Informatik 89

Modellbasierte Extraktion, Repräsentation und Analyse von Traceability-Informationen

von
Josef Adersberger

1. Auflage

Utz, Herbert 2013

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 8316 4193 2

Josef Adersberger

Modellbasierte Extraktion, Repräsentation und Analyse von Traceability-Informationen

Herbert Utz Verlag · München

Informatik

Band 89

Coverbild: Entwurf von Martin Gassner für die QAware GmbH

Zugl.: Diss., Universität Erlangen-Nürnberg, 2012

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.

ISBN 978-3-8316-4193-2

Printed in EC
Herbert Utz Verlag GmbH, München
089-277791-00 · www.utzverlag.de

Danksagung

Diese Dissertation wäre niemals ohne die Hilfe, das Wissen, den Beistand und die Passion vieler wundervoller Menschen entstanden.

Mein besonderer Dank gilt Herrn Professor Kips für die zielführende und stets konstruktive Betreuung. Ebenso geht mein Dank an Herrn Professor Philippsen für die Zusammenarbeit im Forschungsprojekt Softwareleitstand, das die Grundlagen dieser Arbeit geschaffen hat. Herrn Professor Riehle, Herrn Professor Meyer-Wegener und Herrn Professor Schmauss will ich für das reibungslose Promotionsverfahren danken.

Herzlich bedanken will ich mich auch bei meinen Kollegen bei der QAware. Besonders danken will ich dabei Johannes Weigend, Christian Kamm und Bernd Schlüter, die mir stets den Freiraum dafür gegeben haben, mein Promotionsprojekt erfolgreich abschließen zu können.

Ebenso will ich mich bei den Menschen bedanken, die mir wichtige Anregungen für meine wissenschaftliche Arbeit gaben: Norbert Tausch, Norbert Oster, Dirk Wischermann, Florin Pinte, Sven Söhnlein, Marc Spisländer, Frau Professor Saglietti, Henrik Kemmesies, Andreas Janning, Christian Neuhaus, Samir Al-Hilank, Johannes Drexler, Ralf Ellner, Martin Jung, Johannes Weigend, Herr Professor Gerd Beneken, Herr Professor Reiner Hüttl und Herr Professor Johannes Siedersleben.

Christine Kantsperger, Nancy Pfeifer und Josef Adersberger sen. danke ich sehr für ihre Jagd nach Rechtschreibfehlern in dieser Arbeit.

Mein innigster Dank geht an meine Freunde und besonders auch an meine Familie für ihre Unterstützung und Nachsicht.

Amélie, Du bist die Energie, die diese Arbeit zu einem erfolgreichen Ende geführt hat.

Nina, diese Arbeit ist Dir in Liebe und Dankbarkeit gewidmet.

Kurzfassung

Diese Arbeit beschreibt und evaluiert die *TraceML*, eine Modellierungssprache für Traceability-Informationen. Traceability ist der Grad, in dem Zusammenhänge zwischen Erzeugnissen im Entwicklungsprozess von Software verfolgt werden können. Die Erhebung und Verarbeitung von Traceability-Informationen hilft dabei, die Komplexität moderner Softwareentwicklung besser zu beherrschen und diese damit effizienter zu machen.

Aktuell hat die Traceability ein Problem in ihrer Anwendung: Sie ist oft teuer und wenig nützlich. Es fehlt an durchgängigen und wirksamen Ansätzen und Werkzeugen. Genau hier leistet die TraceML einen Beitrag. Sie ist eine Modellierungssprache der Traceability, mit der hierzu Ansätze und Anwendungen einheitlich und durchgängig formuliert werden können.

