

# Agentenbasierte dynamische Testfallpriorisierung

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

Vorgelegt von  
Christoph Malz  
aus Danzig

Hauptberichter:	Prof. Dr.-Ing. Dr. h. c. Peter Göhner
Mitberichter:	Prof. Dr.-Ing. Birgit Vogel-Heuser
Tag der Einreichung:	25.09.2012
Tag der mündlichen Prüfung:	13.02.2013

Institut für Automatisierungs- und Softwaretechnik  
der Universität Stuttgart

2013



IAS-Forschungsberichte

Band 2/2013

**Christoph Malz**

**Agentenbasierte dynamische Testfallpriorisierung**

D 93 (Diss. Universität Stuttgart)

Shaker Verlag  
Aachen 2013

**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., 2013

Copyright Shaker Verlag 2013

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

Printed in Germany.

ISBN 978-3-8440-1829-5

ISSN 1610-4781

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen

Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9

Internet: [www.shaker.de](http://www.shaker.de) • E-Mail: [info@shaker.de](mailto:info@shaker.de)

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

Mein besonderer Dank gilt meinem Doktorvater, Herrn Prof. Dr.-Ing. Dr. h. c. Peter Göhner, für die Anregungen und die fortwährende Unterstützung während der Entstehung dieser Arbeit sowie für die Übernahme des Hauptberichts.

Frau Prof. Dr.-Ing. Birgit Vogel-Heuser danke ich für das Interesse an meiner Arbeit und die Übernahme des Mitberichts.

Allen ehemaligen Kolleginnen und Kollegen am IAS gilt mein Dank für die offene und kollegiale Atmosphäre und die gute Zusammenarbeit. Ganz besonders bedanke ich mich bei Dr.-Ing. Nasser Jazdi für die regelmäßigen und hilfreichen Diskussionen über mein Forschungsthema sowie die kritische und gründliche Durchsicht des Manuskripts.

Erwähnen möchte ich auch die vielen Studierenden, die im Rahmen ihrer Diplom-, Master-, Studien- und Bachelorarbeiten einen wichtigen Beitrag zum Gelingen der Arbeit geleistet haben.

Schließlich möchte ich meinen Eltern für die langjährige Unterstützung während meiner gesamten Ausbildungszeit und meiner Frau Susanne für ihr Verständnis, ihre Geduld und ihre Unterstützung zum Gelingen dieser Arbeit herzlich danken.

Stuttgart, im März 2013

Christoph Malz



# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b> .....	<b>iv</b>
<b>Tabellenverzeichnis</b> .....	<b>vi</b>
<b>Abkürzungsverzeichnis</b> .....	<b>vii</b>
<b>Begriffsverzeichnis</b> .....	<b>viii</b>
<b>Zusammenfassung</b> .....	<b>xi</b>
<b>Abstract</b> .....	<b>xii</b>
<b>1 Einleitung</b> .....	<b>1</b>
1.1 Bedeutung qualitativ hochwertiger Software für automatisierte Systeme.....	1
1.2 Bedeutung des Testens und des Testmanagements für qualitativ hochwertige Software.....	1
1.3 Problematik des Testens und des Testmanagements .....	2
1.4 Zielsetzung der Arbeit .....	4
1.5 Gliederung der Arbeit .....	5
<b>2 Grundlagen des Testens und des Testmanagements</b> .....	<b>7</b>
2.1 Aufgabe des Testens .....	7
2.2 Testprozess .....	8
2.2.1 Testplanung und Teststeuerung .....	9
2.2.2 Testspezifikation .....	9
2.2.3 Testausführung.....	10
2.2.4 Testauswertung .....	10
2.3 Teststufen .....	11
2.3.1 Modultest .....	12
2.3.2 Integrationstest.....	12
2.3.3 Systemtest.....	12
2.3.4 Abnahmetest .....	13
2.4 Test nach Änderungen (Regressionstest) .....	13
2.5 Aufgaben des Testmanagements .....	15
2.5.1 Testzyklusplanung und Testzyklussteuerung.....	16
2.5.2 Priorisierung der Testfälle .....	17
2.6 Herausforderungen des Testmanagements .....	22
<b>3 Anforderungen und Ansätze zur Unterstützung der Testfallpriorisierung</b> .....	<b>24</b>
3.1 Anforderungen an die Unterstützung der Testfallpriorisierung .....	24
3.2 Unterstützung durch Computer Aided Software Testing-Werkzeuge (CAST-Werkzeuge) .....	26

