3D „Fly-Through“ Darstellung von komplexen Software-Strukturen mit und ohne Virtueller Realität

Carsten Lecon und Roy Oberhauser, Hochschule Aalen

Je komplexer eine Software wird, desto schwieriger wird es, dessen Strukturen zu durchschauen. Herkömmliche Darstellungstechniken wie die Unified Modeling Language stoßen dabei schnell an ihre Grenzen. Wir stellen einen Ansatz vor, in dem Software-Strukturen mittels dreidimensionalen Welten erkundbar werden- sowohl auf dem Bildschirm (FlyThruCode), als auch in der Virtuellen Realität (VR-FlyThruCode).

Alternative Darstellung von Software-Strukturen

In der Informatik ist Software Engineering ein essentieller Bestandteil der Ausbildung an Hochschulen. Das professionelle Entwickeln von Software geht weit über das reine Programmieren heraus: Es geht auch darum, komplexe Software-Architekturen zu entwickeln, bei denen es viele Beziehungen zwischen den einzelnen Komponenten (z.B. Klassen, Pakete) gibt. Nach F. P. Brooks, Jr. bleibt die Unsichtbarkeit von Software eine essenzielle Schwierigkeit bei dessen Konstruktion, weil die Realität von Software nicht in Raum eingebettet ist [1]. Je komplexer die Software-Systeme werden, desto schwieriger wird es, eine adäquate und übersichtliche Darstellung zu finden. Die gängige Unified Modeling Language (UML) wird bei sehr komplexen Software-Strukturen schnell unübersichtlich. Unsere Idee war es deshalb, die zweidimensionale Darstellung von UML um eine dritte Dimension zu erweitern und mehrere visuelle Metaphern anzubieten. Als visuelle Metapher für Klassen und Pakete haben wir uns für zwei Varianten entschieden: eine Terrestrische- und eine Weltraumansicht. Die Vision von 3D Darstellungen von Software-Architekturen besteht schon länger [2], und einige Ansätze hierzu gibt es bereits [3].

Lösungsansatz

Um dieses Vorhaben umzusetzen, wurde von Prof. Oberhauser (Professor für Software Engineering) eine Arbeitsgruppe zusammen mit Prof. Dr. Carsten Lecon (Professor für Medieninformatik) gebildet. Für den FlyThruCode 3D-Prototypen diente eine Spiel bzw. Game Engine (Unity 3D) als Grundlage: Ein Software Framework speziell für Computerspiele, das das Spielverhalten und die visuelle 3D Spieldarstellung unterstützt. Zudem bietet die Engine eine flexible Programmierschnittstelle, sodass sämtliche Funktionalitäten leichter in einer höheren Programmiersprache – in unserem Fall C#-programmiert werden können. Da Unity 3D plattformunabhängig ist, lässt sich die Anwendung auf unterschiedlichen Geräten und unterschiedlichen Betriebssystemen nutzen: Windows PC, Unix, Linux, Mac, Tablets, Smartphones.

Mehrere Code Analyse Werkzeuge wurden eingesetzt, um Informationen wie Programm- Metriken (zum Beispiel die Anzahl von Code-Zeilen in einer Datei) und Abhängigkeiten von Programmteilen zu extrahieren, und dessen Ausgaben dann in ein einheitliches Format transformiert wurde.

Die einzelnen Prozessschritte der Anwendung sind:

- Modellierung von Programm- und Projektstrukturen sowie von visuellen Objekten
- Extraktion der relevanten Programmstrukturen und -metriken durch entsprechende Tools
- Mapping der konkreten Projektstrukturen und -metriken auf konkrete visuelle Metapher
- Visualisierung eines Software-Modells in eine konkrete visuelle Metapher
- Bereitstellung von Navigationsmöglichkeiten durch den dreidimensionalen Raum

Visuelle Metapher

Im Terrestrischen Modell werden Klassen als Häuser abgebildet, deren Höhe eine Eigenschaft, wie die Größe (z.B. Anzahl der Methoden) der entsprechenden Klassen, widerspiegelt. Eine Gruppierung in einer „Stadt“ entspricht einem Paket (package). Assoziationen zwischen den Klassen werden durch gefärbte Röhre (3D-Zylinder) dargestellt (Abb. 1). Im Weltraum Modell werden Klassen durch Planeten und Pakete als Sonnensysteme illustriert. Auch hier werden Assoziationen mittels Röhren visualisiert (Abb. 2).