Die Arbeit beschreibt zunächst ein Ordnungsschema für die Informationen und den Prozess der Traceability über eine Ontologie und eine Wertschöpfungskette. Darauf aufbauend wird die TraceML in ihrer Struktur und Semantik beschrieben. Ihre Tauglichkeit wird dann über zwei Anwendungsbeispiele evaluiert: *ReflexML*, eine Analyse der Konsistenz zwischen Architektur und Code, und über einen *Softwareleitstand*, der Traceability-Informationen und Softwaremetriken zur Steuerung von Softwareprojekten verdichtet. ReflexML stellt darüber hinaus einen eigenständigen wissenschaftlichen Beitrag im Bereich der Softwarearchitektur dar.

Abstract

This work describes and evaluates *TraceML*, a modeling language for traceability information. Traceability is the degree to which relationships between products of a software development process can be established. Processing traceability information helps to handle the complexity of modern software development efforts and so to make software development more efficient.

Up to date traceability has a problem in its application: Often it's too expensive and of little use. Integrated and efficient approaches and tools for traceability are missing. TraceML contributes to solve this problem. It proposes a modeling language which allows to formulate traceability approaches and applications uniformly.

First of all this work describes an organization scheme for the information and the process of traceability by an ontology and a value chain. Following TraceML is described in its structure and semantics. The capability of the approach is then evaluated in two applications: *ReflexML*, consistency analysis between architecture and code, and the *Softwareleitstand* a tool which condenses traceability information and software metrics to control software projects more efficiently. ReflexML is a dedicated research contribution in the field of software architectures.

Inhaltsverzeichnis

1. Einleitung

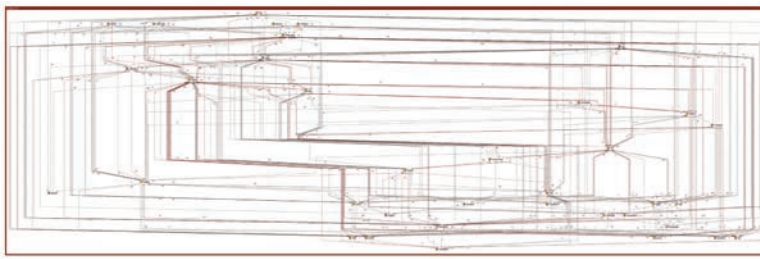

Abbildung 1.1.: Abhängigkeiten im Code einer Software als Graph dargestellt

Software gehört zu den komplexesten Ergebnissen menschlichen Schaffens. Softwaresysteme setzen heutzutage nicht selten tausende Anforderungen in zigmillionen Zeilen Code um. Der Bedarf, mehr und größere Softwaresysteme zu erstellen, steigt.

Gleichzeitig steigt aber auch im internationalen Wettbewerb der Druck, Software mit hoher Produktivität zu entwickeln. Die steigende Komplexität darf also zu keinen Produktivitätseinbußen führen. Wir benötigen Mittel, mit denen wir die Komplexität der Softwareentwicklung effizient beherrschen können.

1.1. Motivation

Ein Mittel, diese Komplexität besser beherrschen zu können, ist die Traceability. Das bedeutet, die Zusammenhänge zwischen allen Zwischen- und Endergebnissen der Softwareentwicklung verstehen, beschreiben und analysieren zu können; die Spur einer Anforderung, einer Architekturkomponente oder auch eines Stakeholders im Entwicklungsprozess verfolgen zu können. Traceability wird von verbreiteten Standards, wie z.B. CMMI, SPICE und dem V-Modell XT, gefordert und von der Wissenschaft und Praxis als wertvoll und nützlich empfunden. Der Nutzen der Traceability liegt darin, übergreifende Fragestellungen - über alle Phasen und Ergebnisse der Softwareentwicklung hinweg -

beantworten zu können: Ist unser Code konsistent zur Softwarearchitektur? Wie viele Anwendungsfälle haben wir bereits umgesetzt? Welche Tests verifizieren das? Welche Codeteile muss ich ändern, wenn ich die Spezifikation eines bestimmten Anwendungsfalls anpasse? Welcher Entwickler hat an einer Komponente mitprogrammiert? Bekommt man Antworten auf diese Fragen, so kann man ein System übergreifend verstehen, die Auswirkungen von Änderungen besser begreifen und Inkonsistenzen und Anomalien über alle Phasen hinweg schneller erkennen. Das hilft dabei, die Komplexität der Softwareentwicklung besser zu beherrschen.

