoch keine Aussage darüber getroffen, wann b sonst noch gelten kann.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb: *Wenn die Bedingung Schrankenanlage.Untere_Endlage_erreicht nicht erfüllt ist, darf die Aktion Bahnübergang.status_melden(gesichert) nicht ausgeführt werden.* Wenn die beschriebene

Bedingung unerfüllt ist, darf im selben Zustand niemals die Variable *Bahnübergang.status_melden(gesichert)* gesetzt sein.

Klassifikationskriterium 10: ***Art der Gültigkeits-/Erreichbarkeitsdauer***

Für Sicherheitsanforderungen über die Gültigkeits- bzw. Erreichbarkeitsdauer einer Eigenschaft muss spezifiziert werden, wie lange genau die Geltung bzw. Erreichbarkeit der Eigenschaft dauern soll. Ähnlich wie unterschiedliche Arten des Gültigkeitsbeginns unterschieden werden können, können auch unterschiedliche Arten der Gültigkeitsdauer unterschieden werden.

Klasse 10.1: ***Gültigkeit / Erreichbarkeit einschließlich bis zu einem bestimmten Zeitpunkt***

Die Sicherheitsanforderungen dieser Klasse beschreiben eine Aussage `q`, die solange gültig ist, bis eine Aussage `p` gültig wird. Dabei wird mit dem Wort „bis“ ausgedrückt, dass `q` vor dem Zeitpunkt gültig ist, zu dem `p` gültig ist, und dass im Zeitpunkt, zu dem `p` gültig ist, `q` auch gültig ist (vgl. Abbildung B.8).

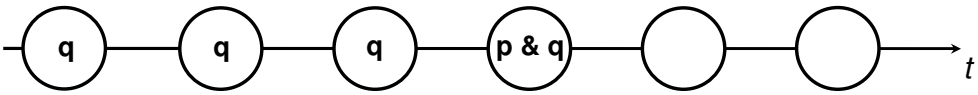


Abbildung B.8: Gültigkeit von `q` bis einschließlich zu dem Zeitpunkt, zu dem `p` eintritt

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss bis zu dem Zeitpunkt, zu dem `a` gültig ist, mindestens `n`-mal gültig sein. `b` ist einschließlich bis zu dem Zeitpunkt mindestens `n`-mal erfüllt, zu dem die Aussage `a` wahr ist.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb: *Strikt nach dem Zeitpunkt, zu dem die FFB-Zentrale die Meldung erhält, dass der Bahnübergang defekt ist, darf solange keine Fahrerlaubnis erteilt werden, bis der Bahnübergang repariert worden ist.* Erst strikt nachdem der Bahnübergang repariert worden ist, darf wieder eine Fahrerlaubnis erteilt werden.

Klasse 10.2: ***Gültigkeit / Erreichbarkeit bevor ein bestimmter Zeitpunkt eintritt***

Die Sicherheitsanforderungen dieser Klasse beschreiben eine Aussage `q`, die gültig ist, bevor eine Aussage `p` gültig wird. Dabei wird mit dem Wort „bevor“ ausgedrückt, dass `q` so lange gültig ist, solange `p` noch nicht gültig ist. Über die Gültigkeit von `q` im Zeitpunkt, zu dem `p` gültig ist, wird dabei keine Aussage getroffen.

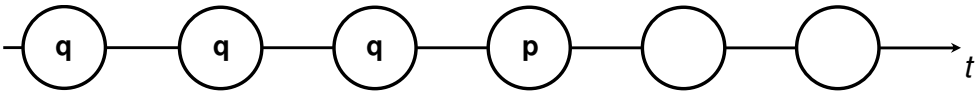


Abbildung B.9: Gültigkeit von `q` vor dem Zeitpunkt, zu dem `p` eintritt

Beispiel einer generischen Sicherheitsanforderungs-Formulierung: `b` muss vor dem Zeitpunkt, zu dem `a` gültig ist, mindestens `n`-mal gültig sein. `b` ist einschließlich bis zu dem Zeitpunkt direkt vor dem Eintritt der Aussage `a` mindestens `n`-mal gültig. Ist `b` im Zeitpunkt, zu dem `a` gilt, erfüllt, so wird dies nicht zu den *n* Zuständen hinzugerechnet, zu denen `b` mindestens gültig sein muss.

Konkretes Beispiel, Sicherheitsanforderung an einen Bahnübergang im Funk-Fahr-Betrieb: *Strikt nach dem Zeitpunkt, zu dem die FFB-Zentrale die Meldung erhält, dass der Bahnübergang defekt ist, und vor dem Zeitpunkt der Bestätigung der Reparierung des Bahnübergangs, darf keine Fahrerlaubnis erteilt werden.* Ab dem Zeitpunkt, zu dem die Reparierung bestätigt wird, darf eine Fahrerlaubnis erteilt werden.

C Gegenbeispiel für die automatische Generierung von Formulierungen in der Safety-Pattern-Normsprache

In Abschnitt 6.2 wurde erläutert, dass die generischen Formulierungen der Safety-Pattern in der Safety-Pattern-Normsprache manuell spezifiziert wurden. Dadurch werden künstliche Formulierungen vermieden und somit die Natürlichkeit der Formulierungen unterstützt. Dies wäre nicht möglich, wenn diese Formulierungen aus den notwendigen Formulierungen automatisch generiert würden. Dann könnte sich leicht ein komplizierter Satzbau mit nicht immer passenden Begriffen ergeben (wenig flexible Begriffswahl). Hierfür wird im Folgenden ein Beispiel gegeben.

Beispiel: Für die folgende generische Formulierung (SP12) soll die Spezifikation in der Safety-Pattern-Normsprache von notwendigen (grundlegenden) Formulierungen generiert werden.

SP12:
Generische Spezifikation in formaler Sprache:
`G (a → G (b → incr)) and G (a → X F (i=n))`

Dabei ist `incr` eine Funktion zur Inkrementierung von `i`. Dieses Safety-Pattern kann aus SP5 und SP11 abgeleitet werden:

SP5:
Generische Spezifikation in formaler Sprache: `G (a → G b)`

Generische Spezifikation in Normsprache: Von dem Zeitpunkt an, zu dem `a` gültig ist, muss `b` permanent gelten.

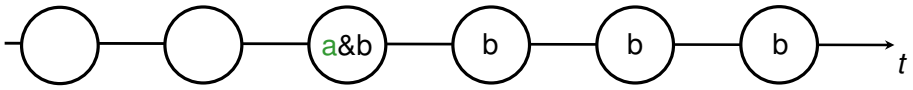


Abbildung C.1: Veranschaulichung der Bedeutung von SP 5

SP11:
Generische Spezifikation in formaler Sprache: `G (a → X F b)`

Generische Spezifikation in Normsprache: Strikt nach dem Zeitpunkt, zu dem `a` gültig ist, muss `b` irgendwann mindestens einmal gültig sein.

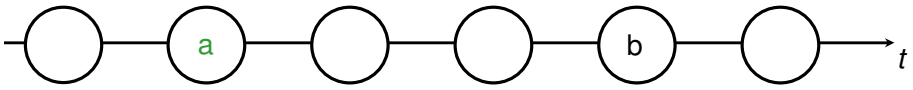


Abbildung C.2: Veranschaulichung der Bedeutung von SP 11

Ableitungsprinzip für die Ableitung von SP12 aus SP5 und SP11:

`SP12:= (SP5`, mit `b:= (b → incr)) and (SP11`, mit `b:= (i = n))`

Dabei bedeutet „`b → incr`“: Wenn `b` gilt, dann muss gleichzeitig auch `incr` gelten.

Generiertes Ableitungsergebnis der Formulierung in Normsprache:

Von dem Zeitpunkt an, zu dem `a` gültig ist, muss (wenn `b` gilt, dann muss auch `incr` gelten) permanent gelten.

`and`

Strikt nach dem Zeitpunkt, zu dem `a` gültig ist, muss `(i = n)` irgendwann mindestens einmal gültig sein.

Manuelle generische Spezifikation in Normsprache:

Strikt nach dem Zeitpunkt, zu dem `a` gültig ist, muss `b` mindestens `n`-mal gelten.