Prof. Dr. C. Lecon

Prof. R. Oberhauser

Vielfältige Funktionalitäten

Optional kann während der Anwendung ein Grundgeräusche abgespielt

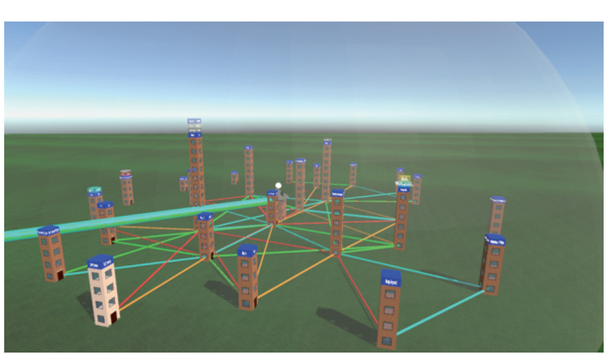
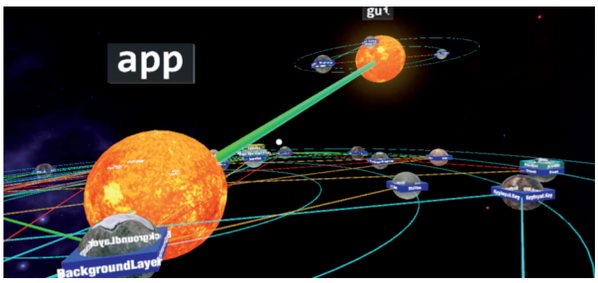


Abb.1: Software-Strukturen, dargestellt mit der City-Metapher in einer Glaskugel



Software-Strukturen, dargestellt mit der Weltraum-Metapher

werden, sodass Umgebungsgeräusche weniger ablenkend wirken: im Weltraum wird Alphawellenmusik (was die Konzentration unterstützt) verwendet, und in der City-Darstellung sind Vogelstimmen zu hören.

Vielfältige Funktionalitäten

Im Gegensatz zu bereits existierenden Ansätzen zu dreidimensionalen Präsentationsformen von Software werden in unserem Prototyp mehrere 3D-Metaphern verwendet, zwischen denen übergangslos gewechselt werden kann. In Analogie zu einem Flugzeug- (Raumschiff-) Cockpit ist die Anzeige und Steuerkonsole gestaltet. Es finden sich beispielsweise Informationen zum aktuellen Projekt oder ein Eingabefeld für die Suchfunktion, mittels derer Klassen oder Methoden direkt gesucht und visuell angesteuert werden können. Zur Anzeige dient ein sogenanntes, frei auf dem Bildschirm verschiebbares Heads-Up-Display (HUD) – ein transparentes Fenster mit weiteren Informationen. Es existieren verschiedene HUD-Anzeigen, z. B.:

- Source Code: Aus Übersichtlichkeitsgründen bestehen die visuellen Repräsentanten für Klassen und Pakete nur aus der geometrischen Form, beschriftet mit dem Namen der Klassen beziehungsweise Paket. Dennoch ist es möglich, den kompletten Source Code einer Klasse im HUD anzeigen zu lassen. Hierzu wird eine virtuelle Leinwand mit dem Source Code eingeblendet – mit Scroll-Möglichkeit.
- UML: Neben der dreidimensionalen Repräsentation ist es auch möglich, UML-Diagramme darzustellen. Diese werden dynamisch generiert (in 2D).
- Tags: Klassen und Pakete lassen sich mit Annotationen versehen. Dies können automatisch erfasste Daten basierend auf Mustern sein oder von den Anwender/innen explizit gesetzte Werte. Tags haben zwei Bedeutungen: Sie dienen der erweiterten Beschreibung. Zudem können sie für Suche genutzt werden. Die manuelle Eingabe der Tags kann auch über eine eingeblendete Tastatur erfolgen.
- Metrics: Metriken (wie Lines of Code, Number of Public Attributes, Number of Inner Classes, verschiedene Ratios) werden als Text dargestellt (Abb. 3).
- Project Management: Projekte können geladen und die Darstellungsform ausgewählt werden.
- Filtering: Zur besseren Übersicht insbesondere bei sehr großen Projekten können bestimmte Packages ein- und ausgeblendet werden.

Die Navigation durch die 3D-Welt erfolgt mittels Maus und Tastatur; per „Autopilot“ kann auch eine bestimmte Klasse „angeflogen“ werden. Weitere Details zur FlyThruCode sind in [4] beschrieben, was auf der 11th International Conference on Computer Science and Education (IEEE-ICCSE 2016) in Nagoya (Japan) präsentiert und dort mit dem best paper award ausgezeichnet wurde.

