Interaktive Extraktion von Software-Komponenten

Fabian Beck, Alexander Pavel und Stephan Diehl

Universität Trier
54286 Trier, Germany

{beckf,diehl}@uni-trier.de

Zusammenfassung

Im Rahmen der Wiederverwendung von Software, des Software-Redesigns oder des Outsourcings kann es sinnvoll sein, eine Komponente eines Software-Systems herauszutrennen, damit dieses Teilsystem unabhängig und effizient weiterentwickelt werden kann. Schwerpunkt dieser Arbeit ist das Design und die Realisierung einer Benutzerschnittstelle, die ein semiautomatisches Verfahren zur Komponentenextraktion nahtlos in die Entwicklungsumgebung Eclipse integriert und dabei eine interaktive und iterative Arbeitsweise unterstützt. Eine besondere Herausforderung stellt die effiziente Darstellung von Abhängigkeiten zwischen extrahierter Komponente und Restsystem dar.

1 Einleitung

Eine Software-Komponente wird gewöhnlich als ein Teil eines Software-Systems verstanden, der mit einer klare Schnittstelle versehen ist, unabhängig ausgeliefert wird und durch Dritte weiterverwendet werden kann [2]. Um ein größeres Anwendungsgebiet zu adressieren, wollen wir im Kontext dieser Arbeit jedoch den Begriff der Komponente etwas weiter fassen: Zusätzlich zu einer klar definierten Schnittstelle fordern wir nur, dass die Komponente unabhängig vom Restsystem weiterentwickelt werden kann. Eine Komponente muss also nicht eigenständig lauffähig sein, sondern kann weiterhin stärkere Abhängigkeiten zu anderen Teilen des Systems aufweisen. Möchte man eine solche Komponente zwecks eigenständiger Weiterentwicklung extrahieren, ist es dennoch vorrangiges Ziel, die Anzahl solcher Abhängigkeiten zu reduzieren – jede Abhängigkeit im Programmtext schafft Abhängigkeiten in der Software-Entwicklung, die einen raschen Fortschritt potentiell gefährden können.

Das hier vorgestellte Werkzeug zur interaktiven Komponentenextraktion basiert auf dem semiautomatischen Verfahren von Marx et al. [1], welches das Software-System als ein durch Abhängigkeiten verbundenes Netzwerk aus Klassen versteht und wie folgt arbeitet: Nachdem der Benutzer eine zu extrahierende Komponente durch Angabe zentraler Klassen grob skizziert hat, berechnet das Werkzeug den minimalen Schnitt im Netzwerk zwischen den zu extrahierenden Klassen und dem Kern des Restsystems (st-Cut-Problem) und schlägt so eine genaue Aufteilung des Systems vor. Zusätzlich wird automatisiert eine Schnittstelle für die so geschaffene Komponente erzeugt.

2 Interaktive Komponentenextraktion

In der Arbeit von Marx et al. [1] wird bereits ein interaktives Werkzeug zur Komponentenextraktion als eigenständige Anwendung beschrieben und in einer qualitativen Benutzerstudie evaluiert. Die im Folgenden eingeführte Benutzerschnittstelle ist als Weiterentwicklung des ursprünglichen Werkzeugs zu verstehen. Sie ist als Plug-In für die Entwicklungsumgebung Eclipse exemplarisch implementiert und erlaubt die Extraktion von Java-Komponenten.

Aus der vorherigen Arbeit übernommen wurde ein dreiteiliger Aufbau der Hauptansicht des Werkzeugs (Abbildung 1): Links befindet sich das Ursprungssystem (*original component*), rechts die zu extrahierende Komponente (*extracted component*); getrennt sind beide durch eine Repräsentation der Komponenten-Schnittstelle (*contract*). Diese Ansicht ist als *view* in Eclipse realisiert und kann gleichberechtigt zu anderen Ansichten angezeigt und genutzt werden.

Die drei Teile der Hauptansicht wurden mit Hinblick auf eine verbesserte Skalierbarkeit neu gestaltet. Sie stellen jeweils eine Menge von Klassen und Interfaces in einer durch die Package-Struktur des Software-Projekts definierten hierarchischen Gliederung dar. Wie im *Package Explorer* von Eclipse können Packages bis auf Methodenebene interaktiv auf- und zugeklappt werden. Der Benutzer weist nun interaktiv Elemente zu den Komponenten fest zu (schwarze Schrift). Der automatische Algorithmus zur Komponentenextraktion teilt die übrigen, grau dargestellten Elemente optimal auf die Komponenten auf. Diese Aufteilung kann iterativ verfeinert und verbessert werden.

Besonderes Merkmal der hierarchischen Darstellung der Komponenten ist eine aggregierte Repräsentation von Abhängigkeiten, welche die verschiedenen Software-Komponenten verbinden. Wir betrachten dazu drei aus UML entlehnte Typen von Abhän-