Vergleicht man das generierte Ergebnis mit dem manuell spezifizierten Ableitungsergebnis, so wird ersichtlich, dass das Zweite wesentlich schneller und einfacher verständlich ist.

D Grafische Notation zur Spezifikation mit Safety-Pattern

In Abschnitt 6.3 wurde GNSS als geeignete grafische Notation für die Veranschaulichung der generischen Formulierungen der Safety-Pattern hergeleitet. Im Folgenden wird eine Einführung in GNSS gegeben. In Tabelle D.1 sind sämtliche grafische Spezifikationselemente von GNSS aufgelistet. Die geringe Anzahl an erforderlichen Elementen dient der einfachen Erlernbarkeit. Zudem können die Symbole, die aus der Aussagenlogik bekannt sind (vgl. Abschnitt 2.4.1.1) in ihrer ursprünglichen Bedeutung verwendet werden.

Tabelle D.1: Grafische Notation von GNSS

Spezifikationselement	Bedeutung
○	Zustandssymbol: beliebiger Zustand
◇ q	Zustandssymbol: möglicher Zustand, in dem q gilt
△ q	Zustandssymbol: erforderlicher Zustand, in dem q gilt
[ ]	Bedingungskennzeichnung für einzelne Variablen, aussagenlogische Ausdrücke, Zustände oder ganze Relationen
⟶	Transitionssymbol: genau ein einziger Zustandsübergang
- - - - - - - - ➔	Transitionssymbol: beliebig viele Zustandsübergänge
●	Anfangszustand[23]
◉	Endzustand

Zur Bildung von GNSS-Ausdrücken gelten die folgenden Spezifikations-Regeln. Diese müssen nicht selbstständig beherrscht, sondern nur verstanden werden, weil das Safety-Pattern-Konzept nicht die selbstständige Spezifikation in GNSS erfordert, sondern GNSS nur als Hilfsmittel bei der Instanziierung und Interpretation anbietet.

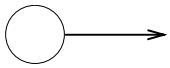
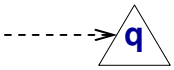
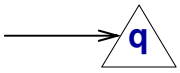
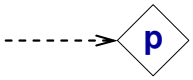
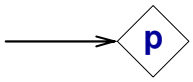
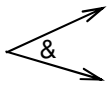
[23] Wie von UML bekannt, ist dies nicht der erste Zustand, sondern ein Pseudo-Zustand. Es ist nicht der erste Zustand, aber das erste Symbol.

Syntax-Regeln:

1. Jeder Ausdruck beginnt mit einem Anfangszustand oder dem Symbol für beliebige Zustände.
2. Das letzte Element muss immer der Endzustand sein.
3. Zwischen Zustandssymbolen muss immer ein Transitionssymbol oder als Ausnahme ein logisches UND stehen.
4. Zwischen Transitionssymbolen muss immer ein Zustandssymbol stehen.
5. Aussagenlogische Symbole werden nur innerhalb von Zuständen verwendet (Ausnahme: logisches UND-Symbol, vgl. Erläuterung der Bedeutung von Symbolkombinationen).

In Tabelle D.2 wird die Bedeutung besondere Symbole in GNSS-Ausdrücken sowie besonderer Symbolkombinationen erläutert.

Tabelle D.2: Erläuterung der Bedeutung besonderer Symbole und Symbolkombinationen

Symbolkombinationen	**Bedeutung**
	„für jeden Zustand gilt“
	„q muss irgendwann erreicht werden“
	„q muss im unmittelbar folgenden Zustand gelten“
	„p muss erreichbar sein“
	„p muss im unmittelbar folgenden Zustand erreichbar sein“
	Mögliche Alternativen werden durch eine Verzweigung ausgedrückt.
	Wenn von einem Zustand ausgehende, unterschiedliche Relationen zusammen erfüllt sein müssen, wird eine solche Verzweigung durch ein „&“ gekennzeichnet.

In Abbildung D.1, Abbildung D.2 und Abbildung D.3 sind drei typische Beispiele aufgeführt.

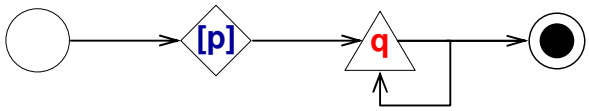


Abbildung D.1: Beispiel mit der permanenten Gültigkeit einer Aussage

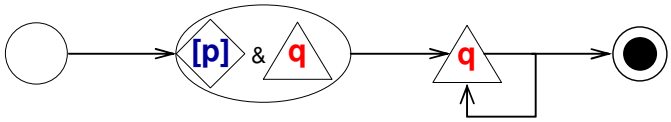


Abbildung D.2: Beispiel für die Gültigkeit einer Aussage ab einem bestimmten Zeitpunkt

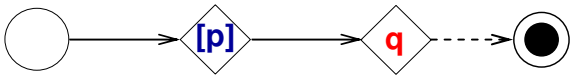


Abbildung D.3: Beispiel für die Erreichbarkeit der Gültigkeit einer Aussage

Äquivalente Spezifikation für den GNSS-Ausdruck aus Abbildung D.1 in der Safety-Pattern-Normsprache: `q` muss strikt nach dem Zeitpunkt, zu dem `p` gültig ist, ununterbrochen gelten.

Äquivalente Spezifikation für den GNSS-Ausdruck aus Abbildung D.1 in CTL: `AG (p → AX AG q)`

Äquivalente Spezifikation für den GNSS-Ausdruck aus Abbildung D.2 in der Safety-Pattern-Normsprache: `q` muss ab dem Zeitpunkt, zu dem `p` gültig ist, ununterbrochen gelten.

Äquivalente Spezifikation für den GNSS-Ausdruck aus Abbildung D.2 in CTL: `AG (p → AG q)`

Äquivalente Spezifikation für den GNSS-Ausdruck aus Abbildung D.3 in der Safety-Pattern-Normsprache: Zum Zeitpunkt, zu dem `p` gültig ist, muss die Gültigkeit von `q` direkt erreichbar sein.

Äquivalente Spezifikation für den GNSS-Ausdruck aus Abbildung D.3 in CTL: `AG (p → EX q)`

Bei der Entwicklung des Safety-Pattern-Katalogs wurden die generischen Spezifikationen in GNSS manuell erstellt. Eine automatische Ableitung der Spezifikationen aus den formalen generischen Spezifikationen ist möglich. Bei einer manuellen Spezifizierung kann jedoch für jedes Safety-Pattern individuell die beste Darstellung gewählt werden. Auch in GNSS können verschiedene Ausdrücke spezifiziert werden, die auf der Bedeutungsebene dieselbe Bedeutung haben. Da es bei den GNSS-Spezifikationen um die klare Verständlichkeit geht, ist eine optimale Darstellung erforderlich. Diese kann am besten manuell ausgewählt werden.

Die GNSS ist so konzipiert, dass auch Hierarchien gebildet werden können, d. h., GNSS-Spezifikationen können in andere eingesetzt werden. Dies ist bei der Anwendung des Safety-Pattern-Konzepts zur Bildung komplexer Sicherheitsanforderungen erforderlich. Bei der

Instanziierung können Ausdrücke eingesetzt werden, die durch Safety-Pattern-Selektion ermittelt wurden. Die Kontrolle, ob der Ausdruck der gewünschten Formulierung entspricht, kann z. B. durch die Einsetzung der jeweiligen entsprechenden GNSS-Ausdrücke ineinander erfolgen, wodurch die Gesamtbedeutung veranschaulicht wird.

E Die Symbole des Simulations-Konzepts und zusätzliche Informationen zur Konzept-Umsetzung

E.1 Übersicht über die Symbole des Simulations-Konzepts

In Tabelle E.1 werden die grafischen Symbole des Simulations-Konzepts aufgelistet und beschrieben.

