Identifikation der Potenziale dreidimensionaler Softwarevisualisierung - Zielstellung und Vorgehen

Richard Müller
Institut für Wirtschaftsinformatik
Universität Leipzig
rmueller@wifa.uni-leipzig.de

Abstract: Die Softwarevisualisierung ist ein wesentlicher Bestandteil der ingenieurmäßigen Softwareentwicklung. Sie macht statische, dynamische und evolutionäre Aspekte von Softwareartefakten in den verschiedenen Phasen des Softwarelebenszyklus sichtbar. Mithilfe von aufgaben- und rollenspezifischen Sichten kann die Software schneller verstanden und ihr Entwicklungsprozess besser gesteuert und kontrolliert werden. Das zentrale Ziel der Arbeit liegt in der Optimierung des Softwareentwicklungsprozesses, indem Nutzenpotenziale der dreidimensionalen Softwarevisualisierung für einzelne Aufgaben dieses Prozesses identifiziert werden. Der tatsächliche Nutzen dieser Potenziale soll dann theoretisch und empirisch überprüft werden und neue Erkenntnisse für Wissenschaft und Praxis liefern.

1 Motivation und Problemstellung

Während des Softwareentwicklungsprozesses werden unterschiedliche Artefakte von verschiedenen Interessensbeteiligten erstellt. Hierzu zählen zum Beispiel von Kunden geäußerte Anforderungen, von Softwarearchitekten erstellte Modelle, von Programmieren geschriebener Quelltext und von Testdurchläufen erzeugte Protokolle. Die Softwarevisualisierung wird eingesetzt, um aufgaben- und rollenspezifische Sichten auf diese Artefakte zu erzeugen, die den Fähigkeiten und Informationsbedürfnissen der betreffenden Interessensbeteiligten entsprechen. Damit eröffnet die Softwarevisualisierung die Möglichkeit, der an sich körperlosen Software ein körperliches Modell zu geben, um dem menschlichen Benutzer die Wahrnehmung, das Verständnis und das Behalten zu erleichtern. Ein korrektes und nachhaltiges Verständnis komplexer Softwaresysteme bildet die Voraussetzung für ein effektives und effizientes Softwareprojektmanagement.

Bisher werden beispielsweise unterschiedliche Arten von UML-Diagrammen (engl. *unified modeling language*) eingesetzt, um Struktur- oder Verhaltensaspekte von Software zweidimensional zu visualisieren. Die einzelnen Diagramme stehen zwar inhaltlich miteinander in Beziehung, doch diese Beziehung selbst ist nicht Bestandteil der Visualisierungen.

Um die Visualisierung eines Softwareartefakts einschließlich der Beziehungen zu anderen Softwareartefakten effizient und ausdrucksstark zu gestalten, bedarf es einer körperlichen

Metapher. Damit werden räumlich sinnvolle Anordnungen der Visualisierungselemente eingegrenzt. Die Metapher kann dann zweidimensional (2D), zweieinhalbdimensional (2,5D) und dreidimensional (3D) dargestellt werden. Dabei ist davon auszugehen, dass entlang der 2D-, der 2,5D- und der 3D-Visualisierung die körperlichen Aspekte immer deutlicher und präziser hervortreten und in virtuellen Umgebungen (engl. *virtual reality environments*) schließlich einer relativ direkten Manipulation zugänglich werden. Dies unterstützt wiederum ihre interaktive Exploration, die dem Menschen im Falle der Interaktion des eigenen Körpers mit anderen Körpern besonders leicht fällt. Demnach ist es erst auf diese Art und Weise möglich, Körper in Raum und Zeit angemessen darzustellen und so eine ganzheitliche Sicht auf Software und ihren Entwicklungsprozess zu vermitteln.

Im Softwarelebenszyklus und in der Softwareentwicklung erwarten die Anwender von Softwarevisualisierungswerkzeugen folgenden projekt- und softwarespezifischen Nutzen [BK01].