Obwohl die Vorteile durch Traceability im Entwicklungsprozess offensichtlich sind, gilt sie in der Industrie und Wissenschaft immer noch als schwer anzuwenden. Der Aufwand zur Erhebung und Pflege von Traceability-Informationen gilt als hoch. Der entsprechende Prozess gilt als fehleranfällig. Grund dafür sind besonders die folgenden drei Probleme: Eiserne Vorhänge, Traceability-Mikronesien und fehlendes Augenmaß.

Eiserne Vorhänge: Wir erheben und spezifizieren Anforderungen und leiten daraus Systemfunktionen ab. Wir entwerfen Softwarearchitekturen. Wir entwickeln Code und Testfälle. Die Entwicklung von Software ist mannigfaltig, die verschiedenen Disziplinen jedoch oft schlecht miteinander integriert. Die einzelnen Disziplinen sind isoliert. Die Grenzen an den Übergängen zwischen ihnen sind oft dicht - eiserne Vorhänge. Grenzübertritte erfolgen weitestgehend unkontrolliert über unbefestigte Pfade. Das bedeutet: Traceability-Informationen liegen verteilt in heterogenen Werkzeugen vor. Verbindungen zwischen den Ergebnissen in den unterschiedlichen Phasen der Softwareentwicklung liegen oft nur implizit vor und können deshalb nur schwer erschlossen werden. Oder sie müssen aufwändig gepflegt werden, wie z.B. am Übergang zwischen Anwendungsfällen und der Architektur oder der Architektur und dem Code. Vorgehensmodelle beschreiben oft ausführlich einzelne Disziplinen, aber kaum die Übergänge zwischen den Disziplinen. Was besonders fehlt, sind effiziente und durchgängige Werkzeuge, die Traceability-Informationen mit hohem Automatisierungsgrad verarbeiten. *Es gilt Grenzen zu sprengen.*

Traceability-Mikronesien: Es gibt eine Vielzahl an wissenschaftlichen Ansätzen auf dem Gebiet der Traceability, die jedoch oft nur kleine Insellösungen darstellen. Es fehlt ein gemeinsamer Unterbau. Die Ansätze sind schlecht miteinander kombinierbar, ein Insel-Hopping ist aufwändig und zum Teil unmöglich. Dadurch ist aktuell noch kein Ökosystem an Ansätzen und Lösungen zur Traceability entstanden. Es mangelt an einheitlichen Definitionen und uniformen Modellen zur Beschreibung und Verarbeitung von Traceability-Informationen. *Es gilt Brücken zu bauen.*

Fehlendes Augenmaß: Die Gewährleistung von Traceability im Entwicklungsprozess wirkt mitunter ausschließlich für die Erfüllung von Vorgaben aus Standards bestimmt. Dabei stehen oft die Interessen der Produzenten und Konsumenten von Traceability-Informationen im Konflikt: Die Konsumenten möchten so viele Zusammenhänge wie

möglich zugänglich haben, die Produzenten akzeptieren Traceability als Nebenprodukt ihrer Arbeit, nicht aber als Aufwandstreiber. Traceability ist der Grad, in dem Zusammenhänge zwischen Erzeugnissen im Entwicklungsprozess von Software verfolgt werden können. Es stellt sich hier die Frage: Was ist der richtige Grad? Darauf liefert kein Standard eine Antwort oder impliziert mit wenig Augenmaß 100%. Es mangelt an pragmatischen Anwendungen der Traceability, bei denen der Nutzen und Wert für ein Projekt im Zentrum stehen: Wie nutzen wir die Traceability-Informationen? Was ist der Mehrwert für uns, der daraus entsteht? *Es gilt Augenmaß zu wahren.*