Tabelle E.1: Verwendete Symbole in der Zeitdiagramm-Simulation des Safety-Pattern-Konzepts

Symbol	Bedeutung
	Der Wert muss in diesem Zustand „0“ sein.
	Der Wert muss in diesem Zustand „1“ sein.
	Der Wert ist in diesem Zustand undefiniert, d. h. beliebig.
	Der Wert ist in diesem Zustand zum Beispiel „0“.
	Der Wert ist in diesem Zustand zum Beispiel „1“.
	Von diesem Zustand aus muss eine Erreichbarkeit gewährleistet sein.
	Zum Beispiel in diesem Zustand wird der Wert „1“ erreicht.

Symbol	Bedeutung
	In diesem Zustand wird zum Beispiel die permanente Geltung des Wertes „1“ erreicht.
	In diesem Zustand wird der Wert „1“ zum Beispiel erreicht, und von diesem Zustand aus muss eine Erreichbarkeit gewährleistet sein.
	In diesem Zustand wird der Wert „1“ zum Beispiel nicht erreicht.

E.2 Umsetzung des Simulations-Konzepts zur Unterstützung der Interpretation von Safety-Pattern

Damit die Simulationen einfach aus der Webapplikation SAPIS aufgerufen werden können, wurden sie als Java-Applets realisiert. Mit Java wurden die Applikationslogik und die Logik für die einzelnen Safety-Pattern implementiert. Java-Swing wurde für die dynamischen grafischen Darstellungen eingesetzt. Abbildung E.1 zeigt ein Beispiel für die Umsetzung.

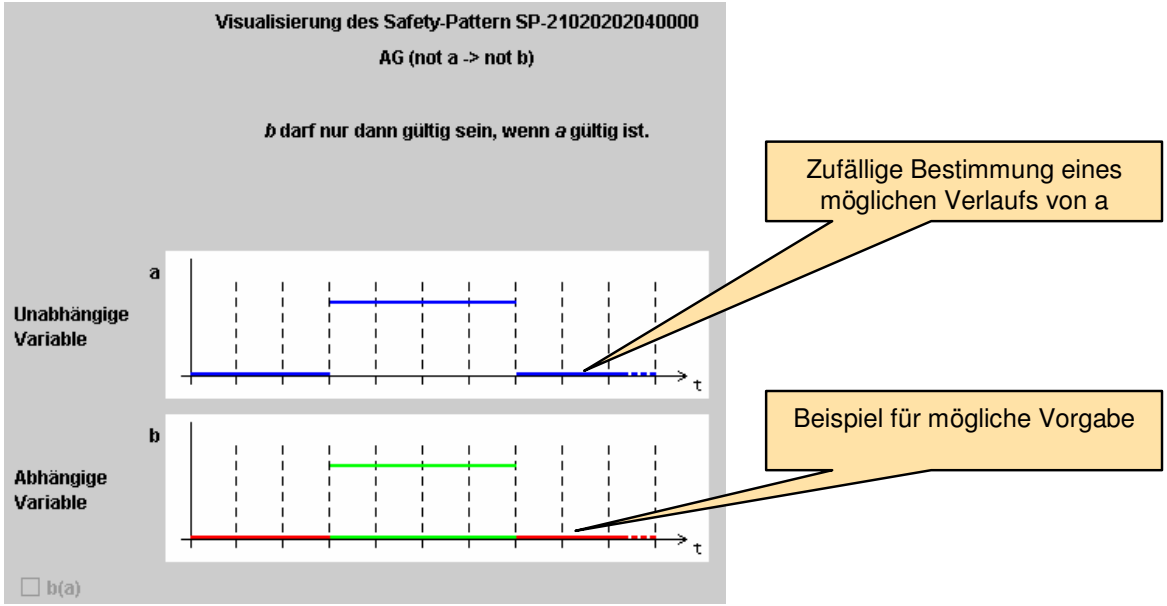


Abbildung E.1: Beispiel für Rückwärtssimulation

Bei der Umsetzung des Simulations-Konzepts (vgl. Abschnitte 6.4.3 und 7.3.2) mussten zwei Möglichkeiten der Simulation berücksichtigt werden. Zum einen kann der Benutzer einen möglichen

Werteverlauf für die unabhängige bzw. die unabhängigen Safety-Pattern-Variablen vorgeben, wofür werkzeuggestützt der entsprechende Werteverlauf der abhängigen Variable automatisch berechnet wird. In diesem Fall gibt der Benutzer einen möglichen Werteverlauf vor (blaue Linien mit dem Wert „wahr“ oder „falsch“, vgl. Abbildung 6.21 und Übersicht in Anhang E.1). Zum anderen kann der Benutzer auch einen möglichen Verlauf für die abhängige Safety-Pattern-Variable vorgeben und sich dann einen möglichen dafür erforderlichen Verlauf der unabhängigen Variable anzeigen lassen (vgl. Abbildung E.1 und Übersicht in Anhang E.1). Für diese Vorgabe stehen ihm alle Symbole des Simulationskonzepts zur Verfügung. Abbildung E.1 ist hierfür ein Beispiel. Dabei wurde für den Werteverlauf von `a` ein möglicher Verlauf für `b` berechnet und visualisiert.

In [Teuf04] wurde für die Rückwärtssimulation eine Lösung realisiert, welche die in [Jime03] implementierte Safety-Pattern-Logik-Relationen (vgl. Abschnitt 7.3.2) ohne Änderung verwendet. Während die erste Art der Visualisierung schrittweise geschieht, ist dies bei der zweiten Art nicht möglich, da die erforderlichen Werte der unabhängigen Variable immer den gegenwärtigen oder einen Zeitpunkt in der Vergangenheit betreffen. Die abhängige Variable steht jedoch immer in einer solchen temporallogischen Relation zu der unabhängigen bzw. den unabhängigen Variablen, welche die Gegenwart oder die Zukunft betrifft (vgl. Abschnitte 2.4.1.3 und 6.1.2).

Für die Rückwärtsvisualisierung ist zusammenfassend Folgendes zu beachten:

- Der Benutzer kann eine von verschiedenen möglichen Verläufen für die abhängige Variable vorgeben (vgl. Abbildung E.1).
- Der Werteverlauf der abhängigen Variable kann nicht beliebig gewählt werden, da je nach Safety-Pattern nur bestimmte Variablenwertefolgen möglich sind (vgl. Beispiel für eine unmögliche Rückwärtsvisualisierung in Abbildung E.2).

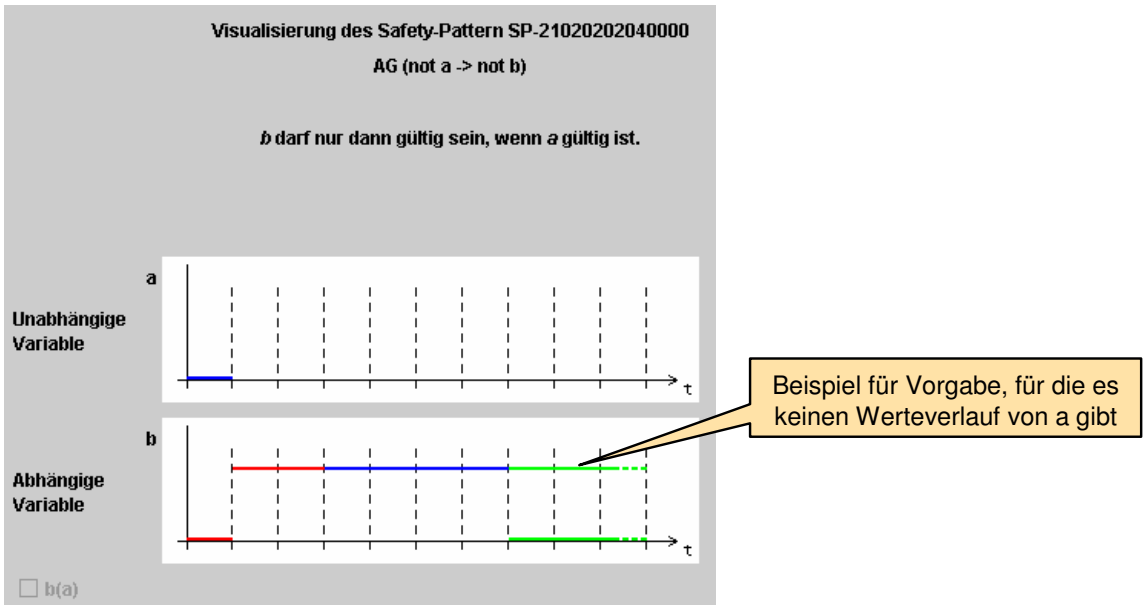


Abbildung E.2: Beispiel mit Werteverlaufvorgabe ohne Rückwärtssimulierbarkeit

E.3 Umsetzung des Simulations-Konzepts für Safety-Pattern-Kompositionen

In Abschnitt 6.4.3 wurde bereits diskutiert, dass für die Überprüfung von UND-Kompositionen die Unterstützung durch Simulationen hilfreich ist. Interessant sind dabei die Fälle, in denen eine Variable von mehreren anderen (Safety-Pattern 1 mit $c(a)$ und Safety-Pattern 2 mit $c(b)$, vgl. Beispiel in Abbildung E.3) oder in verschiedener Weise von einer anderen Variablen abhängt (Safety-Pattern 1 mit $c(a)$ und Safety-Pattern 2 mit $c(a)$).

