Dissertationen

Arnd Löbbert: Beurteilung der Eignung von Softwaresystemen für eine lange Lebensdauer

Promotion: Technische Universität Clausthal, Fakultät für Bergbau, Hüttenwesen und Maschinenwesen
Erstgutachter: Prof. Dr.-Ing. Peter F. Elzer
Zweitgutachter: Prof. Ph. D. Gerhard R. Joubert
Datum der Prüfung: 1. Februar 2002
Veröffentlichung: Papierflieger, Clausthal-Zellerfeld, 2002; ISBN 3-89720-608-0

Kurzfassung:
Es existieren zahlreiche Wirtschaftszweige, in denen Unternehmen ihre elektronisch gespeicherten Informationen aus wirtschaftlichen, wartungstechnischen und rechtlichen Gründen über einen langen Zeitraum nutzbar halten müssen. Dies setzt voraus, dass sich die Nutzungsphasen der verwendeten Dateitypen und Anwendungsprogramme überschneiden, denn ohne Anwendungsprogramm sind gespeicherte Informationen nicht existent. Die Problematik besteht nun darin, die Überschneidungen der beiden Nutzungsphasen trotz permanenter Umgebungsänderungen aufrecht zu erhalten. Hierzu müssen die Dateitypen und Anwendungsprogramme spezielle Qualitätsmerkmale besitzen, damit die Informationen nicht verloren gehen bzw. der Änderungsaufwand bei entsprechenden Anpassungen gering gehalten werden kann. In der bestehenden Literatur wird qualitätssichernden Maßnahmen zur Erhaltung von Softwaresystemen inkl. deren gespeicherten Informationen wenig Aufmerksamkeit geschenkt. Insbesondere Standardsoftware wurde in dieser Hinsicht bisher nicht untersucht. Es fehlt somit an geeigneten Qualitätsmodellen und maßen, die die einzuhaltenden Qualitätsmerkmale spezifizieren bzw. die Einhaltung dieser überprüfen und bewerten können. Die vorliegende Arbeit versucht einen Beitrag zur Schließung dieser Lücken zu leisten. Auf der Grundlage von modifizierten bzw. erweiterten Qualitätsmodellen wurde das neu entwickelte Qualitätsmaß abgeleitet. Mit diesem neuen Maß können Softwaresysteme auf ihre Fähigkeit hin überprüft werden, wie gut sie geeignet sind, Informationen über einen langen Zeitraum nutzbar zu halten. Es können Softwaresysteme bewertet werden, die sowohl aus Standardsoftwaremodulen oder eigenentwickelten Modulen als auch aus einer Kombination aus beiden aufgebaut sind. Zur Unterstützung des Benutzers bei der Werteermittlung wurde ein Werkzeug in Gestalt von Formblättern entwickelt. Um den unterschiedlichen Anforderungen der Wirtschaft in Bezug auf die Genauigkeit solcher Bewertungen Rechnung zu tragen, wurden zwei Varianten zur Bestimmung der Eignung von Softwaresystemen entwickelt. Zum einen ein Basisverfahren, das Softwaresysteme mit relativ wenig Aufwand grob beurteilen kann und zum anderen ein aufwendigeres detailliertes Verfahren, das Softwaresysteme genauer bewertet, indem feiner gegliederte Softwarekomponenten untersucht werden. Aufgrund der Transparenz der Verfahrensvarianten ist es möglich, Schwachstellen, die eine Einschränkung der Nutzbarkeit mit sich bringen, leicht zu identifizieren und zu lokalisieren sowie Verbesserungsmaßnahmen abzuleiten. Anhand der Einhaltung spezieller Kriterien, die ein Maß erfüllen muss, und den erwartungsgemäßen Ergebnissen bei der Durchführung einer Fallstudie, konnte das entwickelte Qualitätsmaß mit seinen beiden Varianten seine Tauglichkeit beweisen. Eine grobe Aufwandsschätzung zeigte, dass die aufzuwendende Zeit für die Durchführung des Verfahrens sehr vom Wissenstand des Prüfers abhängt. Deshalb sollten nur entsprechende Know-How-Träger die Prüfung durchführen, damit die Anwendung des Maßes nicht unwirtschaftlich und somit unpraktikabel wird.