Der Beitrag dieser Arbeit zur Lösung dieser Probleme ist eine Modellierungssprache der Traceability, *TraceML*, mit der Traceability-Informationen uniform beschrieben und verarbeitet werden können. Auf dieser Basis können Ansätze und Anwendungen der Traceability formuliert und integriert werden. Die TraceML ist fokussiert auf den Nutzen und Mehrwert der Traceability. Sie stellt eine Sprache zur Verfügung. Welche Traceability-Informationen damit formuliert werden, bestimmt der Nutzer. Sie stellt Hilfsmittel entlang der Wertschöpfungskette der Traceability zur Verfügung - für Extraktion, Repräsentation und Analyse. Sie ist damit fokussiert auf die Wertschöpfung und schafft die Grundlage für durchgängige und einheitliche Traceability-Werkzeuge.

Die TraceML wird, im Rahmen der vorliegenden Arbeit, erprobt durch die Formulierung eines umfangreichen Ansatzes und einer durchgängigen Anwendung der Traceability: *ReflexML* ist ein Ansatz zur Ermittlung von Inkonsistenzen zwischen Architektur und Code auf Basis von Traceability-Informationen mit dem Zweck, dem Verfall einer Softwarearchitektur im Code entgegen zu wirken. Der Mehrwert von ReflexML ist es, die Wartbarkeit von Software nachhaltig zu verbessern und damit Kosten in der Wartung und Weiterentwicklung zu sparen. Der *Softwareleitstand* ist eine Anwendung, ein Werkzeug, das Informationen zur Steuerung von Softwareprojekten integriert, aufbereitet und verdichtet. Er integriert dabei Traceability-Informationen und Metriken aus einer Vielzahl an Werkzeugen des Software Engineering aus verschiedenen Disziplinen. Der Mehrwert des Softwareleitstands ist, dass Softwareprojekte mit ihm besser gesteuert werden können.

1.2. Einordnung und Beitrag

Die Arbeit enthält zwei wissenschaftliche Beiträge: Die TraceML und die ReflexML.

Die TraceML (siehe Kapitel 3) ist eine Sprache zur Beschreibung und Verarbeitung von Traceability-Informationen und ist damit der Forschung im Bereich der Traceability im Software Engineering zuzuordnen. Die TraceML adaptiert eine Reihe an wissenschaftlichen Ansätzen auf diesem Gebiet (siehe Abschnitt 3.7), stellt darüber hinaus aber auch einen eigenständigen wissenschaftlichen Beitrag dar:

- Sie definiert ein Ordnungsschema sowohl für die Informationen als auch für den Prozess der Traceability durch die Beschreibung einer Ontologie (siehe Abschnitt 2.1.2) und einer Wertschöpfungskette (siehe Abschnitt 2.1.3).
- Sie ist der einzige bekannte modellbasierte Ansatz, der die komplette Wertschöpfungskette der Traceability unterstützt (siehe Abschnitt 3.5).
- Sie enthält ein Schema zur rigorosen Beschreibung der Semantik von Traceability Links, das bekannten Ansätzen auf dem Gebiet überlegen ist (siehe Abschnitt 3.3.2).
- Sie ist formal fundiert beschrieben, sowohl über ein formalisiertes Metamodell als auch über eine operationale Semantik.