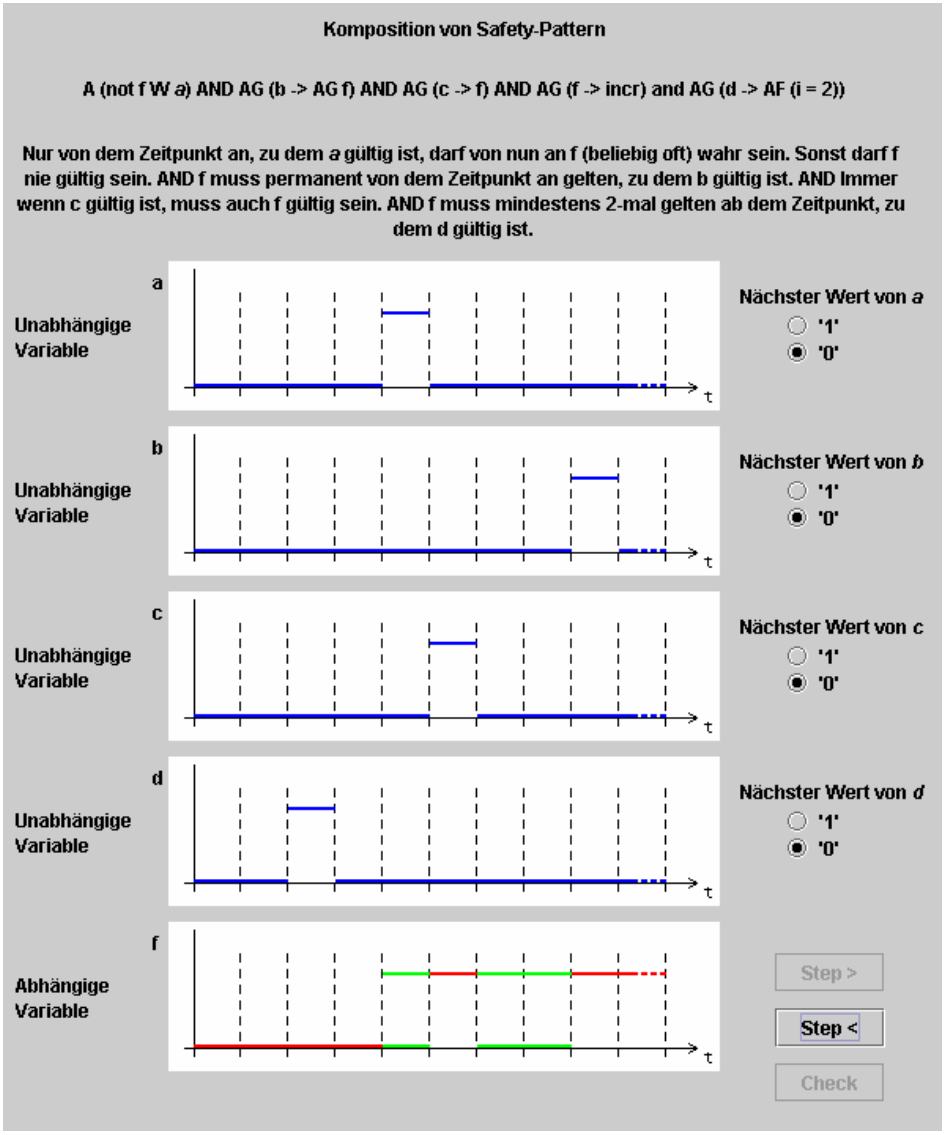


Abbildung E.3: Beispiel für die Simulation der Komposition mehrerer Safety-Pattern

Bei der Simulation einer solchen Komposition werden schrittweise die Werte der abhängigen Variable angezeigt. Diese werden durch die Komposition der für die einzelnen Safety-Pattern berechneten Simulations-Werte in Abhängigkeit zu der vom Benutzer vorgegebenen unabhängigen Variablen bestimmt. Ein Wert, welcher notwendigerweise wahr bzw. falsch gesetzt wird, ist dabei dominant. Muss die abhängige Variable aufgrund unterschiedlicher Safety-Pattern in einem Zustand

sowohl wahr als auch falsch sein, ist ein Widerspruch vorhanden, der somit aufgedeckt wird. Der Benutzer erhält eine entsprechende Meldung. Wenn es z. B. bei der Berechnung der Werte bei einem Safety-Pattern *A* einen Zustand gibt, bei dem die Variable den Wert 1 annehmen muss, dann darf in diesem Zustand die Berechnung für ein konjungiertes Safety-Pattern *B* weder den Wert 0 ergeben noch, dass die Variable in diesem Zustand nicht erreichbar ist (vgl. Beispiel in Abbildung E.4).