- Handhabung der Komplexität von Software
- Erlangung eines besseren Verständnisses von Software
- Kosten- und Zeiteinsparungen bei der Softwareentwicklung und -wartung
- Vereinfachung der Fehlersuche in Software
- Produktivitäts- und Qualitätssteigerungen in der Softwareentwicklung und -wartung

In der Arbeit wird davon ausgegangen, dass sich dieser potenzielle Nutzen der Softwarevisualisierung auf die 3D-Softwarevisualisierung übertragen und weiter steigern lässt. Dazu soll erforscht werden, wie der Softwareentwicklungsprozess durch die 3D-Softwarevisualisierung unterstützt und über die konventionellen Visualisierungsmethoden hinausgehend optimiert werden kann. Der tatsächliche Nutzen der 3D-Softwarevisualisierung im Vergleich zur 2D-Softwarevisualisierung soll dann theoretisch und empirisch untersucht werden. Im Folgenden schließt der Begriff 3D-Softwarevisualisierung auch virtuelle Umgebungen ein.

Primär ist die Arbeit der Softwaretechnik zuzuordnen. Gleichzeitig besitzt sie Bezüge zur Wirtschaftsinformatik, da es darum geht, den Entwicklungsprozess von Informations- und Kommunikationssystemen zu verbessern und in diesem Zusammenhang nicht ausgeschöpfte Nutzenpotenziale aufzuzeigen [FSV05, S. 6 ff.].

2 Stand der Forschung

In der Literatur finden sich verschiedene Definitionen der Softwarevisualisierung. Diese Arbeit folgt der verhältnismäßig weit gefassten Definition von Diehl [Die07, 3 f.], die neben der Visualisierung von Struktur und Verhalten auch die Evolution von Softwareartefakten einbezieht. Softwareartefakte sind alle Zwischen- oder Endergebnisse, die während des Softwareentwicklungsprozesses entstehen. Die Softwarevisualisierung ist auch ein Teilgebiet der Informationsvisualisierung. Dadurch ist es möglich, etablierte Konzepte der Informationsvisualisierung zu adaptieren. Außerdem tangiert sie die Gebiete Mensch-Maschine-Interaktion, kognitive Psychologie und Computergrafik [MCS05].

Eine jede Visualisierung wird in einer Visualisierungspipeline erstellt. Diese spezifiziert eine Prozesskette zur Erzeugung graphischer Repräsentationen [dSB04]. Um eine Darstellung zu erzeugen, müssen Informationen extrahiert, analysiert, gefiltert, auf eine graphische Repräsentation abgebildet und schließlich gerendert werden. Dies gilt gleichermaßen für die Softwarevisualisierung.

Einen effizienten Ansatz, um diesen Visualisierungsprozess umzusetzen, verfolgen Bull et al. mit Model Driven Visualization (MDV) [BSFL06]. MDV ist eine Vorgehensweise zur Erzeugung von Visualisierungen mittels Modellen und Transformationen. MDV baut konzeptionell auf Model Driven Software Development (MDSD) [SVEH07] und Model Driven Architecture (MDA) [Obj03] auf. In [Mül09] wurde dieser Ansatz adaptiert und das modellgetriebene Paradigma mit dem generativen Paradigma (engl. *generative software engineering*, GSE) [CE00] kombiniert, um einen Softwarevisualisierungsgenerator zu entwickeln. Die Erkenntnisse und Ergebnisse aus der vorangegangenen Entwicklung bilden die Basis für diese Arbeit. Mithilfe dieses allgemeinen Visualisierungsprozesses lassen sich sowohl 2D-, 2,5D- und 3D-Visualisierungen erzeugen.

Die Vor- und Nachteile der 2D- bzw. 3D-Visualisierung werden kontrovers diskutiert. Einen kompakten Überblick des anhaltenden wissenschaftlichen Diskurs geben [TC09].