ReflexML (siehe Kapitel 4) ist ein Ansatz zur Analyse der Konsistenz zwischen Architektur und Code. Es handelt sich dabei um einen eigenständigen Beitrag, der grundsätzlich unabhängig von der TraceML ist, jedoch zur praktischen Anwendung auf ihrer Basis formuliert ist. ReflexML ist wissenschaftlich sowohl dem Bereich der Traceability als auch dem der Softwarearchitekturen zuzuordnen. Der wesentliche wissenschaftliche Beitrag der ReflexML ist:

- Es ist der einzige bekannte Ansatz, der es erlaubt, die Traceability von Architektur auf Code direkt über die UML zu beschreiben. Die Traceability-Informationen können in bestehenden UML-Komponentenmodellen über ein entsprechendes UML-Profil mit Reflexion-Ausdrücken (siehe Abschnitt 4.3) beschrieben werden. Diese Reflexion-Ausdrücke formulieren auf Basis einer Technik der aspektorientierten Programmierung ausdrucksstark und evolutionsstabil die Abbildung von Architekturelementen auf Code.
- Der Ansatz bietet eine fundierte Analyse der Architektur-zu-Code-Konsistenz durch einen Satz an vordefinierten Konsistenzprüfregeln (siehe Abschnitt 4.4), die auf der Semantik von komponenten-orientierten Architekturen basieren. Dieser Satz an Regeln ist reichhaltiger als in anderen bekannten Ansätzen.

1.3. Übersicht

Der folgende Abschnitt bietet eine Übersicht zum Aufbau der vorliegenden Arbeit. Die Arbeit gliedert sich in vier Teile:

1. Die Beschreibung der wissenschaftlichen Grundlagen (Kapitel 2).
2. Die Beschreibung der TraceML als Modellierungssprache für Traceability-Informationen (Kapitel 3).

3. Die Beschreibung der ReflexML als Ansatz zur Konsistenzanalyse zwischen Architektur und Code sowie dessen Abbildung auf die TraceML (Kapitel 4).

4. Eine weitere Evaluierung der TraceML im Rahmen eines Werkzeugs, einem Prototypen für einen Softwareleitstand (Kapitel 5).

Die Beschreibung der wissenschaftlichen Grundlagen der Arbeit gliedert sich auf in die Grundlagen zur Traceability (Abschnitt 2.1) sowie den Grundlagen zur Modellierung und Metamodellierung (Abschnitt 2.2). Als Grundlagen zur Traceability werden wichtige Begriffe definiert (Abschnitt 2.1.1) und in eine Ontologie eingeordnet (Abschnitt 2.1.2) sowie die Traceability-Wertschöpfungskette (Abschnitt 2.1.3) und der aktuelle Stand der Wissenschaft zu dem Thema beschrieben (Abschnitt 2.1.4). Als Grundlage zur Modellierung und Metamodellierung wird zunächst der allgemeine Modellbegriff eingeführt (Abschnitt 2.2.1) und dann dessen Anwendung in der Modellierung und Metamodellierung im Software Engineering beschrieben (Abschnitt 2.2.2). Abschließend werden Festlegungen zur Modellierung im Rahmen dieser Arbeit getroffen (Abschnitt 2.2.3).

Kapitel 3 beschreibt die TraceML. Dabei werden zunächst die Grundlagen der TraceML erarbeitet: Der Ansatz zur Beschreibung von Traceability-Link-Typen, das Link-Schema (Abschnitt 3.3.2), sowie ein formales Modell der relevanten Traceability-Informationen (Abschnitt 3.3.1). Auf Basis dieser Grundlagen wird dann die TraceML beschrieben. Sie ist untergliedert in die Teile *Infrastructure* und *Superstructure*. Die Infrastructure enthält den allgemeinen Sprachkern (Abschnitt 3.4). Die Superstructure enthält die eigentliche Modellierungssprache, formuliert auf Basis des Sprachkerns (Abschnitt 3.5). Die Beschreibung der TraceML wird abgeschlossen mit einem Anwendungsbeispiel, einer Betrachtung der Einflüsse aus anderen wissenschaftlichen Arbeiten (Abschnitt 3.7) sowie einer Beschreibung, wie die TraceML im Rahmen der vorliegenden Arbeit validiert wurde (Abschnitt 3.8).