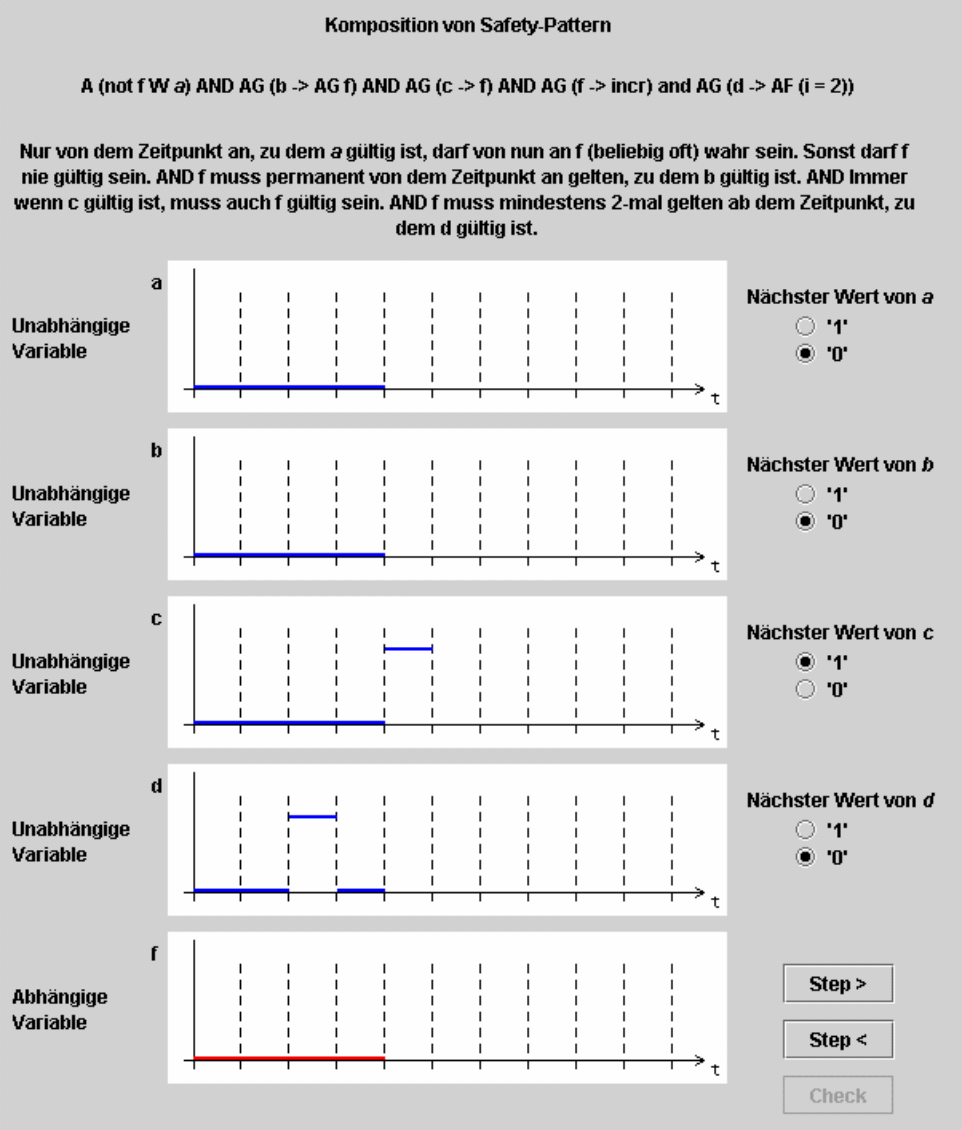


Abbildung E.4: Kompositions-Beispiel mit widersprüchlichem Werteverlauf

In [Jime03] und [Teuf04] wurde die Kompositions- und Widerspruchsaufdeckungsfunktionalität für bis zu fünf Safety-Pattern prototypisch realisiert. Die Zahl der unabhängigen Variablen ist auf fünf begrenzt, da die Darstellung der Komposition von bis zu fünf Variablen auf einem gewöhnlichen 17-Zoll Bildschirm praktikabel ist.

Es gibt Kompositionen, die für keinen Werteverlauf widerspruchslos sind. Durch eine automatische Prüfung, bei der alle möglichen Variablen-Kombinationen durchgespielt werden (Aufruf über den Button „Check“, vgl. Abbildung E.4), kann dies erkannt und dem Benutzer gemeldet werden. Da

sich diese Widerspruchsprüfung auf einen Zeitausschnitt beschränkt, kann eine Widerspruchsfreiheit nicht allgemein gültig nachgewiesen werden. Sie dient jedoch der Aufdeckung von Widersprüchen. In den Formulierungen der Safety-Pattern geht es maximal um 3 Zeitintervalle (das Intervall, in dem eine Eigenschaft gewährleistet sein soll und jeweils das Intervall davor und danach). Damit ist der betrachtete Zeitausschnitt mit 10 Zuständen praktikabel, um die Widersprüche von maximal fünf konjugierten Safety-Pattern aufdecken zu können. Wenn bei einem der Safety-Pattern die abhängige Variable mit einer gewissen Häufigkeit gelten oder erreichbar sein soll, muss überprüft werden, ob diese Häufigkeitsanzahl für den verwendeten Werteverlauf in der betrachteten Zeit erreicht wurde. Ist dies nicht der Fall, muss der Benutzer durch eine Meldung darauf hingewiesen werden, dass Widersprüche vorliegen können.

F Beispiel eines Safety-Pattern

In Abbildung F.1 und Abbildung F.2 wird beispielhaft ein Safety-Pattern dargestellt. Überschrieben ist das Safety-Pattern mit dem Namen „DS-miZ-nG-84“, der sich aus der Safety-Pattern-Klassifikation und einer Safety-Pattern-ID ableitet. Dabei werden nicht alle Klassifikationskriterien durch Abkürzungen aufgeführt, sondern nur Wesentliche, sodass eine grobe Einordnung des Safety-Pattern möglich ist. Eindeutig ist die Safety-Pattern-ID, mit der alle Safety-Pattern durchnummeriert werden. Im Beispiel steht „DS“ für „Dynamische Sicherheitsanforderung“, „miZ“ für „Sicherheitsanforderung mit ausschließlich impliziten Zeitangaben“ und schließlich „nG“ für „notwendige Geltung“. Hinter dem Namen folgt der Safety-Pattern-ID-Code „SA-2-1-0-2-3-1-1-0-1-1-0-0-0“, der sich aus der Klassifikation ergibt.

Safety-Pattern DS-miZ-nG-84 (SA-2-1-0-2-3-1-1-0-1-1-0-0-0)

Klassenzuordnung

1.2 Dynamische Sicherheitsanforderung
2.1 Sicherheitsanforderung mit ausschließlich impliziten Zeitangaben
4.2 Sicherheitsanforderungen über zeitliche Aufeinanderfolge
5.3 Gültigkeitsbeginn und –dauer
6.1 Ununterbrochene Geltung/Erreichbarkeit im betrachteten Zeitintervall
7.1 Notwendige Geltung
9.1 Gültigkeit/Erreichbarkeit ab bestimmtem Zeitpunkt
10.1 Gültigkeit/Erreichbarkeit einschließlich bis zu einem bestimmten Zeitpunkt

Generische Formulierung in Normsprache

Wenn *a* gültig ist, muss von nun an *b* solange ununterbrochen gelten, bis *c* gültig ist.

Generische Formulierung in formalen Sprachen

CTL: `AG (a → A(b W (c and b)))`

LTL: `G (a → (b W (c and b)))`

µ-Kalkül: `nu Z1.((a → nu Z0.((c and b) or (b and [-]Z0))) and [-]Z1)`

Generische Formulierung in GNSS

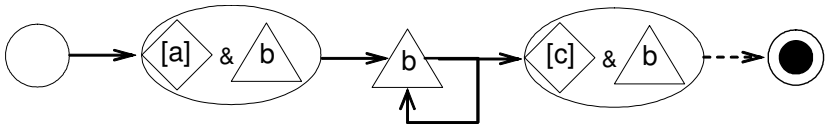


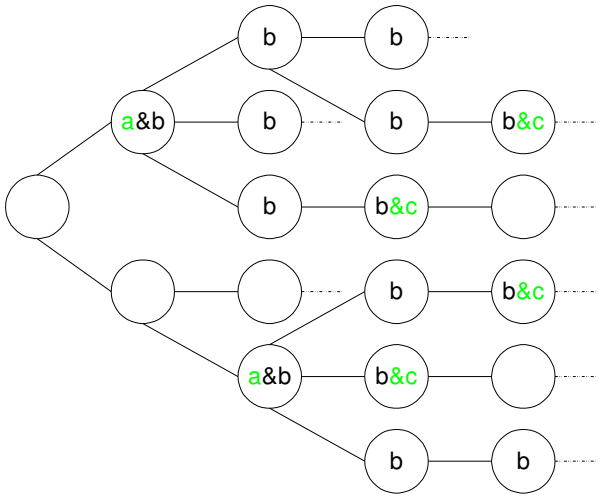
Abbildung F.1: Safety-Pattern-Beispiel – Teil 1

Erläuterung in natürlicher Sprache

Ein wesentliches Merkmal dieses Safety-Pattern ist zum einen, dass *b* immer von jedem Systemzustand an gültig sein muss, zu dem *a* wahr ist. Zum anderen muss *b* dann solange erfüllt sein, bis zu dem Systemzustand, in dem *c* eintritt. Dies gilt ab jedem Systemzustand, in dem a erfüllt ist.

Das betrachtete Zeitintervall beginnt einschließlich mit dem Zustand, zu dem *a* gilt, und endet zusammen mit dem Zustand, zu dem *c* gilt. In dem betrachteten Zeitintervall muss *b* permanent gelten, d. h., die Gültigkeit von *b* darf nie unterbrochen sein. Sie darf weder irgendwann vor dem Eintritt von *c* enden noch im Zustand direkt zuvor. *a* sowie *c* können irgendwann eintreten. Es sind keine bestimmten Zeitpunkte festgelegt, zu denen sie gelten müssten oder nicht gelten dürften. Zudem müssen *a* sowie *c* nicht irgendwann gelten. Über die Geltung von *b* vor dem Zustand, zu dem *a* gilt und strikt nach dem Zustand, zu dem *c* gilt, wird keinerlei Aussage gemacht.