Marc Monecke: Adaptierbare CASE-Werkzeuge in prozeßorientierten Software-Entwicklungsumgebungen

Promotion: Universität Siegen
Erstgutachter: Prof. Dr. Udo Kelter, Universität Siegen
Zweitgutachter: Prof. Dr. Volker Gruhn, Universität Leipzig
Datum der Prüfung: 7. Februar 2003
Veröffentlichung: http://www.ub.uni-siegen.de/epub/diss/monecke.htm

Kurzfassung:

Die Entwicklung von Software-Systemen ist eine komplexe und anspruchsvolle Aufgabe, die bei großen Systemen meist auf mehrere Entwickler verteilt wird. *Software-Entwicklungsmethoden* strukturieren die Arbeit der Beteiligten: Sie legen Arbeitsschritte und Produkte fest, die in den einzelnen Schritten erzeugt oder verarbeitet werden. Produkte sind hier verschiedene Arten von *Software-Dokumenten.* Die Arbeit konzentriert sich auf die frühen Software-Entwicklungsphasen Analyse und Entwurf, in denen Anforderungen ermittelt und Modelle wichtiger Aspekte des Systems erstellt werden. In diesen Phasen kommen *Upper-CASE-Werkzeuge* zum Einsatz, mit denen die verschiedenen Arten von Software-Dokumenten bearbeitet, geprüft und in andere Formate transformiert werden können. Die Werkzeuge müssen also die im jeweiligen Dokument verwendeten Konzepte (wie Entitätstyp, Klasse, Vererbungsbeziehung) und die zugehörigen Notationen und Darstellungen unterstützen.

Die Anforderungen an CASE-Werkzeuge unterscheiden sich stark, da sie in unterschiedlichen Organisationen, von Personen mit unterschiedlichen Aufgaben und Zielen und zur Entwicklung unterschiedlicher Arten von Systemen eingesetzt werden – zum Teil sind die Anforderungen auch unbekannt, wenn in einem Projekt neue Wege beschritten werden. Eine wichtige Anforderung an die Werkzeuge ist daher, daß sie an die jeweilige Einsatzsituation angepaßt werden können. Herkömmliche CASE-Werkzeuge erfüllen diese Anforderung nur unzureichend. *Meta-Umgebungen* erleichtern den Bau angepaßter Werkzeuge, allerdings konzentrieren sich die meisten bekannten Ansätze auf nur einen bestimmten Aspekt: Den Bau von CASE-Werkzeugen für einen gegebenen Dokumenttyp oder die Unterstützung des gegebenen Vorgehens- oder *Prozeßmodells.*

In der Arbeit wird ein Ansatz zum Bau von *prozeßorientierten Software-Entwicklungsumgebungen* entwickelt, mit dem CASE-Werkzeuge maßgeschneidert *und* mit dem Software-Entwicklungsprozeß integriert werden können. Die resultierenden Umgebungen nutzen ein *Object Management System* (OMS) als zentrales Repository. Wichtige Merkmale sind die feingranulare Modellierung der Dokumente und eine Werkzeugarchitektur, mit der die Dienste des OMS zur Implementierung der Werkzeugfunktionen direkt ausgenutzt werden können. In der Arbeit wird ein Framework mit Werkzeugkomponenten entwickelt, das zum Bau mehrbenutzerfähiger und verteilt einsetzbarer Werkzeuge verwendet wird. Werkzeuge können einfach und schnell aus vorhandenen Komponenten zusammengesetzt und Parameter zur Steuerung dieser Komponenten festgelegt werden. Weiterhin können Komponenten erweitert und angepaßt werden, wodurch eine hohe Flexibilität bei der Adaptierung von Werkzeugen erreicht wird.

Zur Beschreibung des Prozeßmodells wird eine einfache Prozeßmodellierungssprache vorgeschlagen. Mit Framework-Komponenten kann eine Prozeßmaschine zur Steuerung und Überwachung des Prozesses gebaut und flexibel um prozeßspezifische Funktionen erweitert werden. Das Prozeßmodell dient hier als Spezifikation für die Prozeßmaschine. Die gebauten CASE-Werkzeuge werden mit dem Prozeß integriert, passen sich also an die aktuelle Prozeßsituation an und ermöglichen es dem Entwickler, den Prozeßfortschritt zu beeinflussen. Mit dem Bau von Planungs- und Managementwerkzeugen und ihrer Integration in die Umgebung wird die Flexibilität des Ansatzes geprüft.