3.3	Unterstützung durch Regressionstechniken basierend auf White-Box-Informationen .....	30
3.4	Unterstützung durch wissensbasierte Systeme .....	32
3.5	Unterstützung durch agentenbasierte Systeme .....	34
3.6	Zusammenfassende Bewertung und Folgerung .....	36
<b>4</b>	<b>Grundkonzept der dynamischen Testfallpriorisierung .....</b>	<b>39</b>
4.1	Ziel und Aufgaben des zu konzipierenden Testfallpriorisierungssystems .....	39
4.2	Einflussfaktoren auf die Testfallpriorisierung .....	43
4.2.1	Lebenszyklusphasen des Softwaresystems mit Einfluss auf die Testfallpriorisierung .....	44
4.2.2	Informationen und Einflussfaktoren aus der Systementwicklung .....	45
4.2.3	Informationen und Einflussfaktoren aus dem Systemtest .....	49
4.2.4	Informationen und Einflussfaktoren aus dem Systembetrieb .....	52
4.2.5	Dynamik und Verfügbarkeit der Werte der Einflussfaktoren .....	53
4.3	Priorisierungsregeln bei der Testfallpriorisierung .....	54
4.4	Grundidee für die dynamische Testfallpriorisierung .....	56
4.5	Vorgehen zur Testfallpriorisierung .....	58
4.5.1	Bestimmung der Testwichtigkeit eines Softwaremoduls .....	61
4.5.2	Bestimmung der lokalen Prioritäten eines Testfalls bezüglich der Softwaremodule .....	63
4.5.3	Bestimmung der globalen Priorität eines Testfalls.....	64
4.6	Dynamische Anpassung der globalen Prioritäten zwischen Testzyklen und während eines Testzyklus.....	67
4.7	Bestimmung fehlender Testfälle .....	68
4.8	Agentenbasierter Ansatz für die dynamische Testfallpriorisierung .....	70
4.8.1	Entitätenbasierte Dekomposition der Problemstellung in Softwareagenten ..	71
4.8.2	Integration von Wissen.....	72
4.8.3	Anbindung an Entwicklungs- und Testwerkzeuge.....	74
<b>5</b>	<b>Dynamische Testfallpriorisierung durch den Einsatz von Softwareagenten.....</b>	<b>76</b>
5.1	Agententypen für die dynamische Testfallpriorisierung.....	76
5.1.1	Softwaremodul-Agent .....	77
5.1.2	Testfall-Agent .....	79
5.1.3	Schnittstellen-Agent.....	81
5.2	Bestimmung der Testwichtigkeit eines Softwaremoduls durch den Softwaremodul-Agenten .....	82
5.2.1	Umgebung des Softwaremodul-Agenten .....	82
5.2.2	Wissen des Softwaremodul-Agenten.....	85
5.2.3	Aktionen und Interaktionen des Softwaremodul-Agenten zur Bestimmung der Testwichtigkeit eines Softwaremoduls .....	88
5.3	Bestimmung von lokalen Prioritäten eines Testfalls durch den Testfall-Agenten.....	92
5.3.1	Umgebung des Testfall-Agenten .....	92
5.3.2	Wissen des Testfall-Agenten .....	95
5.3.3	Aktionen und Interaktionen des Testfall-Agenten zur Bestimmung der lokalen Prioritäten eines Testfalls.....	97