Bei der 3D-Visualisierung werden der höhere Rechen- und Entwicklungsaufwand kritisiert. Der Rechenaufwand spielt aber durch die steigende Rechenleistung eine immer geringere Rolle. Um den Entwicklungsaufwand zu minimieren, wurden unter anderen Modellierungssprachen wie die Virtual Reality Modeling Language (VRML) beziehungsweise deren Nachfolger Extensible 3D (X3D) entwickelt. Eine weitere Schwierigkeit liegt in der Heranführung der Benutzer an neue 3D-Metaphern und Interaktionstechniken. Deshalb sollte bei deren Gestaltung auf Intuitivität und auf Gebrauchstauglichkeit (engl. *usability*) geachtet werden.

Der Mensch lebt und agiert in einer dreidimensionalen Welt. Bei 3D-Darstellungen werden seine visuellen Fähigkeiten ausgenutzt. Ähnelt die 3D-Darstellung darüber hinaus noch der Realität, wird das Verständnis zusätzlich gefördert. Außerdem bietet die 3D-Darstellung einen größeren Raum für Informationen und komplexe Beziehungen können schneller erfasst und verstanden werden. Insbesondere durch die stetige Komplexitätszunahme von Software spielen diese Aspekte eine wichtige Rolle.

Verschiedene Forscher setzen VRML und X3D zur statischen und dynamischen 3D-Softwarevisualisierung ein [ANMB08]. Betrachtet man die Werkzeuge zur 3D-Softwarevisualisierung genauer, lassen sich verschiedene Herausforderungen identifizieren, die bisher nicht hinreichend bewältigt werden konnten [SP91, BK01, TC09].

- Entkopplung der Visualisierungswerkzeuge vom Entwicklungsprozess beziehungsweise der -umgebung
- Dominanz proprietärer Visualisierungsformate
- Geringer Automatisierungsgrad des Visualisierungsprozesses
- Unzureichende Ausnutzung neuer Darstellungs- und Interaktionstechniken
- Fehlende oder unzureichende Untersuchung der Gebrauchstauglichkeit

Neben diesen konkreten Schwächen existierender Werkzeuge zur 3D-Softwarevisualisierung fehlen vor allem auf diesem Gebiet empirische Untersuchungen [TC09]. Aber gerade empirische Untersuchungen sind in der Softwaretechnik und der Softwarevisualisierung erforderlich [HT07, PT07]. Demzufolge müssen bei der Entwicklung von Visualisierungswerkzeugen sowohl die visuelle Repräsentation als auch die dazugehörigen Interaktionstechniken im Kontext einer spezifischen Aufgabe evaluiert werden.

3 Zielstellung und Forschungsfrage

Das Ziel der Arbeit liegt in der Optimierung des Softwareentwicklungsprozesses, indem nicht ausgeschöpfte Nutzenpotenziale der 3D-Softwarevisualisierung für spezifische Aufgaben dieses Prozesses identifiziert werden. Der tatsächliche Nutzen dieser Potenziale soll dann theoretisch und empirisch überprüft werden. Damit wird ein konkreter Forschungsbeitrag zum aktuellen wissenschaftlichen Diskurs 2D versus 3D geliefert. Darüber hinaus sollen die Erkenntnisse und Ergebnisse ebenfalls einen konkreten Mehrwert für die Praxis liefern. Um dieses Ziel erreichen zu können, müssen zwei weitere Teilziele definiert werden.

Einerseits bedarf es eines theoretischen Modells zur Softwarevisualisierung, das den Untersuchungsbereich beschreiben, erklären und prognostizieren kann. Aus diesem Modell werden dann die Hypothesen zur Identifikation der Potenziale abgeleitet.