Erläuterung durch grafische Darstellung



Musterbeispiel

Automatisierungssystem
Bahnübergang im Funk-Fahr-Betrieb

Konventionelle Spezifikation der Sicherheitsanforderung
Nachdem `Bahnübergang.Defektmeldung_Lichtzeichenanlage or Bahnübergang.Defektmeldung_Schrankenanlage` eintritt, muss `Bahnübergang.Empfangsbestätigung`[24] `= 0` gelten, bis `Bahnübergang.repariert` eintritt.

Sicherheitsanforderung in Normsprache
Wenn `Bahnübergang.Defektmeldung_Lichtzeichenanlage or Bahnübergang.Defektmeldung_Schrankenanlage` gültig ist, muss von nun an `Bahnübergang.Empfangsbestätigung = 0` solange ununterbrochen gelten, bis `Bahnübergang.repariert` gültig ist.

Sicherheitsanforderung in formaler Sprache (CTL)
`AG ((Bahnübergang.Defektmeldung_Lichtzeichenanlage or Bahnübergang.Defektmeldung_Schrankenanlage) →`
`A((Bahnübergang.Empfangsbestätigung = 0) W (Bahnübergang.repariert and Bahnübergang.Empfangsbestätigung = 0)))`

Abbildung F.2: Safety-Pattern-Beispiel – Teil 2

[24] Erhält im Funkfahrbetrieb ein Bahnübergang vom Eisenbahnfahrzeug per Funk eine Anweisung zur Sicherung, muss er diese im störungsfreien Betrieb durch eine „Empfangsbestätigung“ per Funk quittieren, vgl. Abschnitt 8.2.1.

G Erörterung der Generierung der nützlichen generischen Formulierungen in SAPIS

Wie in Abschnitt 6.1 ausgeführt wurde, sind die nützlichen generischen Formulierungen des Safety-Pattern-Konzepts aus den notwendigen ableitbar. Für die formalen Formulierungen ist diese Ableitung auf jeden Fall sinnvoll. Falls die Safety-Datenbank um eine weitere formale Sprache ergänzt werden soll, müssen nur die notwendigen generischen Formulierungen in dieser formalen Sprache beschrieben werden, während die nützlichen generischen Formulierungen daraus abgeleitet werden. Somit kann der Aufwand für Wartungsarbeiten reduziert werden. Für die anderen Formulierungen und Erläuterungen muss der Nutzen im Folgenden genauer analysiert werden. Dabei gibt es die folgenden Bewertungskriterien:

1. Realisierung und Wartung von SAPIS
2. Benutzung des Safety-Pattern-Konzepts

Das zweite Bewertungskriterium hat ein höheres Gewicht als die Werkzeug-Realisierung und -Wartung, weil die Anwendbarkeit eine wesentliche Konzept-Anforderung für das Verfahren ist. Im Gegensatz zu den formalen Formulierungen kommt es bei den anderen Safety-Pattern-Bestandteilen auf die klare Verständlichkeit und Nachvollziehbarkeit an.

Ein Beispiel für ein abzuleitendes Safety-Pattern ist das mit der Safety-Pattern-ID 63:

Generische Spezifikation in formaler Sprache: `AG (a → AG (not b))`

Die Ableitung kann aus folgenden Safety-Pattern erfolgen:

Safety-Pattern-ID 13
Generische Spezifikation in formaler Sprache: `AG (a → c)`

Erläuterung in natürlicher Sprache: `c` muss immer dann wahr sein, wenn gleichzeitig auch `a` gültig ist. Dies gilt nicht nur einmal, sondern für jeden Systemzustand, in dem `a` erfüllt ist. Ist `a` nicht erfüllt, darf `c` dennoch gelten. Es wird jedoch keine Aussage darüber getroffen, wann `c` außerdem gelten soll und darf. Ist `c` erfüllt, kann daraus nicht geschlossen werden, dass auch `a` gelten muss. Ist `a` jedoch erfüllt, muss daraus geschlossen werden, dass auch `c` erfüllt sein muss. Zu welchen Zeitpunkten `a` erfüllt ist, ist unbestimmt.

Safety-Pattern-ID 1
Generische Spezifikation in formaler Sprache: `AG d`

Erläuterung in natürlicher Sprache: Das Charakteristische dieser Art von Sicherheitsanforderung ist, dass eine Eigenschaft `d` ab dem aktuellen Zustand in allen folgenden Zuständen gilt. Diese

Eigenschaft gilt bedingungslos, d. h., die Eigenschaft `d` ist in jedem Zustand auf jeden Fall erfüllt. Eine Bedingung für die Gültigkeit der Eigenschaft gibt es nicht.

Ableitung des Safety-Pattern mit der ID 63 aus denen mit der ID 1 und der ID 13:
Ableitungsprinzip für Ableitung der generischen Spezifikation in formaler Sprache:

- Das Safety-Pattern mit ID 63 kann durch das Safety-Pattern mit ID 13 durch Einsetzen des Safety-Pattern mit ID 1 in `c` und Einsetzen von `(not b)` in `d` gebildet werden.

Bewertung der Ableitung der Erläuterung in natürlicher Sprache:
Die abgeleitete Erläuterung hätte eine sehr komplizierte und unnatürliche Struktur. Die Erläuterungen der Safety-Pattern sind jedoch wichtig, um durch Detailinformationen die klare Verständlichkeit zu unterstützen. Dies ist zum einen für das Spezifizieren mit Safety-Pattern bedeutsam (einfache Handhabbarkeit). Darüber hinaus soll zum anderen die Bedeutung spezifizierter Sicherheitsanforderungen über die Erläuterung der verwendeten Safety-Pattern nachgeschlagen und interpretiert werden (einfache Nachvollziehbarkeit). Deshalb ist wiederum die einfache Verständlichkeit der Erläuterungen wesentlich. Somit macht ein Ableiten der Erläuterungen keinen Sinn. Es würde nur zur Unlesbarkeit der Erläuterungen und zu äußerst schwieriger Verständlichkeit führen, womit die eigentliche Intention der Erläuterung verfehlt werden würde.

Eine Ableitung der ausführlichen Erläuterungen mit seinen komplizierten Textstrukturen ist somit nicht möglich. Dies gilt ebenso für die anderen Erläuterungsarten. In ähnlicher Weise ist auch die Ableitung der generischen Formulierungen in GNSS und in Normsprache nicht sinnvoll. Hier wäre zwar eine Ableitung möglich, die jedoch weniger verständlich als eine kreativ gewählte Formulierung ist (vgl. Beispiel in Anhang C und Abschnitt 5.3.4.2). Die formale Formulierung muss der Benutzer nicht verstehen. In den anderen Fällen reicht es nicht aus, dass eine korrekte Formulierung verwendet wird. Vielmehr muss die Formulierung einfach strukturiert und klar verständlich sein.

Wenn bei der Spezifikation komplexer Sicherheitsanforderungen bereits abgeleitete nicht natürlich formulierte generische Spezifikationen in Normsprache zur Ableitung verwendet werden müssten, so würde dies die Verständlichkeit der Formulierung in Normsprache erheblich beeinträchtigen. In der Menge der 454 Safety-Pattern sind die schwierigen Kompositionsfälle enthalten, die sich in komplexeren Spezifikationsproblemen wiederholen. Deshalb sind die klaren und ausführlichen Erläuterungen der Safety-Pattern nicht nur erforderlich, sondern auch hinreichend für die Spezifikation komplexerer Sicherheitsanforderungen.

Zusammenfassende Bewertung für die Generierung der nützlichen generischen Formulierungen:

1. ***Realisierung und Wartung des Safety-Pattern-Unterstützungssystems:*** Für eine Generierung der nützlichen Safety-Pattern ist die Applikationslogik komplizierter. Formulierungen müssen

nicht nur einfach aus der Datenbank ausgelesen und für die Darstellung aufbereitet werden, sondern aus anderen Formulierungen abgeleitet werden. Dadurch sind die Realisierung sowie Wartungsarbeiten aufwendiger. Allerdings sind keine bestimmten Änderungen der Applikationslogik absehbar. Hingegen kann jedoch die Erweiterung des Werkzeugs um weitere generische Spezifikationssprachen sinnvoll sein. In dem Fall ist die Wartung bei einer Generierungsfunktionalität wesentlich weniger aufwendig, da lediglich die notwendigen generischen Formulierungen explizit formuliert werden müssen.