5.4	Bestimmung der globalen Priorität eines Testfalls durch den Testfall-Agenten .....	101
5.4.1	Umgebung des Testfall-Agenten .....	101
5.4.2	Wissen des Testfall-Agenten .....	102
5.4.3	Aktionen und Interaktionen des Testfall-Agenten zur Bestimmung der globalen Prioritäten eines Testfalls .....	103
5.5	Ergebnis der agentenbasierten dynamischen Testfallpriorisierung .....	105
5.6	Agentenbasierte Anpassung der globalen Prioritäten .....	107
5.7	Agentenbasierte Bestimmung fehlender Testfälle .....	109
<b>6</b>	<b>Realisierung des agentenbasierten dynamischen Testfallpriorisierungssystems .....</b>	<b>111</b>
6.1	Überblick über das agentenbasierte dynamische Testfallpriorisierungssystem und dessen Umgebung .....	111
6.2	Schnittstellen zu den Informationsquellen .....	113
6.2.1	Schnittstelle zum XML- Architekturmodell .....	113
6.2.2	Schnittstelle zum Konfigurationsmanagementwerkzeug Subversion .....	114
6.2.3	Schnittstelle zur MySQL-Entwickler-Datenbank .....	114
6.2.4	Schnittstelle zum Testmanagementwerkzeug Testopia .....	114
6.2.5	Schnittstelle zum Fehlermanagementwerkzeug Bugzilla .....	115
6.3	Agentenbasiertes dynamisches Testfallpriorisierungssystem basierend auf JADE..	115
6.3.1	Initialisierung des Agentensystems durch den Pionier-Agenten .....	116
6.3.2	Einflussfaktoren und Priorisierungsregeln der realisierten Softwareagenten .....	117
6.3.3	Gesamtablauf der Testfallpriorisierung mit dem realisierten Testfallpriorisierungssystem .....	121
6.3.4	Ergebnisse der agentenbasierten dynamischen Testfallpriorisierung .....	122
<b>7</b>	<b>Evaluierung des agentenbasierten dynamischen Testfallpriorisierungssystems anhand eines Beispielprojekts .....</b>	<b>126</b>
7.1	Beschreibung des Beispielprojekts .....	126
7.2	Evaluierung der Konzeption .....	129
<b>8</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>132</b>
8.1	Ergebnisse der Arbeit .....	132
8.2	Vorteile und Grenzen des Lösungskonzepts .....	134
8.3	Ausblick .....	135
	<b>Literaturverzeichnis .....</b>	<b>137</b>