Andererseits müssen für die empirischen Untersuchungen 2D- und 3D-Visualisierungen einfach und effizient erzeugt werden können. Es wurde bereits dargestellt, dass die Werkzeugunterstützung im Bereich der 3D-Softwarevisualisierung Schwächen aufweist. Deshalb wird ein Softwarevisualisierungsgenerator konzipiert und als Eclipse-Plugin implementiert. Der zu entwickelnde Generator wird in der Lage sein, vollständig automatisiert X3D-basierte 2D- und 3D-Visualisierungen aus Softwareartefakten gemäß Benutzeranforderungen zu erzeugen. Die Grundlage des Generators bildet ein Metamodell zur Softwarevisualisierung. Dieses beschreibt die formale Struktur von Visualisierungen, die mit dem Generator erzeugt werden.

Die Forschungsfrage der Arbeit leitet sich aus dem zentralen Ziel ab und wird durch drei weitere Teilfragen konkretisiert.

- Forschungsfrage: Wo liegen die Potenziale der 3D-Softwarevisualisierung während des Softwareentwicklungsprozesses?
    1. Teilfrage: Welche Aufgaben im Rahmen des Softwareentwicklungsprozesses lassen sich durch die 3D-Visualisierung unterstützen?
    2. Teilfrage: Welche 3D-Visualisierungstechniken eignen sich für welche Aufgaben?
    3. Teilfrage: Wie lässt sich der tatsächliche Nutzen der 3D-Softwarevisualisierung im Vergleich zur 2D-Softwarevisualisierung quantifizieren?

4 Methodisches Vorgehen

Der allgemeine Forschungsrahmen der Arbeit folgt dem konstruktionsorientierten Paradigma [HMPR04]. Im Mittelpunkt stehen die Konstruktion und Evaluation neuer und innovativer Artefakte, die auf die Lösung wichtiger und relevanter Probleme ausgerichtet sind. Das theoretische Modell zur Softwarevisualisierung und der Softwarevisualisierungsgenerator sind solche Artefakte.

Um die zentrale Forschungsfrage zu beantworten, wird dem in Abbildung 1 dargestellten Forschungsprozess gefolgt [Kor07, S. 195, modifiziert]. Dabei werden gezielt Forschungsmethoden der Softwaretechnik und der Wirtschaftsinformatik eingesetzt [PT07, WH07]. Zunächst wird in den relevanten Gebieten nach theoretischen und empirischen Erkenntnissen gesucht, die in Bezug zur Forschungsfrage stehen. Das gesammelte Material wird anschließend metaanalytisch zusammengefasst und ausgewertet. Im nächsten Schritt werden aus dem aggregierten Material Aussagen abgeleitet, um damit das theoretische Modell zur Softwarevisualisierung zu entwerfen. Parallel dazu wird der Softwarevisualisierungsgenerator konzipiert und prototypisch implementiert. Anschließend werden aus dem theoretischen Modell Hypothesen deduktiv und abduktiv abgeleitet. Die Hypothesen werden mit den generierten 2D- und 3D-Visualisierungen in Laborexperimenten empirisch untersucht und überprüft. Hierfür werden etablierte Methoden des Usability-Engineerings (UE) und Evaluationskriterien aus den Taxonomien der Informations- und Softwarevisualisierung miteinander kombiniert. Die Ergebnisse der Hypothesentests bilden dann die Basis zur Klärung der Forschungsfrage respektive zur Identifikation der Potenziale der 3D-Softwarevisualisierung.