Jörg Zettel: Anpassbare Methodenassistenz in CASE-Werkzeugen

Promotion: Universität Kaiserslautern
Erstgutachter: Prof. Dr. H. Dieter Rombach, Universität Kaiserslautern
Zweitgutachter: Prof. Dr. Michael M. Richter, Universität Kaiserslautern
Datum der Prüfung: 28.2.2003
Veröffentlichung: Stuttgart: Fraunhofer IRB Verlag, 2003 (PhD Theses in Experimental Software Engineering 13). Zugl.: Kaiserslautern, Univ., Diss., 2003; ISBN 3-8167-6284-0

Kurzfassung:

Heutige Softwaresysteme und Softwareentwicklungsprozesse sind derart komplex, dass sie ohne methodische Vorgehensweisen und technische Hilfsmittel nicht mehr zu beherrschen sind. Professionelle Softwareentwicklung ist daher ohne den Einsatz von „computer aided software engineering“ (CASE) nicht denkbar. Mit CASE-Werkzeugen sollen darüber hinaus entscheidende Produktivitätssteigerungen und Qualitätsverbesserungen erzielt werden können. Dies gilt insbesondere für CASE-Werkzeuge, die Unterstützung für solche Aktivitäten der Softwareentwicklung, des Softwaremanagements und des Softwarequalitätsmanagements bereitstellen, bei denen die graphische Modellierung im Vordergrund steht. Diese CASE-Werkzeuge basieren auf anerkannten Methoden der Softwaretechnik und assistieren ihren Benutzern bei der Anwendung dieser Methoden.

Dem intensiven Einsatz von CASE-Werkzeugen steht allerdings ihre geringe Akzeptanz seitens der Benutzer im Wege. In der Mehrzahl aller Fälle werden CASE-Werkzeuge zwar angeschafft, aber kaum eingesetzt. Eine wesentliche Ursache dafür ist, dass die von heutigen CASE-Werkzeugen angebotene Methodenassistenz nicht an die Bedürfnisse der Benutzer angepasst ist. Insbesondere die von dem CASE-Werkzeug angebotene automatisierte Konsistenzsicherung kann nicht an die Notwendigkeiten der jeweiligen Aufgabe oder an persönliche Vorlieben des Benutzers angepasst werden.

In dieser Dissertation wird eine Technik vorgestellt, die die Entwicklung anpassbarer Methodenassistenz ermöglicht. Sie ist auf Methoden anwendbar, bei denen die Erstellung vorwiegend graphischer Modelle im Vordergrund steht. Damit werden Anbieter von CASE-Werkzeugen in die Lage versetzt, ihre Produkte mit anpassbarer automatisierter Konsistenzsicherung von Modellen auszustatten. Durch die Möglichkeit zur Anpassung an die Bedürfnisse des Benutzers und die Erfordernisse der jeweiligen Aufgabe soll die Akzeptanz von CASE-Werkzeugen verbessert werden. Die Technik umfasst: 1) ein Verfahren zur Spezifikation des Verhaltens automatisierter Konsistenzsicherung von Modellen durch systematische Herleitung aus dem Metamodell und den Konsistenzregeln der Methode, 2) ein Entwurfsmuster zur Konstruktion anpassbarer automatisierter Konsistenzsicherung und 3) Empfehlungen zur Gestaltung der Mensch-Maschine-Schnittstelle eines derartigen CASE-Werkzeuges.

In einer Fallstudie wurde die Technik auf die deskriptive Modellierung von Softwareentwicklungsprozessen und das CASE-Werkzeug SPEARMINT angewendet. Die Entwicklung anpassbarer Methodenassistenz stellte sich dabei als machbar heraus. In einem kontrollierten Experiment wurde daraufhin gezeigt, dass die angebotene Methodenassistenz ein Einflussfaktor für die Akzeptanz eines CASE-Werkzeuges ist. Eine daran anschließende Umfrage unter erfahrenen Benutzern von SPEARMINT ergab, dass die bevorzugte Form automatisierter Konsistenzsicherung u. a. von der Art der Aufgabe abhängt.