Der ReflexML-Ansatz ist im Kapitel 4 beschrieben. Er besteht aus einer Möglichkeit, die Traceability zwischen Architektur und Code zu beschreiben (Abschnitt 4.3), sowie einer Reihe an Konsistenzprüfregeln bezogen auf diese Traceability-Informationen (Abschnitt 4.4). Es wird die Fallstudie beschrieben, mit der ReflexML erprobt wurde (Abschnitt 4.7) und die Formulierung der ReflexML auf Basis der TraceML erörtert (Abschnitt 4.5).

Abschließend wird die Anwendung der TraceML im Softwareleitstand beschrieben (Abschnitt 5.3). Zusätzlich werden die grundlegenden Konzepte eines Softwareleitstands eingeführt (Abschnitt 5.1) und die Architektur des Prototypen beschrieben (Abschnitt 5.2).

Eilige Leser, die ausschließlich am wissenschaftlichen Beitrag dieser Arbeit interessiert sind, können dem Lesepfad entlang der Referenzen im vorangegangenen Abschnitt 1.2 folgen. Allen anderen Lesern empfiehlt sich ein sequenzieller Lesepfad durch die Arbeit.

1.4. Konventionen

Zur Typographie und Orthographie gelten im Rahmen der Arbeit die folgenden Konventionen:

- Reihung von Substantiven: Eine Reihe an deutschen Substantiven wird entweder zusammengeschrieben oder per Bindestrich verbunden, wenn dies die Lesbarkeit verbessert. Eine Reihe an englischen Substantiven werden einzeln aufgereiht und weder zusammengeschrieben noch per Bindestrich verbunden. Sind deutsche und englische Substantive kombiniert in Reihe, so gilt die Regel für deutsche Substantive für die gesamte Reihe.
- Namen von Modell- und Code-Elementen: Die Namen von Modell- oder Code-Elementen sind im Fließtext in `Schreibmaschinenschrift` gesetzt.
- Begriffe und Konzepte sind bei ihrer ersten Verwendung in einem Kapitel *kursiv* gesetzt.
- Hyperlinks sind ohne das Datum des letzten Zugriffs angegeben. Das Datum des letzten Zugriffs für alle Links ist der 9. Februar 2012. An diesem Tag wurden alle angegebenen Links noch einmal auf ihren Inhalt und ihre Gültigkeit hin überprüft.

Informatik

Band 89: Josef Adersberger: **Modellbasierte Extraktion, Repräsentation und Analyse von Traceability-Informationen**
2012 · 300 Seiten · ISBN 978-3-8316-4193-2

Band 88: Karl R. Brendel: **Parallele oder sequentielle Simulationsmethode?** · Implementierung und Vergleich anhand eines Multi-Agenten-Modells der Sozialwissenschaft
2010 · 316 Seiten · ISBN 978-3-8316-4013-3

Band 87: Daniel Motus: **Referenzmodell für die Montageplanung in der Automobilindustrie**
2009 · 204 Seiten · ISBN 978-3-8316-0860-7

Band 86: Joachim Wolfgang Kaltz: **An Engineering Method for Adaptive, Context-aware Web Applications**
2006 · 196 Seiten · ISBN 978-3-8316-0647-4

Band 85: Stephanie Spranger: **Calendars as Types** · Data Modeling, Constraint Reasoning, and Type Checking with Calendars
2006 · 308 Seiten · ISBN 978-3-8316-0564-4

Band 84: Sascha Vogel: **Eine Methode zur Entwicklung flexibler Dienste auf der Basis von Interaktionsmustern**
2004 · 202 Seiten · ISBN 978-3-8316-0334-3

Band 83: Andreas Rausch: **Componentware** · Methodik des evolutionären Architekturentwurfs
2004 · 205 Seiten · ISBN 978-3-8316-0326-8

Band 82: Peter Birke: **Und es geht doch! Wie rational ist irrational?** · Näherungsweise Berechnung von Wurzeln natürlicher Zahlen mittels eines divisionsfreien Algorithmus auf eine Genauigkeit von 1000 geltenden Ziffern
2004 · 64 Seiten · ISBN 978-3-8316-0299-5