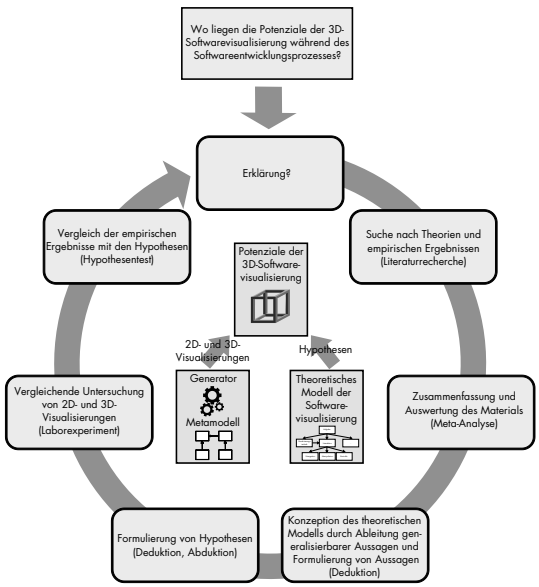

Abbildung 1: Forschungsprozess

Literatur

[ANMB08] C. Anslow, J. Noble, S. Marshall und R. Biddle. Evaluating Extensible 3D Graphics For Use in Software Visualization. In *Symposium on Visual Languages and Human-Centric Computing (VLHCC)*, 2008.

[BK01] S. Bassil und R. K. Keller. Software Visualization Tools: Survey and Analysis. In *Proceedings of the 9th International Workshop on Program Comprehension*, Seiten 7–17, 2001.

[BSFL06] R. I. Bull, M.-A. Storey, J.-M. Favre und M. Litoiu. An Architecture to Support Model Driven Software Visualization. In *International Conference on Program Comprehension*, Seiten 100–106, 2006.

[CE00] K. Czarnecki und U. W. Eisenecker. *Generative Programming Methods, Tools and Applications*. Addison-Wesley, 2000.

[Die07] S. Diehl. *Software Visualization - Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, 2007.

[dSB04] S. d. Santos und K. Brodlie. Gaining understanding of multivariate and multidimensional data through visualization. *Computers & Graphics*, 28(3):311–325, Juni 2004.

[FSV05] A. Fink, G. Schneidereit und S. Voß. *Grundlagen der Wirtschaftsinformatik*. Birkhäuser, Mai 2005.

[HMPR04] A. R. Hevner, S. T. March, J. Park und S. Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105, 2004.

[HT07] A. Höfer und W. F. Tichy. Status of Empirical Research in Software Engineering. In *Empirical Software Engineering Issues*, Jgg. 4336/2007, Seiten 10–19. Springer, 2007.

[Kor07] M. Kornmeier. *Wissenschaftstheorie und wissenschaftliches arbeiten: Eine Einführung für Wirtschaftswissenschaftler*. Springer, Dezember 2007.

[MCS05] A. Marcus, D. Comorski und A. Sergeyev. Supporting the Evolution of a Software Visualization Tool Through Usability Studies. In *IWPC '05: Proceedings of the 13th International Workshop on Program Comprehension*, Seiten 307–316. IEEE Computer Society, 2005.

[Mül09] R. Müller. *Konzeption und prototypische Implementierung eines Generators zur Softwarevisualisierung in 3D*. Number 4 in Forschungsberichte des Instituts für Wirtschaftsinformatik. Leipzig, November 2009. ISSN: 1865-3189.

[Obj03] Object Management Group. *MDA Guide Version 1.0.1*. Juni 2003.

[PT07] F. Padberg und W. F. Tichy. Empirische Methodik in der Softwaretechnik im Allgemeinen und bei der Software-Visualisierung im Besonderen. In Gesellschaft für Informatik, Hrsg., *Software Engineering 2007 - Beiträge zu den Workshops*, LNI, Seiten 211–222, 2007.

[SP91] J. T. Stasko und C. Patterson. Understanding and characterizing program visualization systems. Bericht, Oktober 1991.

[SVEH07] T. Stahl, M. Völter, S. Efftinge und A. Haase. *Modellgetriebene Softwareentwicklung - Techniken, Engineering, Management*. dpunkt.verlag, 2007.

[TC09] A. R. Teyseyre und M. R. Campo. An Overview of 3D Software Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):87–105, 2009.

[WH07] T. Wilde und T. Hess. Forschungsmethoden der Wirtschaftsinformatik. *WIRTSCHAFTSINFORMATIK*, 49(4):280–287, 2007.