# Abbildungsverzeichnis

Abbildung 2.1: Phasen des Testprozesses .....	9
Abbildung 2.2: Allgemeines V-Modell mit Teststufen [SpLi05].....	11
Abbildung 2.3: Entwicklungszyklen, Testzyklen und Betriebszyklen .....	14
Abbildung 2.4: Zeitpunkte der Testzyklusplanung und Testzyklussteuerung .....	16
Abbildung 2.5: Beispielhaftes Softwaresystem mit zwei Softwaremodulen.....	18
Abbildung 2.6: Testfälle für beispielhaftes Softwaresystem.....	19
Abbildung 2.7: Erweiterung des beispielhaften Softwaresystems .....	20
Abbildung 3.1: Auflistung von Testfällen im Testmanagementwerkzeug Testopia [TESTOPIA].....	27
Abbildung 3.2: Auflistung von Fehlern im Fehlermanagementwerkzeug Bugzilla [BUGZILLA] .....	28
Abbildung 3.3: Auflistung von Codeänderungen im Konfigurationsmanagementwerkzeug Subversion [SVN] .....	29
Abbildung 3.4: Prinzipieller Ablauf von Regressionstechniken [GHK+98] .....	30
Abbildung 3.5: Aufbau eines Expertensystems [Pupp91] .....	32
Abbildung 3.6: Grundkonzepte der Agentenorientierung.....	34
Abbildung 4.1: Ein- und Ausgänge des Testfallpriorisierungssystems .....	40
Abbildung 4.2: Berücksichtigung der Anforderungen durch das Testfallpriorisierungssystem..	42
Abbildung 4.3: Informationsquellen des Testfallpriorisierungssystems .....	45
Abbildung 4.4: Grundlegende Informationen aus der Systementwicklung .....	46
Abbildung 4.5: Einflussfaktoren aus der Systementwicklung .....	48
Abbildung 4.6: Grundlegende Informationen aus dem Systemtest.....	49
Abbildung 4.7: Einflussfaktoren aus dem Systemtest.....	52
Abbildung 4.8: Informationen und Einflussfaktoren aus dem Systembetrieb .....	53
Abbildung 4.9: Fehlerwahrscheinlichkeiten (FW) und Fehlerfindungswahrscheinlichkeiten (FFW) der Entitäten.....	57
Abbildung 4.10: Lokale Perspektive des Softwaremoduls SM 1 zur Bestimmung der Testwichtigkeit .....	59
Abbildung 4.11: Lokale Perspektive des Testfalls TF B zur Bestimmung der lokalen Prioritäten bezüglich Softwaremodul SM 1 und bezüglich Softwaremodul SM 2.....	60
Abbildung 4.12: Lokale Perspektive des Testfalls TF B zur Bestimmung der globalen Priorität .....	61
Abbildung 4.13: Bestimmung der Testwichtigkeit eines Softwaremoduls.....	62
Abbildung 4.14: Bestimmung der lokalen Priorität eines Testfalls bezüglich eines Softwaremoduls .....	64
Abbildung 4.15: Berechnung der globalen Priorität eines Testfalls .....	65
Abbildung 4.16: Lokale Perspektive des Testfalls TF B zur Bestimmung der globalen Priorität.....	66
Abbildung 4.17: Ansatz für die Umsetzung des dynamischen Testfallpriorisierungssystems ...	71
Abbildung 4.18: Fuzzy-Regelwerk .....	74
Abbildung 5.1: Gesamtansatz für die Umsetzung des Grundkonzepts .....	76
Abbildung 5.2: Repräsentation eines Softwaremoduls durch einen Softwaremodul-Agenten ...	77
Abbildung 5.3: Vertretung aller Softwaremodule durch jeweils einen Softwaremodul-Agenten.....	78
Abbildung 5.4: Bestimmung der Testwichtigkeit durch den Softwaremodul-Agenten .....	79
Abbildung 5.5: Repräsentation eines Testfalls durch einen Testfall-Agenten.....	79

