Informatik

Norbert Tausch

Eine domänenspezifische Sprache zur Analyse von Software-Verfolgbarkeitsinformationen

SHAKER
VERLAG

Eine domänenspezifische Sprache zur Analyse von Software-Verfolgbarkeitsinformationen

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg
zur
Erlangung des Doktorgrades Dr.-Ing.

vorgelegt von
N o r b e r t T a u s c h
aus Schwabach

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg.

Tag der mündlichen Prüfung: 24. November 2017

Vorsitzender des Promotionsorgans: Prof. Dr.-Ing. Reinhard Lerch

Gutachter: Prof. Dr. Michael Philippsen
Prof. Dr. Bernhard Westfechtel (Universität Bayreuth)

Berichte aus der Informatik

Norbert Tausch

Eine domänenspezifische Sprache zur Analyse von Software-Verfolgbarkeitsinformationen

D 29 (Diss. Universität Erlangen-Nürnberg)

Shaker Verlag
Aachen 2017

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.: Erlangen-Nürnberg, Univ., Diss., 2017

Printed in Germany.

ISBN 978-3-8440-5689-1
ISSN 0945-0807

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

Kurzfassung

Die Software-Verfolgbarkeit ist ein etabliertes Instrument heutiger Software-Entwicklung und wird daher von vielen Entwicklungs- und Qualitätsstandards gefordert. Sie zeigt zeitliche und strukturelle Zusammenhänge zwischen den Software-Artefakten eines Systems auf, erhöht dadurch dessen Transparenz und fördert insgesamt das Verständnis für dieses System.

Die Umsetzung von Verfolgbarkeit umfasst im Wesentlichen die Arbeitsschritte Planung, Erstellung, Wartung und Analyse. Der Schwerpunkt der verfügbaren Veröffentlichungen liegt dabei klar auf der Erstellung und Wartung von Verfolgbarkeitsinformationen und beschreibt hauptsächlich Verfahren und Technologien, die die Erkennung bzw. Wiederherstellung von Verknüpfungen zwischen Artefakten vereinfachen bzw. gar automatisieren. Heute stehen hierzu eine Vielzahl von Werkzeugen bereit, die diesbezüglich Erleichterung versprechen.

Der eigentliche Nutzen von Verfolgbarkeit ergibt sich allerdings durch die Analyse der zur Verfügung stehenden Software-Verfolgbarkeitsinformationen, da nur dadurch die eigentlich gestellten Fragestellungen beantwortet werden können. In der Praxis werden Analysen heute manuell, mittels Werkzeugen oder durch selbst programmierte Analyseroutinen durchgeführt. Insgesamt gesehen, ist die wiederholbare und automatisierte Analyse von Verfolgbarkeitsinformationen aufwändig, insbesondere dann, wenn sie über Werkzeuggrenzen hinweg erfolgen und kontinuierlich angepasst werden muss. Verfolgbarkeit lässt sich daher, trotz des erwiesenen Nutzens, im industriellen Umfeld nur schwer umsetzen. Der hohe Implementierungsaufwand erschwert zudem die Erforschung und Entwicklung neuer Analysearten.

Die vorliegende Dissertation greift das Problem aufwändiger Verfolgbarkeitsanalysen auf, untersucht die Eigenschaften bestehender Ansätze und leitet daraus den eigenen Ansatz der *Traceability Analysis Language* (*TracAL*) ab. Die domänenspezifische Sprache erleichtert die Erstellung von Verfolgbarkeitsanalysen und bietet aufgrund ihrer wesentlichen Eigenschaften Vorteile gegenüber bestehenden Ansätzen:

- *Anwendbarkeit:* Die Sprache bietet die zur Umsetzung wiederholbarer und automatisierbarer Verfolgbarkeitsanalysen notwendige Funktionalität. Besonderer Wert wird dabei auf eine hohe Ausdrucksstärke und eine hohe Performanz gelegt.