Band 81: Claudia Gold: **Framework-basierte Unterstützung bei der Realisierung von Lastverteilung**
2003 · 193 Seiten · ISBN 978-3-8316-0272-8

Band 80: Bärbel Ripplinger: **Linguistic Knowledge in Cross-Language Information Retrieval**
2002 · 182 Seiten · ISBN 978-3-8316-0181-3

Band 79: Andreas Dehmel: **A Compression Engine for Multidimensional Array Database Systems**
2002 · 170 Seiten · ISBN 978-3-8316-0139-4

Band 78: Helmut Reiser: **Sicherheitsarchitektur für ein Managementsystem auf der Basis Mobiler Agenten**
2002 · 259 Seiten · ISBN 978-3-8316-0108-0

Band 77: Peter M. Borst: **An Architecture for Distributed Interpretation of Mobile Programs**
2002 · 576 Seiten · ISBN 978-3-8316-0103-5

Band 76: Holger Schmidt: **Entwurf von Service Level Agreements auf der Basis von Dienstprozessen** · 2. Auflage
2005 · 301 Seiten · ISBN 978-3-8316-0455-5

Band 75: Rainer Hauck: **Architektur für die Automation der Managementinstrumentierung bausteinbasierter Anwendungen**
2001 · 201 Seiten · ISBN 978-3-8316-0056-4

Band 74: Marco Pötke: **Spatial Indexing for Object-Relational Databases**
2001 · 226 Seiten · ISBN 978-3-8316-0043-4

Band 73: Michael Bader: **Robuste, parallele Mehrgitterverfahren für die Konvektions-Diffusions-Gleichung**
2001 · 152 Seiten · ISBN 978-3-8316-0040-3

Band 72: Robert Müller: **Fingerprint Verification with Microprocessor Security Tokens**
2001 · 172 Seiten · ISBN 978-3-8316-0015-1

Band 71: Katharina Spies, Bernhard Schätz (Hrsg.): **Formale Beschreibungstechniken für verteilte Systeme** · 9. GI/ITG Fachgespräch, München, Juni 1999
1999 · 260 Seiten · ISBN 978-3-89675-918-4

Band 70: Ricarda Weber: **Accounting and Payment Concepts for Fee-Based Scientific Digital Libraries**
2000 · 360 Seiten · ISBN 978-3-89675-875-0

Band 69: Dieter A. Bartmann: **Benutzerauthentisierung durch Analyse des Tippverhaltens mit Hilfe einer Kombination aus statistischen und neuronalen Verfahren**
2000 · 204 Seiten · ISBN 978-3-89675-836-1

Band 68: Anton Christian Frank: **Organisationsprinzipien zur Integration von geometrischer Modellierung, numerischer Simulation und Visualisierung**
2000 · 166 Seiten · ISBN 978-3-89675-796-8

Band 67: Markus Podolsky: **Ein durchgängiger, integrierter Ansatz für den Entwurf objektorientierter, Workflow-basierter Systeme**
2000 · 292 Seiten · ISBN 978-3-89675-771-5

Band 66: Thomas Paintmayer: **Einbettung von Sicherheitsverfahren in ein offenes heterogenes Sicherheitsmanagement auf der Basis der OSI Managementarchitektur**
2000 · 250 Seiten · ISBN 978-3-89675-770-8

Band 65: Stephen Heilbronner: **Konzeption einer Architektur für das integrierte Management der Ressourcennutzung nomadischer Systeme in Datennetzen**
2000 · 248 Seiten · ISBN 978-3-89675-733-3

Erhältlich im Buchhandel oder direkt beim Verlag:
Herbert Utz Verlag GmbH, München
089-277791-00 · info@utzverlag.de

Gesamtverzeichnis mit mehr als 3000 lieferbaren Titeln: www.utzverlag.de