Abbildung 5.6: Vertretung aller Testfälle durch jeweils einen Testfall-Agenten .....	80
Abbildung 5.7: Bestimmung der lokalen Prioritäten und der globalen Priorität durch den Testfall-Agenten .....	81
Abbildung 5.8: Umgebung des Softwaremodul-Agenten zur Bestimmung der Testwichtigkeit .....	85
Abbildung 5.9: Fuzzy-Regelwerk des Softwaremodul-Agenten zur Bestimmung der Testwichtigkeit .....	86
Abbildung 5.10: Fuzzifizierungsfunktion für „Anzahl der geänderten Codezeilen“ .....	87
Abbildung 5.11: Interaktionspartner und Interaktionen des Softwaremodul-Agenten zur Bestimmung der Testwichtigkeit .....	88
Abbildung 5.12: Aktionen und Interaktionen des Softwaremodul-Agenten zur Bestimmung der Testwichtigkeit .....	91
Abbildung 5.13: Umgebung des Testfall-Agenten zur Bestimmung der lokalen Prioritäten .....	95
Abbildung 5.14: Fuzzy-Regelwerk des Testfall-Agenten zur Bestimmung der lokalen Priorität .....	96
Abbildung 5.15: Interaktionspartner und Interaktionen des Testfall-Agenten zur Bestimmung der lokalen Prioritäten .....	98
Abbildung 5.16: Aktionen und Interaktionen des Testfall-Agenten zur Bestimmung der lokalen Prioritäten .....	100
Abbildung 5.17: Umgebung des Testfall-Agenten zur Bestimmung der globalen Priorität .....	102
Abbildung 5.18: Wissen des Testfall-Agenten zur Bestimmung der globalen Priorität .....	102
Abbildung 5.19: Interaktionspartner und Interaktionen des Testfall-Agenten zur Bestimmung der globalen Priorität .....	103
Abbildung 5.20: Aktionen und Interaktionen des Testfall-Agenten zur Bestimmung der globalen Prioritäten .....	104
Abbildung 5.21: Ergebnis der Testfallpriorisierung .....	106
Abbildung 6.1: Agentenbasiertes dynamisches Testfallpriorisierungssystem mit Umgebung ..	111
Abbildung 6.2: Auslesen von Informationen aus dem XML-Architekturmodell .....	113
Abbildung 6.3: Auslesen von Informationen aus Subversion .....	114
Abbildung 6.4: Auslesen von Informationen aus der Entwickler-Datenbank .....	114
Abbildung 6.5: Auslesen von Informationen aus Testopia .....	115
Abbildung 6.6: Auslesen von Informationen aus Bugzilla .....	115
Abbildung 6.7: Instanzierte Softwareagenten des agentenbasierten Testfallpriorisierungssystems .....	117
Abbildung 6.8: Gesamttablauf der agentenbasierten dynamischen Testfallpriorisierung .....	122
Abbildung 6.9: Priorisierungs-Benutzungsoberfläche .....	123
Abbildung 6.10: Testwichtigkeiten-Benutzungsoberfläche .....	124
Abbildung 6.11: Werte der Einflussfaktoren für die Testwichtigkeit eines Softwaremoduls ...	125
Abbildung 6.12: Ausgangswert der Testwichtigkeit für ein Softwaremodul .....	125
Abbildung 7.1: Architekturmodell des Beispielprojekts .....	126
Abbildung 7.2: Subversion-Log-Datei [SVN] .....	127
Abbildung 7.3: Informationen zu einem Testzyklus in Testopia [TESTOPIA] .....	128
Abbildung 7.4: Informationen zu einem Fehler in Bugzilla [BUGZILLA] .....	128
Abbildung 7.5: Vergleich des Fehlerentdeckungsgrads, der Testeffektivität und der Testeffizienz .....	130
Abbildung 7.6: Vergleich der Priorisierungszeit .....	131

## Tabellenverzeichnis

Tabelle 3.1: Vergleich der Ansätze zur Unterstützung der Testfallpriorisierung .....	37
Tabelle 7.1: Ergebnisse der Evaluierung .....	130

## Abkürzungsverzeichnis

CASE	Computer Aided Software Engineering
CAST	Computer Aided Software Testing
DB	Datenbank
DOM	Document Object Mode
F	Funktion
FFW	Fehlerfindungswahrscheinlichkeit
FW	Fehlerwahrscheinlichkeit
GP	Globale Priorität
IAS	Institut für Automatisierungs- und Softwaretechnik
JADE	Java Agent DEvelopment Framework
LP	Lokale Priorität
RPC	Remote Procedure Call
SA	Schnittstellen-Agent
SM	Software <b>m</b> odul
SQL	Structured Query Language
SVN	Subversion
TF	Testfall
TW	Testwichtigkeit
TZP	Testzyklusplanung
TZS	Testzyklussteuerung
UML	Unified Modeling Language
XML	Extensible Markup Language

## Begriffsverzeichnis

**Automatisierungssoftware:** Menge an Programmen zum Betrieb eines rechnerbasierten automatisierten Systems inklusive Dokumentation [Lind08].

**Black-Box-Verfahren:** Ein Verfahren zur Herleitung und Auswahl von Testfällen. Es basiert auf einer Analyse der funktionalen oder nicht-funktionalen Anforderungen (Spezifikationen) eines Softwaremoduls oder Softwaresystems ohne die Berücksichtigung ihrer internen Struktur [ISTQB09].

**Defekt:** Zustand eines Softwaresystems oder einer seiner Softwaremodule, der unter spezifischen Bedingungen eine geforderte Funktion des Softwaresystems beeinträchtigen kann bzw. zu einer Fehlerwirkung führt [ISTQB09].