- *Repräsentationsunabhängigkeit:* Der Ansatz arbeitet mit einem graphbasierten Modell, das von den einzelnen Datenquellen im Software-Projekt abstrahiert und somit Artefakte und deren Verknüpfungen in einheitlicher Form repräsentiert. Dadurch wird die Erstellung durchgängiger Analysen ermöglicht, die auf die Verfolgbarkeitsinformationen mehrerer Werkzeuge und Datenquellen zurückgreifen kann. Ein mehrstufiges Adapterkonzept ermöglicht zudem die einfache Anbindung neuer, bisher noch nicht unterstützter Datenquellen.

- *Konfigurierbarkeit: TracAL* ermöglicht die Bereitstellung modularer, wiederverwendbarer Bausteine, mit denen sich neue Analysen nahezu beliebig zusammenstellen lassen bzw. bestehende Analysen an aktuelle Rahmenbedingungen angepasst werden können. Hierzu wird auf Konzepte der funktionalen Programmierung zurückgegriffen und mit den Möglichkeiten heutiger objektorientierter Sprachen kombiniert.

Der Nachweis dieser Eigenschaften erfolgt im Rahmen einer umfangreichen Evaluation anhand von Fallstudien, die die Vorteile gegenüber anderen Ansätzen aufzeigen und belegen.

Abstract

Software traceability is a well-established instrument in todays software development and is required by many development and quality standards. It displays the chronological and structural relationships among software artifacts, increases the system's transparency, and hence improves its understandability.

The tracability process comprises mainly of the steps strategy, creation, maintenance and analysis. The clear focus of available publications is the creation and maintenance of software traceabiliy information. It mostly covers only methods and techniques that ease or even automate the process of trace retrieval and trace recovery. Today, there is a plethora of tools available that promise ease within this matter.

However, the answers to traceability based questions are only given by the analysis of the available software traceability information. In practice, analyses are performed manually, by using appropriate tools, or by utilizing a programming language. As a matter of fact, the creation of repeatable and automatable traceability analysis is expensive. Analyses are in particular difficult to perform and maintain, if their data is spread across multiple different tools or if they have to be adjusted to continuously changing requirements. Hence, traceability is difficult to apply in an industrial environment, even if it provides a clear advantage. The high implementation effort also affects the development and research of new analysis approaches.

This PhD thesis addresses the problem of the difficult creation of traceability analyses. It investigates the characteristics of current approaches and derives the *Traceability Analysis Language* (*TracAL*) from the results. This domain-specific language eases the creation of traceability analyses and offers advantages compared to current approaches due to its fundamental properties:

- *Applicability:* The language provides the means and the functionality for creating repeatable and automatable traceability analyses. Its focus lies on the expressiveness and performance of the resulting analyses.
- *Representation Independence:* The approach leverages a graph-based model. It abstracts artifacts and links from their concrete representations within the different data sources that have to be dealt with during a software project. Due to this uniform representation, the traceability information of different tools and data sources can be used in an integrated way within a single analysis. *TracAL* provides a multi-level adapter concept that helps to connect to new data sources, that are currently unsupported.
- *Configurability: TracAL* allows for the creation of modular and re-usable code blocks that either can be combined to new analyses, or that can be used to adjust existing analyses for new requirements. The concept leverages ideas of the functional programming paradigm but combines it with the power of todays object-oriented languages.

These properties are verified by comprehensive case studies that show and prove the advantages of *TracAL* compared to other approaches.

Inhaltsverzeichnis

Kurzfassung i

Abstract ii

Abbildungsverzeichnis vii

Tabellenverzeichnis ix

1. Einführung 1
- 1.1. Problemstellung . . . 2
  - 1.1.1. Beispiel zur Umsetzung von Verfolgbarkeit . . . 2
  - 1.1.2. Ursachen für aufwändige Verfolgbarkeitsanalysen . . . 3
- 1.2. Einordnung und Beitrag . . . 4
- 1.3. Kapitelübersicht . . . 6

2. Grundlagen 7
- 2.1. Verfolgbarkeit . . . 7
  - 2.1.1. Begriffe . . . 7
  - 2.1.2. Verfolgbarkeitsprozess . . . 9
- 2.2. Modell . . . 11
  - 2.2.1. Modellbegriff . . . 11
  - 2.2.2. Metamodell . . . 13
  - 2.2.3. Graph . . . 13
- 2.3. Domänenspezifische Sprache . . . 15