2. ***Benutzung des Safety-Pattern-Konzepts:*** Wie bereits diskutiert, ist die Qualität und Natürlichkeit der generischen Formulierungen in Normsprache und in GNSS für die Generierungsfunktionalität geringer oder sogar wesentlich geringer. Sinnvolle Erläuterungen sind durch Generierung nicht erzeugbar. Durch die Generierung der formalen Formulierungen wird die Benutzbarkeit jedoch nicht eingeschränkt mit einer Ausnahme: Für die Personen, welche die temporallogischen Formulierungen verstehen möchten (vgl. Abschnitt 5.3.1), kann die Interpretation aufwendiger sein.

Aus diesen Gründen ist nur für die generische Formulierung in formalen Sprachen die Ableitung sinnvoll und dies auch nur dann, wenn das Safety-Pattern-Konzept hauptsächlich für Personen eingesetzt werden soll, die nicht den Formalismus der generischen Formulierungen verstehen möchten.

Für den Fall der Erweiterung der Safety-Pattern-Datenbank um eine weitere formale Sprache wurde deshalb ein Konzept [Touc04] entwickelt, bei dem zur Spezifikation der generischen Formulierungen lediglich die notwendigen generischen Formulierungen in formaler Sprache spezifiziert werden müssen. Auf dieser Basis werden dann die nützlichen generischen Formulierungen generiert. Für diese Generierung wurde eine einfache Ableitungssprache entwickelt [Touc04]. In dieser Ableitungssprache wird spezifiziert, aus welchen notwendigen generischen Formulierungen die Ableitung erfolgen soll und in welcher logischen Relation dabei diese Basis-Formulierungen zueinander stehen.

Die verschiedenen notwendigen generischen Formulierungen werden mit N<x> bezeichnet. <x> steht für die Nummer der generischen Formulierung in der Datenbank. Beispielsweise gibt es die notwendigen generischen Formulierungen N1 und N12:

```
AG p (N1)
AG (p → q) (N12)
```

Die Ableitungssprache enthält alle Zeichen der Aussagenlogik sowie die zusätzlichen Zeichen „N<x>“, „[“, „]“ und „;“. Tabelle G.1 erläutert die Bedeutung dieser Zeichen.

Tabelle G.1: Bedeutung der besonderen Zeichen der Ableitungssprache

Zeichen	**Bedeutung**
N<x>	Notwendige generische Formulierung mit der Nummer x.
[...]	„von“. Beispiel: N1[q = b] bedeutet für q *von* N1 wird b eingesetzt.
;	Trennungszeichen. Mehrere Zuweisungen zu einem N<x> werden durch ein ; abgetrennt. Beispiel: N1[p = a; q = b] bedeutet: für p von N1 wird a und für q von N1 wird b eingesetzt.

Zusätzlich zur Aussagenlogik gelten folgende Syntaxregeln:

- [darf nur in direkter Folge auf ein N<x> verwendet werden.
- Jedes [muss irgendwann durch ein] geschlossen werden.
- In einer Klammer [] muss immer mindestens eine Zuweisung erfolgen.
- Ein ; steht immer zwischen zwei Zuweisungen.
- Eine Zuweisung enthält einen aussagenlogischen Ausdruck mit einzelnen Variablen (z. B. p = a) oder mit notwendigen generischen Formulierungen (z. B. p = N1[p = a]).

Beispielsweise kann die nützliche generische Formulierung AG (b → AG (not a)) mit der Safety-Pattern-ID 63 aus N1 und N12 folgendermaßen abgeleitet werden:

- Einsetzen von N1 in q von N12.
- a für p von N12 einsetzen.
- b für p von N1 einsetzen.

Diese Ableitung wird folgendermaßen ausgedrückt: N12[p=b; q=N1[p=not a]]

H Beispiel der Anzeige eines Safety-Pattern durch SAPIS

Safety Pattern: SS-1 (SA-1-0-0-0-0-0-0-0-0-0-0-0)

Klassifikation

1 - Vorhandensein eines zeitlich-logischen Aspekts in der Sicherheitsanforderung: **1 - statische Sicherheitsanforderung**
2 - Art der Zeitspezifikation: **0 - nicht relevant**
3 - Art der Systemsteuerung: **0 - nicht relevant**
4 - Vorhandensein der Abhängigkeit zwischen Aussagen: **0 - nicht relevant**
5 - Zeitliche Begrenzung der Geltung: **0 - nicht relevant**
6 - Häufigkeit der Geltung im Zeitintervall: **0 - nicht relevant**
7 - Modalität der Forderung: **0 - nicht relevant**
8 - Art der Erreichbarkeit: **0 - nicht relevant**
9 - Art des Gültigkeits-/Erreichbarkeitsbeginns: **0 - nicht relevant**
10 - Art der Gültigkeits-/Erreichbarkeitsdauer: **0 - nicht relevant**
11 - Zulässige Geltung vor/bis Beginn des Geltungs-Zeitintervalls: **0 - nicht relevant**
12 - Zulässige Geltung/Erreichbarkeit nach/ab Ende des Geltungs-/Erreichbarkeits-Zeitintervalls: **0 - nicht relevant**
13 - Art des Erreichbarkeitsverbots: **0 - nicht relevant**

Generische Formulierung in Normsprache

1. a muss immer gültig sein.
2. a muss in allen erreichbaren Systemzuständen wahr sein.
3. a muss permanent erfüllt sein.
4. a ist immer erfüllt.

Spezialfall

1. Bei Systemstart muss a gültig sein.
2. Im ersten Systemzustand muss a erfüllt sein.

Generische Formulierung in formaler Sprache

CTL:
1. AG a

Spezialfall
1. a

LTL:
1. G a

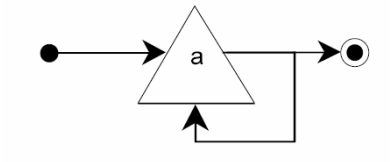
Spezialfall
1. a

µ-Calculus:
1. nu Z0.(a and [-]Z0)

Spezialfall
1. a

Abbildung H.1: Erster Teil der Anzeige eines Safety-Pattern durch SAPIS

Generische Formulierung in GNSS



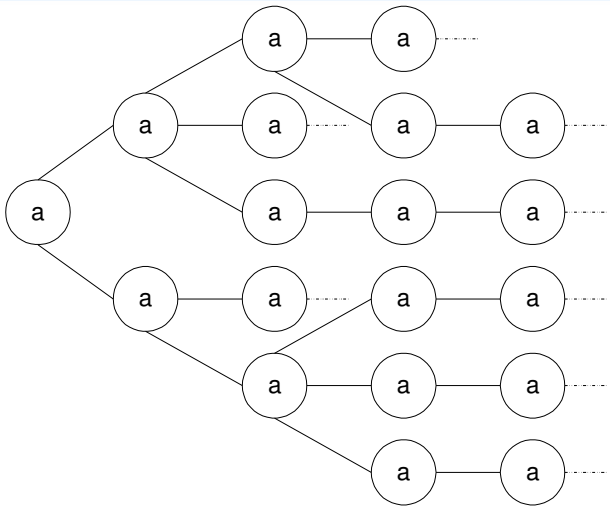
Spezialfälle in GNSS anzeigen

Instanziierung des Safety-Pattern / Speichern

Erläuterung in natürlicher Sprache

Das Charakteristische dieser Art von Sicherheitsanforderung ist, dass eine Eigenschaft a in allen Zuständen des Zustandsraums gilt. Diese Eigenschaft a gilt bedingungslos. Eine Bedingung für die Gültigkeit der Eigenschaft a gibt es nicht. a muss permanent gelten, d. h., die Gültigkeit von a darf nie unterbrochen sein. Dies gilt von Systembeginn an. Es gibt keinen Systemzustand, in dem a nicht gilt.