**Fehlerfindungsrate:** Anzahl der Fehler gewichtet mit der Fehlerschwere, die in einer Teststufe gefunden wurden, dividiert durch die Gesamtzahl der Fehler, die in dieser Teststufe und danach mit jeglichen Mitteln gefunden wurden [SRWL08].

**Fehlerschwere:** Der Grad der Auswirkung, den ein Fehler auf Entwicklung oder Betrieb eines Softwaremoduls oder eines Softwaresystems hat [ISTQB09].

**Fehlerwirkung:** Abweichung eines Softwaremoduls oder eines Softwaresystems von der erwarteten Lieferung, Leistung oder dem Ergebnis [ISTQB09].

**Fehlerzustand:** Zustand eines Softwaresystems oder einer seiner Softwaremodule, der unter spezifischen Bedingungen eine geforderte Funktion des Softwaresystems beeinträchtigen kann bzw. zu einer Fehlerwirkung führt [ISTQB09].

**Integrationstest:** Testen mit dem Ziel, Fehler in den Schnittstellen und im Zusammenspiel zwischen integrierten Softwaremodulen aufzudecken [ISTQB09].

**Modultest:** Testen eines einzelnen Softwaremoduls [ISTQB09].

**Regressionstest:** Erneutes Testen eines bereits getesteten Testobjekts bzw. einer Teilfunktionalität nach deren Modifikation. Ziel ist es nachzuweisen, dass durch die vorgenommenen Änderungen keine Fehlerzustände eingebaut oder (bisher maskierte Fehlerzustände) freigelegt wurden [ISTQB09].

**Softwareagent:** In der agentenorientierten Softwareentwicklung ist ein Softwareagent das Konzept einer abgrenzbaren Softwareeinheit mit einem definierten Ziel. Ein Softwareagent versucht, dieses Ziel durch autonomes Verhalten zu erreichen und interagiert dabei kontinuierlich mit seiner Umgebung und anderen Softwareagenten [VDI2653-1].

**Systemtest:** Testen eines integrierten Softwaresystems um sicherzustellen, dass es spezifizierte Anforderungen erfüllt [ISTQB09].

**Testaktivität:** Aktivität bzw. Teilaufgabe, die innerhalb des Testprozesses ausgeführt wird [SRWL08].

**Testdaten:** Daten, die vor der Ausführung eines Testfalls existieren, und die die Ausführung des Softwaremoduls bzw. des Softwaresystems beeinflussen bzw. dadurch beeinflusst werden [ISTQB09].

**Testeffektivität:** Anzahl gefundener Fehler gewichtet mit der Fehlerschwere pro Testfall [SRWL08].

**Testeffizienz:** Anzahl gefundener Fehler gewichtet mit der Fehlerschwere pro Zeiteinheit [SRWL08].

**Testendekriterien:** Kriterien, die vorab festgelegt werden und erfüllt sein müssen, um eine (Test-) Aktivität als abgeschlossen bewerten zu können [SpLi05].

**Testfall:** Ein Testfall umfasst folgende Angaben: Die für die Durchführung notwendigen Vorbedingungen, die Menge der Eingabewerte, die Menge der vorausgesagten Ergebnisse sowie die erwarteten Nachbedingungen [SRWL08].

**Testmanagement:** Planung, Aufwandsschätzung, Überwachung und Kontrolle von Testaktivitäten, die üblicherweise durch einen Testmanager erfolgen [ISTQB09].

**Testmethode:** Planmäßiges, auf einem Regelwerk aufbauendes Vorgehen zur Herleitung und/oder Auswahl von Testfällen [SpLi05].

**Testobjekt:** Die Softwaremodule oder das Softwaresystem, welches getestet wird [ISTQB09].

**Testphase:** Eine abgegrenzte Menge von Testaktivitäten, die einer Projektphase zugeordnet sind [ISTQB09].

**Testplan:** Eine Liste von Aktivitäten, Aufgaben oder Ereignissen des Testprozesses, mit Angabe ihrer geplanten Anfangs- und Endtermine sowie ihrer gegenseitigen Abhängigkeiten [ISTQB09].