Immersion mittels VR-Geräte

Eine deutlich höhere Immersion ist mit speziellen VR-Geräten erreichbar. Hierzu besitzt das Tool Schnittstellen zu gängigen VR-Geräten, insbesondere Oculus Rift, Google Cardboard, und HTC Vive. Die letzten zwei wurden praktisch erprobt. Die Google Cardboard hat den Vorteil dass sie relativ günstig ist und somit von einer großen Zielgruppe genutzt werden kann. Nachteil sind die eingeschränkten Interaktionsmöglichkeiten: Da das Sichtfeld komplett verdeckt ist, ist die Bedienung von Menüs und die Eingabe von Texten – zum Beispiel für die Auswahl von Projekten, die Suchfunktion oder das Taggen – erschwert. Die HTC Vive (Abb. 4) bietet hier bessere Möglichkeiten: Die Menüauswahl und die Navigation erfolgt mittels Controller, für die textuelle Eingabe steht eine virtuelle Tastatur zur Verfügung. Insgesamt hat sich gezeigt, dass die Übersicht über die Projektstruktur im dreidimensionalen Raum durch die vollständige Immersion deutlich besser ist, als auf dem Bildschirm. Mittels Kopfbewegung kann man sich im virtuellen Raum orientieren. Um die Gefahr der motion sickness zu minimieren, kann neben der Kopf- auch die Augenbewegung erfasst und für die Navigation genutzt werden. Für eine möglichst optimale Schrifterkennung wurde mit dem Font „Brian James“ eine Schriftart gewählt, die sich durch kleine Serifen und klarer Darstellung auszeichnet. Das HUD wurde als Screen in der Landschaft dargestellt. Weitere Details zum VR-FlyThruCode sind in [5] beschrieben.

Evaluation

Das Projekt war ursprünglich ein proof of concept: Es war zunächst nicht klar, ob das Vorhaben, Software-Strukturen dreidimensional darzustellen, gelingen würde und ob dadurch tatsächlich ein (möglichst messbarer) Mehrwert erzielt werden kann. Aus diesem Grund haben wir eine Evaluation mit der anvisierten Zielgruppe – Studierende der Informatik – durchgeführt.

Bei der Evaluation stammten die Probanden aus unterschiedlichen Semestern, hatten somit unterschiedliche Vorkenntnisse: Im zweiten Semester wird die objektorientierte Programmierung eingeführt, UML ist Bestandteil einer Vorlesung im vierten Semester. Im Rahmen dieser Veranstaltungen werden auch die Tools Eclipse und Enterprise Architect vorgestellt und eingesetzt.

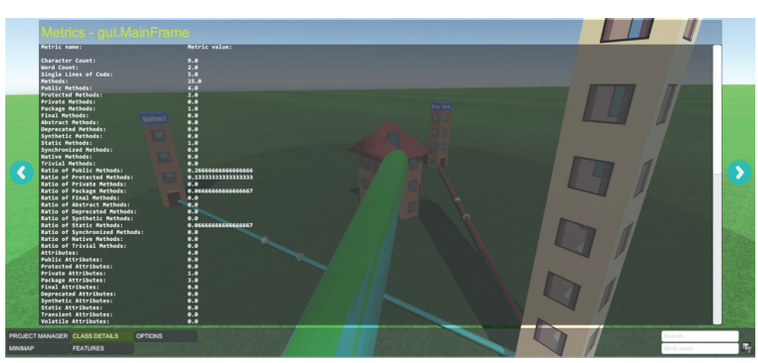


Abb. 3: Metriken Fenster im HUD

Abb. 4: HTC-Vive (auf der Leinwand ist der momentan betrachtete Ausschnitt der virtuellen Welt und der Source Code-Screen für eine ausgewählte Klasse dargestellt)

Die Evaluation zeigte, dass die Anwendung so erfolgreich war, dass sie nun weiter entwickelt wird: Insgesamt wurde ein Mehrwert gegenüber den an der Hochschule vorhandenen und in der Lehre bereits eingesetzten Software-Entwicklungstools (insbesondere Enterprise Architect und Eclipse) registriert. Insbesondere die hohe Motivation der Studierenden bei der Bearbeitung von Fragestellungen bezüglich Programm-Code-Strukturen war deutlich erkennbar.

Danksagung

Wir danken den Studierende Lisa Philipp, Dominik Bergen, Alexandre Matic und Camil Pogolski für die Implementierung und Evaluierung, bei der auch der wissenschaftliche Mitarbeiter Christian Silfang mitgewirkt hat.

Referenzen

[1] F. P. Brooks, Jr., The Mythical Man-Month. Boston, MA: Addison-Wesley Longman Publ. Co., Inc., 1995.
[2] L. Feijs and R. De Jong, „3D visualization of software architectures," Comm. of the ACM, vol. 41, no. 12, 1998, pp. 73-78.
[3] A. R. Teyseyre and M. R. Campo, „An overview of 3D software visualization," Visualization and Computer Graphics, IEEE Transactions on, vol. 15, no. 1, 2009, pp. 87-105.
[4] R. Oberhauser, C. Silfang, and C. Lecon. „Code structure visualization using 3D-flythrough," 11th International Conference on Computer Science & Education (ICCSE), 2016. IEEE, 2016, pp. 365-370.
[5] R. Oberhauser and C. Lecon. „Immersed in Software Structures: A Virtual Reality Approach," 10th International Conference on Advances in Computer-Human Interactions (ACHI 2017). IARIA, 2017.

Kontakt

Prof. Dr. Carsten Lecon, Hochschule Aalen, Tel 07361/576-4365, E-Mail: carsten.lecon@hs-aalen.de

Prof. Roy Oberhauser, Hochschule Aalen, Tel 07361/576-4206, E-Mail: roy.oberhauser@hs-aalen.de