3. Stand von Wissenschaft und Technik 17
- 3.1. Analysearten . . . 17
- 3.2. Bewertungskriterien . . . 20
  - 3.2.1. Anwendbarkeit . . . 20
  - 3.2.2. Repräsentationsunabhängigkeit . . . 22
  - 3.2.3. Konfigurierbarkeit . . . 23
- 3.3. Manuelle Ansätze . . . 24
  - 3.3.1. Visualisierungstechniken . . . 25
  - 3.3.2. Bewertungskriterien . . . 26
- 3.4. Werkzeugbasierte Ansätze . . . 28
  - 3.4.1. Allgemeine Werkzeuge . . . 28
  - 3.4.2. Domänenspezifische Werkzeuge . . . 32
  - 3.4.3. Domänenspezifische Arbeitsumgebungen . . . 34
- 3.5. Sprachbasierte Ansätze . . . 36
  - 3.5.1. Allgemeine Programmiersprachen . . . 36
  - 3.5.2. Allgemeine domänenspezifische Sprachen . . . 37
  - 3.5.3. Domänenspezifische Sprachen . . . 38
- 3.6. Zusammenfassung . . . 41

4. TracAL: Die Traceability Analysis Language **43**
4.1. Anforderungen . . . 43
4.1.1. Zielgruppe . . . 43
4.1.2. Einsatzszenarien . . . 44
4.1.3. Realisierungsaufwand . . . 45
4.2. Festlegung der Funktionalität . . . 45
4.2.1. Datenanbindung . . . 45
4.2.2. Anfrage . . . 45
4.2.3. Auswertung und Verarbeitung . . . 46
4.3. Auswahl der Ansatzkategorie . . . 47
4.3.1. Spracheigenschaften . . . 47
4.3.2. Modelleigenschaften . . . 51
4.4. Realisierung . . . 53
4.4.1. Funktionsbibliothek . . . 53
4.4.2. Syntaxerweiterungen . . . 56
4.4.3. Verteilung . . . 57
4.5. Arbeiten mit *TracAL* . . . 57
4.5.1. Datenanbindung . . . 58
4.5.2. Anfrage . . . 60
4.5.3. Auswertung und Verarbeitung . . . 61
4.5.4. Erweiterungen . . . 62
4.6. Verwandte Arbeiten . . . 64
4.6.1. Alternative Ansätze zur Umsetzung von Verfolgbarkeitsanalysen . . . 65
4.6.2. Abgrenzung zu den Arbeiten des Forschungspartners . . . 68
4.7. Zusammenfassung . . . 69

5. Repräsentationsunabhängigkeit **71**
5.1. Statisch typisierte Graphen . . . 71
5.1.1. Problembeschreibung und Lösungsansatz . . . 71
5.1.2. Erweiterung des Eigenschaften-Graphmodells . . . 74
5.1.3. Filterkonzept . . . 76
5.1.4. Generische und konkrete Typfilter . . . 78
5.1.5. Allgemeine Filter . . . 80
5.1.6. Statisch typisierte Graphklassen . . . 83
5.1.7. Polymorphie statisch typisierter Graphklassen . . . 87
5.1.8. Allgemeine Graphoperationen . . . 89
5.1.9. Graphschema . . . 91
5.1.10. Verwandte Arbeiten . . . 95
5.2. Adapterkonzept . . . 98
5.2.1. Allgemeiner Prozess zur Datenanbindung . . . 98
5.2.2. Problembeschreibung und Lösungsansatz . . . 100
5.2.3. Mehrstufiges Adapterkonzept . . . 101
5.2.4. Modulare Bausteine . . . 106
5.2.5. Verbesserte Repräsentationsunabhängigkeit . . . 110
5.2.6. Verwandte Arbeiten . . . 111
5.3. Zusammenfassung . . . 115