Grafische Darstellung



Zweite Darstellung | Simulation

Beispiel

Sicherheitsanforderung an das System: Verkehrsampel

Konventionelle Spezifikation: In keiner System-Situation ist es erlaubt, dass die Ampel der Hauptstraße und die Ampel der Nebenstraße grünes Signal geben.

Spezifikation in Normsprache: (not (Ampel_Seitenstraße = gruen and Ampel_Hauptstraße=gruen)) muss immer gültig sein.

Spezifikation in formaler Sprache: AG (not (Ampel_Seitenstraße = gruen and Ampel_Hauptstraße=gruen))

Ähnliche Safety Pattern

SP 1 | SP 2

Startseite

Abbildung H.2: Zweiter Teil der Anzeige eines Safety-Pattern durch SAPIS

I Beispiel für die Selektion eines Safety-Pattern

Für die in Abschnitt 8.4 beschriebene funktionale Sicherheitsanforderung erfolgen bei der Selektion mit SAPIS in der geführten Dialogform nachstehende Selektionsschritte hintereinander:

1. Selektionsschritt:

Kriterium: **Vorhandensein eines zeitlich-logischen Aspekts in der Sicherheitsanforderung**

Klassifikationsfrage: Soll die Sicherheitsanforderung eine zeitlich-logische Aussage enthalten?

Diskussion: Die Eigenschaft `(SchA_ausschalten = 0)` und `(LzA_ausschalten = 0)` darf nur zu bestimmten Zeiten wahr sein. Sie soll nur unter bestimmten Bedingungen gültig sein. Deshalb ist die zu wählende Safety-Pattern-Klasse *Dynamische Sicherheitsanforderungen.*

2. Selektionsschritt:

Kriterium: **Vorhandensein einer Abhängigkeit zwischen Aussagen**

Klassifikationsfrage: Beschreibt die Sicherheitsanforderung eine Abhängigkeit zwischen logischen Ausdrücken oder geht es um eine nicht-zeitlich bedingte Erreichbarkeit oder Zusicherung des Erreichens?

Diskussion: Es gibt eine logische Abhängigkeit zwischen:

1. der Bedingung des Auftretens der `sendeUeberschreitungsmeldung`,
2. der geforderten Eigenschaft `((SchA_ausschalten = 0) and (LzA_ausschalten = 0))` und
3. der Bedingung des Eintretens von `getAusschaltbefehl`.

Aus diesem Grund gehört die Sicherheitsanforderung zur Klasse *Sicherheitsanforderungen mit logischen Abhängigkeiten zwischen Aussagen.*

3. Selektionsschritt:

Kriterium: **Art der Zeitspezifikation**

Klassifikationsfrage: Soll die Sicherheitsanforderung auch explizite Zeitangaben enthalten?

Diskussion: Nein, es soll keine zeitbehaftete Aussage in Abhängigkeit zu einer Systemuhr gemacht werden. Infolgedessen handelt es sich um eine *Sicherheitsanforderung mit ausschließlich impliziten Zeitangaben*.

4. Selektionsschritt:

Kriterium: **Zeitliche Begrenzung der Geltung**

Klassifikationsfrage: Welche zeitliche Begrenzung der Geltung einer Eigenschaft enthält die Sicherheitsanforderung?

Diskussion: Es soll ausgedrückt werden, ab welcher Zeit die geforderte Eigenschaft `((SchA_ausschalten = 0) and (LzA_ausschalten = 0))` gelten soll

und bis wann sie gelten muss. Folglich muss die Sicherheitsanforderung der Safety-Pattern-Klasse *Gültigkeitsbeginn und -dauer* zugewiesen werden.

5. Selektionsschritt:

Kriterium: **Häufigkeit der Geltung/Erreichbarkeit im Zeitintervall**

Klassifikationsfrage: Wie häufig muss bzw. darf die Eigenschaft im Zeitintervall gelten?

Diskussion: Die geforderte Eigenschaft muss ab dem Zeitpunkt, zu dem die Bedingung für den Gültigkeitsbeginn erfüllt ist, bis hin zur Erfüllung der Bedingung für die Gültigkeitsdauer permanent erfüllt sein. Deshalb ist die korrekte Safety-Pattern-Klasse *Ununterbrochene Geltung/Erreichbarkeit*.

6. Selektionsschritt:

Kriterium: **Modalität der Forderung**

Klassifikationsfrage: Welcher Art ist die Forderung?

Diskussion: Die Sicherheitsanforderung soll kein erlaubtes Verhalten, sondern ein notwendiges Verhalten beschreiben. Die geforderte Eigenschaft *muss* zu bestimmten Zeiten gelten. Darum muss die Klasse *Notwendige Geltung* selektiert werden.

7. Selektionsschritt:

Kriterium: **Art des Gültigkeits-/Erreichbarkeitsbeginns**

Klassifikationsfrage: Ab wann genau soll die geforderte Eigenschaft gelten?

Diskussion: Die geforderte Eigenschaft soll ab dem Zeitpunkt gelten, zu dem die Gültigkeitsbeginn-Bedingung (Auftreten der `sendeUeberschreitungsmeldung`) erfüllt ist, und nicht erst ab einem Zeitpunkt danach. Demzufolge muss die Klasse *Gültigkeit/Erreichbarkeit ab bestimmtem Zeitpunkt* ausgewählt werden.

8. Selektionsschritt:

Kriterium: **Art der Gültigkeits-/Erreichbarkeitsdauer**

Klassifikationsfrage: Bis wann genau soll die geforderte Eigenschaft dauern?

Diskussion: Die geforderte Eigenschaft muss mindestens bis unmittelbar vor dem Zeitpunkt gültig sein, zu dem die Bedingung des *Eintretens des Ausschaltbefehls von der FFB-Zentrale* erfüllt ist. Somit muss die Eigenschaft mindestens bis zum Systemzustand unmittelbar vor dem Ausschaltbefehl erfüllt sein. Infolgedessen muss die Sicherheitsanforderung der Klasse *Gültigkeit/Erreichbarkeit bevor ein bestimmter Zeitpunkt eintritt* zugewiesen werden.

Das Safety-Pattern, das auf der Grundlage der erläuterten Selektionsschritte von SAPIS identifiziert wird, ist in Abschnitt 8.4 beschrieben.

Lebenslauf

Persönliche Daten

26.06.1974	geboren in Konstanz

Schulbildung

1980 – 1984	Sonnenhaldeschule (Grundschule), Konstanz
1984 – 1990	Theodor-Heuss-Realschule, Konstanz, Abschluss Mittlere Reife
1990 – 1993	Zeppelin Gewerbeschule (Technisches Gymnasium), Konstanz, Abschluss Abitur

Studium

1993 – 1999	Studium der Elektrotechnik an der Universität Stuttgart, Studienmodell: Informationstechnik
	Hilfswissenschaftler am Institut für Systemdynamik und Regelungstechnik, am Institut für Netzwerk- und Systemtheorie und am Institut für Automatisierungs- und Softwaretechnik, Universität Stuttgart
	Auslands-Fachpraktikum bei International Electrical Products in 6th-October-City (Kairo), Ägypten
	Fachpraktikum bei Daimler-Benz AG, Abteilung Entwicklung PKW/Gesamtfahrzeug, Elektrik/Elektronik, Motor in Stuttgart
31.03.1999	Abschluss als Diplom-Ingenieur

Berufstätigkeit

April 1999 – März 2005	Wissenschaftlicher Mitarbeiter am Institut für Automatisierungs- und Softwaretechnik der Universität Stuttgart
	Bearbeitung des Projekts *SafeRail* mit dem Thema *Integration von Methoden zur Spezifikation und Verifikation von Sicherungseinrichtungen im spurgeführten Verkehr* im Schwerpunktprogramm *Integration von Techniken der Softwarespezifikation für ingenieurwissenschaftliche Anwendungen* der Deutschen Forschungsgemeinschaft
seit Juli 2005	Mitarbeiter der Informatik Consulting Systems AG in Stuttgart