Insgesamt gesehen stellt die in dieser Dissertation beschriebene Technik zur Entwicklung anpassbarer Methodenassistenz ein für Anbieter von CASE-Werkzeugen interessantes Instrument dar, mit dem sie die Akzeptanz und damit auch den Absatz ihrer Produkte steigern können.

Dirk Beyer: **Formale Verifikation von Realzeit-Systemen mittels Cottbus Timed Automata**

Promotion: Brandenburgische Technische Universität Cottbus
Erstgutachter: Prof. Dr. Claus Lewerentz, Brandenburgische TU Cottbus
Zweitgutachter: Prof. Dr.-Ing. Monika Heiner, Brandenburgische TU Cottbus
Drittgutachter: Prof. Dr. Werner Damm, Universität Oldenburg
Datum der Prüfung: 26. November 2002
Veröffentlichung: Berlin: Verlag Mensch & Buch, 2002. Zugl.: http://www.ub.tu-cottbus.de/hss/diss/fak1/beyer_d; Cottbus, Univ., Diss., 2002. ISBN: 3-89820-450-2

Kurzfassung:
Over the last decade the formalism of timed automata has become more and more important in the field of modeling and verification of real-time systems. However, its application leads to the following main problems which were solved in the thesis:

- Large systems can be modeled only in a complicated and unstructured manner. The model consists of a collection of automata which communicate only on one level. It is not possible to model recurrent parts of the system in a reusable way, they have to be defined as often as needed.

- The algorithmic verification is possible only for small models because of enormous consumption of time and memory (state-space explosion).

The modeling formalism **Cottbus Timed Automata** (CTA), which is introduced in the thesis, extends the existing concepts of timed automata by means of **modularity**. The behavior of a CTA model is defined by mapping it onto timed automata. Because of the ability of hierarchical structuring with modules and the mechanism of instantiation for recurrent use of previously defined modules even large systems can be modeled.

For the verification of CTA models an **efficient BDD-based reachability analysis** is introduced which outperforms the existing verification tools for timed automata. One key to this improvement is to use the modular structure of the model for the computation of good BDD variable orderings. Additionally, **refinement checking** is defined and implemented, which is also based on the efficient BDD representation. This makes it possible to verify large models which are intractable for pure model checking.

Timed automata are not sufficiently expressive to model all aspects of a **hybrid system**. Therefore the CTA formalism is extended from timed to hybrid automata. To integrate the verification of hybrid models into the tool, a representation of the model on the basis of the double description method, which is a matrix-based data structure for convex polyhedra, is implemented.

The theoretical concepts introduced in the first part of the thesis were implemented in the tool **Rabbit**. The architecture principle is flexibility: not only parts of the tool can be easily and flexibly extended or changed, but also the whole data structure for the representation of the model can be chosen at run-time. Therefore, the tool can serve as verification framework for various (future) data structures for representation, when new ideas about data structures and algorithms have to be empirically evaluated. For a lot of example models it has been shown that the tool Rabbit is more efficient than the existing tools.

Finally, the thesis reports results of a **case study for modeling and verification** of a large, realistic system: a production cell with 44 sensors and 28 motors. The procedure of modeling is explained and the hierarchical structure is sketched out. Properties of the system are specified and verified. With the help of modular proof principles, properties of the system were verified not only for small parts of the model, but also carried over to the whole system.

Conclusion: Contradicting the occasionally expressed opinion that the technology of model checking has reached its limitations, this thesis has shown that models of large size can be verified. The key idea is the use of **structure**: a modeling formalism is introduced to provide the expressiveness to reflect the explicit structure of components and its communication in the model in a quite natural way. The structural modeling allows the construction of models which are understandable, easy to use and easy to adopt. Furthermore, this structure is used to improve the performance of the verification task. The BDD implementation reduces the size complexity of the representation of configuration sets by using a structure-oriented variable ordering. This allows efficient reachability analysis of large models. The refinement analysis supports the application of compositional proof techniques. The modular structure of the model does lead to refinement structures which are not artificially created, but they are compatible to the natural structure of the system and therefore they can be used as abstractions in modular proofs.