6. Konfigurierbarkeit **117**
6.1. Motivation . . . 117

6.2. Lösungsansatz . . . 119
6.2.1. Monaden . . . 120
6.2.2. Anwendungsbeispiel . . . 121
6.2.3. Einschränkungen . . . 123
6.2.4. Wissenschaftlicher Beitrag . . . 124
6.3. Klassen mit monadischen Effekten . . . 125
6.3.1. Datentyp . . . 125
6.3.2. Typklasse . . . 127
6.4. Effektstapel . . . 131
6.4.1. Aufbau . . . 132
6.4.2. Syntaxerweiterungen . . . 134
6.4.3. Umsetzung von Stapeloperationen . . . 139
6.4.4. Bearbeitungsreihenfolge . . . 142
6.5. Stapelbasierte Traversierschritte . . . 144
6.5.1. Allgemeine Traversierschritte . . . 144
6.5.2. Elementbasierte Traversierschritte . . . 147
6.5.3. Die Festlegung von Abarbeitungsregeln . . . 149
6.5.4. Das Einbringen von Effektberechnungen . . . 150
6.5.5. Wiederholungen . . . 153
6.6. Konfigurierbare Verfolgbarkeitsanalysen . . . 156
6.6.1. Modularität . . . 156
6.6.2. Erweiterbarkeit . . . 157
6.6.3. Anpassbarkeit . . . 157
6.7. Verwandte Arbeiten . . . 158
6.7.1. Monaden und Monadentransformatoren . . . 158
6.7.2. Funktionale Graphalgorithmen . . . 160
6.7.3. Alternative Umsetzung des Effektstapels . . . 160
6.7.4. Effektberechnungen in anderen graphbasierten Sprachen . . . 162
6.8. Zusammenfassung . . . 163

7. Evaluation **165**
7.1. Testkonfiguration . . . 166
7.1.1. Hardware . . . 166
7.1.2. Software . . . 166
7.2. Eigenschaften des Effektstapels . . . 167
7.2.1. Untersuchung des Laufzeitmehraufwands . . . 167
7.2.2. Untersuchung des Vorteils statischer Typisierung . . . 171
7.2.3. Zusammenfassung . . . 172
7.3. Fallstudie Zyklendetektion . . . 172
7.3.1. Aufbau und Umsetzung der Analyse in *TracAL* . . . 173
7.3.2. Vergleich mit anderen Ansätzen . . . 177
7.3.3. Laufzeitmessungen . . . 178
7.3.4. Erweiterung der Analyse . . . 183
7.3.5. Zusammenfassung . . . 184
7.4. Fallstudie Divergenzdetektion . . . 185
7.4.1. Aufbau und Umsetzung der Analyse in *TracAL* . . . 185
7.4.2. Vergleich mit anderen Ansätzen . . . 191
7.4.3. Vergleich der Ausdrucksstärke . . . 192
7.4.4. Laufzeitmessungen . . . 201
7.4.5. Erweiterung der Analyse . . . 203

7.4.6. Zusammenfassung 204
7.5. Fallstudie iTrust 204
7.5.1. Datenanbindung 206
7.5.2. Anfrage, Auswertung und Verarbeitung 217
7.5.3. Zusammenfassung 222
7.6. Zusammenfassung 223

8. Zusammenfassung und Ausblick **225**
8.1. TracAL: Die Traceability Analysis Language 225
8.1.1. Anwendbarkeit 225
8.1.2. Repräsentationsunabhängigkeit 226
8.1.3. Konfigurierbarkeit 227
8.1.4. Evaluation 228
8.2. Ausblick 229

A. Anhang **231**
A.1. UML-Klassendiagramme für Scala 231
A.1.1. Typparameter 232
A.1.2. Stereotypen 233
A.1.3. Methoden und Funktionen 234
A.2. Referenzimplementierung ausgewählter Effektberechnungen 235
A.2.1. Identitätsklasse 235
A.2.2. Pfadeffekt 238
A.2.3. Besuchseffekt 239

Liste der Veröffentlichungen **243**

Hinweis und Danksagung **244**

Quellenverzeichnis **245**