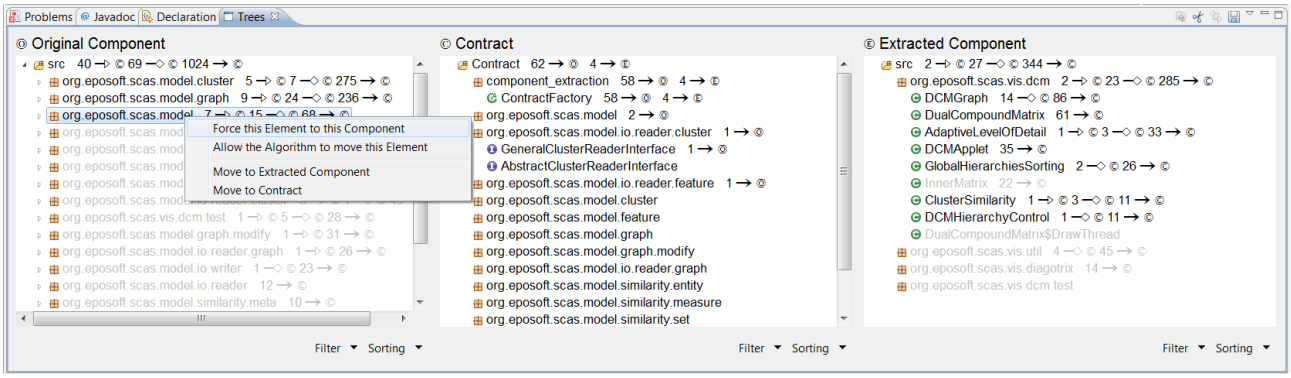

Abbildung 1: Dreigeteilte Hauptansicht des Eclipse Plug-Ins zur Komponentenextraktion.

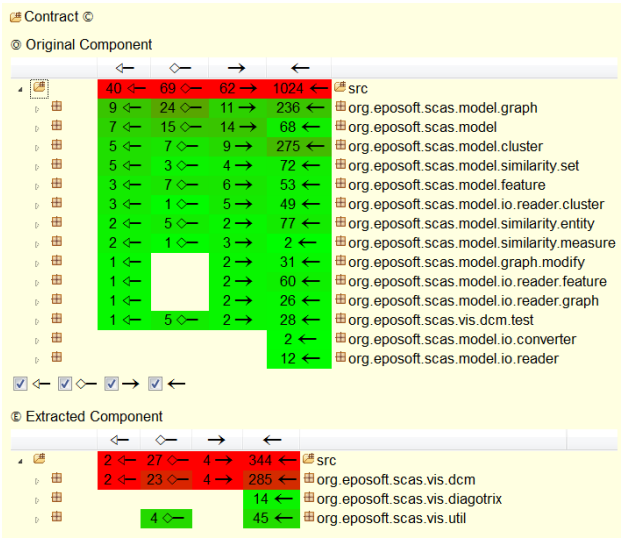

Abbildung 2: Tooltip zur detaillierten Auflistung von Abhängigkeiten.

gigkeiten: Vererbung (—▷), Aggregation (—◇) und Assoziation (→). Unter Verwendung der angegebenen Symbole wird für jedes Hierarchieelement angegeben, wie viele solcher Abhängigkeiten zu den jeweils anderen Software-Komponenten existieren (original component: Ⓞ; contract: Ⓒ; extracted component: Ⓔ). Abhängigkeiten innerhalb einer Komponente sind hier weniger von Bedeutung und werden nicht referenziert. Durch eine Sortierung nach Anzahl der Abhängigkeiten können die problematischsten Stellen der momentanen Komponentenaufteilung, d.h. die am meisten Abhängigkeiten verursachen, schnell erkannt werden.

Zusätzlich zu dieser übersichtlichen, aggregierten Darstellungen ist es auch möglich, die Abhängigkeiten im Detail abzufragen. Dies geschieht über einen Tooltip, der zum jeweiligen Element eingeblendet wird und in ein permanent angezeigtes Fenster umgewandelt werden kann (Abbildung 2). Ein solcher Tooltip zeigt alle relevanten Abhängigkeiten aus der Perspektive des gewählten Elements. Unterschieden wird hierbei erneut der Typ der Abhängigkeiten und zu welcher Komponente diese gerichtet sind. Die Anzahl der Abhängigkeiten ist zur besseren Übersicht tabellarisch angegeben; die Zeilen der Tabellen sind weiterhin hierarchisch gegliedert. Durch eine Farbskala, die die Anzahl der Abhängigkeiten kodiert und spaltenweise normalisiert ist, wird direkt auf potentiell kritische Komponentenzuordnungen hingewiesen. Mittels Doppelklick kann jederzeit der Quelltext der angeklickten Klasse geöffnet und bearbeitet werden. Die Darstellung aus der Perspektive einzelner Elemente erlaubt es, auch große Netzwerke von Abhängigkeiten einfach zu untersuchen und mögliche Probleme geziehlt aufzulösen.

3 Fazit

Das vorgestellte Eclipse Plug-In setzt ein semiautomatisches Verfahren zur Komponentenextraktion als interaktives Werkzeug um. Der entschiedene Vorteil gegenüber der ursprünglichen Benutzerschnittstelle [1] ist die übersichtliche und effektive Repräsentation der Komponenten und deren Abhängigkeiten, die nun auch den Umgang mit größeren Software-Projekten erlaubt. Durch die Darstellung der Abhängigkeiten als explizite Verbindungslinien konnten bisher nur kleine Projekte sinnvoll bearbeitet werden. Zudem erlaubt die nahtlose Integration in die Entwicklungsumgebung Eclipse den einfachen Wechsel zwischen Quelltext-Bearbeitung und Komponentenextraktion. Diese Möglichkeit wurde in der Evaluierung des ursprünglichen Werkzeugs als fehlendes, wichtiges Feature identifiziert. Insgesamt verspricht die interaktive Extraktion von Komponenten die Weiterentwicklung eines Software-Systems zu erleichtern.

Literatur

[1] A. Marx, F. Beck, and S. Diehl. Computer-Aided extraction of software components. In *17th Working Conference on Reverse Engineering (WCRE 2010)*, 2010.

[2] C. Szyperski and C. Pfister. Workshop on component-oriented programming, summary. In M. Mühlhäuser, editor, *Special Issues in Object-Oriented Programming - ECOOP '96 Workshop Reader*. dpunkt Verlag, 1997.