**Testprojekt:** Im Wesentlichen ein einmaliges Vorhaben, bei dem innerhalb einer definierten Zeitspanne definierte Testziele erreicht werden sollen; üblicherweise Teil eines Software- oder Systementwicklungsprojekts [SRWL08].

**Testprozess:** Der fundamentale Testprozess umfasst die folgenden Aktivitäten: Testplanung und Teststeuerung, Testspezifikation, Testausführung und Testauswertung [ISTQB09].

**Testressourcen:** Die Elemente, die für die Durchführung des Testfalls benötigt werden, bestehend aus Testumgebung, Testwerkzeugen und gegebenenfalls Personal [SRWL08].

**Testskript:** Anweisungen zur automatischen Ausführung eines Testfalls in einer geeigneten Programmiersprache [ISTQB09].

**Teststeuerung:** Als Teststeuerung bezeichnet man die Managementaufgaben zur Entwicklung und Anwendung von Korrekturmaßnahmen, um in einem Testprojekt eine Abweichung vom geplanten Vorgehen zu beherrschen [ISTQB09].

**Teststrategie:** Abstrakte Beschreibung der vorgesehenen Teststufen und der Art und Weise, wie innerhalb dieser Teststufe vorzugehen ist, für eine Organisation oder ein Testobjekt – gültig für ein oder mehrere Projekte [ISTQB09].

**Teststufe:** Eine Teststufe ist eine Gruppe von Testaktivitäten, die gemeinsam ausgeführt und verwaltet werden. Teststufen sind mit Zuständigkeiten in einem Projekt verknüpft. Beispiele für Teststufen sind der Modultest, der Integrationstest und der Systemtest [ISTQB09].

**Test Suite:** Die Zusammenstellung mehrerer Testfälle für den Test eines Softwaremoduls oder eines Softwaresystems [ISTQB09].

**Testumgebung:** Eine Umgebung, die benötigt wird, um Testfälle auszuführen. Sie umfasst Hardware, Instrumentierung, Simulatoren, Softwarewerkzeuge und andere unterstützende Hilfsmittel [ISTQB09].

**Testziel:** Ein Grund oder Zweck für den Entwurf und die Ausführung von Testfällen [ISTQB09].

**Testzyklus:** Durchlauf durch den Testprozess auf Basis genau einer Version des Testobjekts, an dessen Ende Fehlerkorrektur- und Änderungsaufträge für die Entwickler vorliegen [SpLi05].

**White-Box-Verfahren:** Ein dokumentiertes Verfahren zur Herleitung und Auswahl von Testfällen, basierend auf der internen Struktur eines Softwaremoduls oder eines Softwaresystems [ISTQB09].

## Zusammenfassung

Die zunehmende Komplexität von Automatisierungssoftware erfordert einen immer größer werdenden Testaufwand dieser Softwaresysteme, um sicherzustellen, dass die automatisierten Systeme zuverlässig funktionieren. Auf der einen Seite verfügt der Testmanager für die Durchführung dieser Systemtests über eine große Menge an Testfällen. Auf der anderen Seite sind jedoch die Testzeit und die Anzahl der Testressourcen, die dem Testmanager zur Disposition stehen, begrenzt. Er ist gezwungen, mit hohem Aufwand sowohl von Testzyklus zu Testzyklus als auch gegebenenfalls während eines Testzyklus des Systemtests eine Priorisierung der zur Verfügung stehenden Testfälle durchzuführen, um trotz der begrenzten Testzeit und der begrenzten Testressourcen verlässliche Testresultate abzuliefern und sicherzustellen, dass die Softwaresysteme ausreichend geprüft werden.

In der vorliegenden Arbeit wird ein Konzept vorgestellt, das mithilfe von Softwareagenten diesen manuellen Aufwand reduziert und den Fehlerentdeckungsgrad, die Effektivität und die Effizienz beim Systemtest erhöht. Softwareagenten werten die im Systemtest bereitstehenden Informationen aus der Systementwicklung, dem Systemtest und dem Systembetrieb dynamisch aus und schlagen aktiv und konstruktiv eine Testfallpriorisierung für die Testzyklen des Systemtests vor. Das Ziel der Softwareagenten ist dabei, eine Testfallpriorisierung zu erreichen, die es ermöglicht, in der zur Verfügung stehenden Testzeit möglichst viele und insbesondere schwerwiegende Fehler zu finden. Die Grundidee der agentenbasierten dynamischen Testfallpriorisierung ist, dass Softwareagenten Softwaremodule und Testfälle des zu testenden Softwaresystems vertreten und aus deren Perspektive in Zusammenarbeit die Testfallpriorisierung durchführen. Zur Testfallpriorisierung bestimmen die Softwareagenten Fehlerwahrscheinlichkeiten auf Basis von Informationen, die in bereits in der Industrie eingesetzten Entwicklungs- und Testwerkzeugen vorhanden sind, was die Anwendbarkeit im Systemtest sicherstellt. Je nach Bedarf bzw. Verfügbarkeit können die Informationen, die zur Auswertung eingesetzt werden, angepasst werden. Die Softwareagenten werten die Informationen mit Priorisierungsregeln aus, die anhand von Erfahrungswissen basierend auf Fuzzy-Regelwerken definiert wurden. Auch die Priorisierungsregeln können je nach Bedarf angeglichen werden. Die Softwareagenten arbeiten aktiv im Hintergrund. Sie registrieren die Veränderung von Informationen aufgrund dynamischer Ereignisse und reagieren auf diese mit der Anpassung der Testfallpriorisierung. Ein Testfallpriorisierungssystem basierend auf dem Konzept der agentenbasierten dynamischen Testfallpriorisierung wurde realisiert und evaluiert. Die Evaluierungsergebnisse zeigen eine deutliche Steigerung des Fehlerentdeckungsgrads, der Testeffektivität und der Testeffizienz sowie eine Reduzierung des zeitlichen Aufwands für die Testfallpriorisierung im Vergleich zu einer Testfallpriorisierung durch einen Menschen mithilfe von herkömmlichen Entwicklungs- und Testwerkzeugen.

## Abstract

The rising complexity of industrial automation software requires an increasing test effort of these software systems in order to ensure that the automated systems work reliable. On one side, a high number of test cases for the execution of these system tests are available for the test manager. However, on the other side, the test time and the extent of test resources, which are available for the test manager, are restricted. The test manager has to execute a prioritization of the available test cases with high effort for each test cycle and, if needed, during the cycle of the system test, in order to deliver reliable test results and to ensure that the software system is tested sufficiently despite the restricted test time and the restricted test resources.

In this work a concept is introduced, which with the help of software agents reduces the manual effort and increases the fault detection rate, the test effectiveness and the test efficiency at the system test. Software agents evaluate dynamically the information from the system development, the system test and the system operation, which is available in the system test, and recommend actively and constructively the test case prioritization for the test cycles of the system test. The goal of the software agents is to reach a test case prioritization, which allows finding as much as possible faults and especially severe faults in the available test time. The basic idea of the agent-based dynamic test case prioritization is that software agents represent software modules and test cases of the software system to be tested and in cooperation execute the test case prioritization from their perspective. The software agents determine fault probabilities for the test case prioritization based on information, which is available in development and test tools used already in industry. This ensures the applicability of the concept in the system test. The information which is used for evaluation can be adapted dependent on the demand and availability. The software agents evaluate the information with prioritization rules which are defined with fuzzy logic on the basis of experience knowledge. The prioritization rules can be also adapted as needed. The software agents work actively in the background. They register modifications of information due to dynamic events and react to these with modifications of the test case prioritization. The test case prioritization system based on the concept of the agent-based dynamic test case prioritization was realized and evaluated. The evaluation results show a considerable increase of the fault detection rate, test effectiveness, test efficiency and a reduction of the time effort for the test case prioritization in comparison to a test case prioritization by a human being using conventional development and test tools.