Hypermodelling

Data-Warehouse-Infrastruktur zur Codeanalyse

Tim Frey, Matthias Gräf

Strukturuntersuchungen von Quelltext sind derzeit mit erheblichem Aufwand verbunden. Hypermodelling setzt auf der Data-Warehouse-Technologie auf und ermöglicht mehrdimensionale Sichtweisen und Analysen von Quelltext. Hierdurch ist es möglich, Codeuntersuchungen mit strukturierten, mehrdimensionalen Abfragen auf einfache Art und Weise durchzuführen. Dieser mehrdimensionale Ansatz kann dann dazu verwendet werden, hochgranulare Quelltextsuchmaschinen zu erstellen, Quelltext-Audits durchzuführen oder auch Cockpits für die Codestruktur zu erzeugen.

Hintergrund

Wiederverwendbarkeit, Wartbarkeit und Architektur von Quelltext stellen wichtige Qualitätsanforderungen für zuverlässige Software dar. Derzeit sind Strukturuntersuchungen und das Auffinden geeigneter Softwareartefakte nur unter erheblichem Aufwand möglich.

Ein Problem hierbei ist die Komplexität heutiger Quelltextbasen. Diese umfassen Millionen Zeilen von Quelltext und sind schwer zu überblicken und zu untersuchen. Zusätzlich existiert eine Vielzahl an mit Quelltext assoziierten Daten, wie zum Beispiel Informationen aus Issue-Tracking-Systemen oder Code-Smell-Detektoren, die, direkt oder indirekt, auch Teil des Softwareentwicklungsprozesses sind. Oftmals ist es nahezu unmöglich, all diese Blickwinkel bei der Entwicklung zu betrachten. Dabei stellt sich stets das Problem, dass die fehlende Integrationsmöglichkeit der verschiedenen Daten es nicht ermöglicht, den Quelltext und dessen Struktur aus unterschiedlichen und differenzierten Blickwinkeln zu betrachten.

Auffinden von Quelltextartefakten

Suchfunktionen in der integrierten Entwicklungsumgebung ermöglichen spezifische Suchmöglichkeiten. Dennoch sind solche Suchfunktionen auf eine lokale Datenbasis beschränkt und mit einer Entwicklungsumgebung verknüpft. Dies macht sie unbrauchbar für einen zentralisierten Zugriff und große Datenmengen.

Um Codefragmente in großen Quelltextbasen aufzudecken, können heute Codesuchmaschinen verwendet werden. Diese unterscheiden sich jedoch kaum von regulären Suchmaschinen. Es werden daher vorwiegend Zeichenketten in Dokumenten gesucht. Es ist zum Beispiel oftmals nicht möglich, alle Methoden, die einen String und eine von `List` abgeleitete Klasse als Parameter haben, direkt durch eine Anfrage zu suchen. Es ist also oftmals schwierig eine Anfrage der Form

```
select all methods with parameters (String and <? extends List>)
```

zu formulieren, um beispielsweise Funktionen wie die folgende

```
public void ResolveCustomer(String name, ArrayList){…}
```

zu finden. Eine weitere Anforderung ist es, neben der Suche von öffentlichem Quelltext, auch ein Auffinden von Artefakten aus firmeneigenem Quelltext zu ermöglichen und hierfür eine Technologie zu verwenden, die sich in eine bestehende Firmeninfrastruktur integrieren lässt, da die öffentlichen Quelltextsuchmaschinen logischerweise den privaten Quelltext nicht kennen sollen und somit hier nicht verwendbar sind.

Schwächen bei der Quelltextsuche

Eine weitere Anforderung, die bei dem gesuchten Quelltext oftmals vor dessen Wiederverwendung gestellt wird, ist die Erfüllung verschiedener Qualitätsstandards. Diese können dabei für verschiedene Unternehmen durchaus variieren und anders spezifiziert sein. Für das eine Unternehmen sind es Namenskonventionen und für ein anderes eine hohe Testabdeckung durch Unit-Tests. Daher wäre es von Vorteil, wenn Codesuchmaschinen flexibel solche Qualitätsstandards bei der Suche berücksichtigen können. Die Klassifikation der Ergebnisse sollte daher mit individuellen Mitteln in einer Abfrage erfolgen können, um eine Anpassung an verschiedene Rahmenbedingungen zu gewährleisten.

Eine Suchmaschine und die Suche in der integrierten Entwicklungsumgebung unterliegen, zusätzlich zu der äußerst eingeschränkten Struktursuche, der Beschränkung, dass keine primäre Unterstützung für Qualitätsmerkmale von Quelltext bei einer Suche verwendet werden kann. Grundlegend ist diese Einschränkung darauf zurückzuführen, dass keine mit Quelltext assoziierten Informationen, wie Qualitätskriterien, direkt in eine Suche mit einbezogen werden. Somit können weder Qualitätsmerkmale beachtet werden, noch ist es möglich, auf individuelle Qualitätsanforderungen von Unternehmen einzugehen. Daher ist es wünschenswert, Suchfunktionen zur Verfügung zu haben, die mit Quelltext assoziierten Informationen umgehen können und auch erweiterte Strukturinformationen von Quelltext in der Suche zulassen.

Herausforderung Strukturanalyse

Oftmals ist es beispielsweise in Outsourcing-Szenarien gewünscht, den fremden Quelltext schnell und effizient auf ein Mindestmaß von Qualitätskriterien zu untersuchen, um eventuelle Reklamationen spezifizieren zu können (s. Kasten „Suchbeschränkung anhand mit dem Quelltext assoziierter Informationen").

Suchbeschränkung anhand mit dem Quelltext assoziierter Informationen

Eine Begrenzung einer Quelltextsuche aufgrund von Qualitätseigenschaften ist heutzutage oftmals nicht flexibel durchzuführen. Beispielsweise könnten die folgenden Methoden nicht mit hundertprozentiger Testabdeckung oder 0 Style Issues gesucht werden.

Methode	Testabdeckung	Style Issues
ResolveCustomer	90 %	3
SaveCustomer	70 %	2
CreateCustomer	100 %	0
DeleteCustomer	100 %	0

Strukturbezogene Kennzahlen

Bei Änderungen in Superklassen ist es wichtig zu wissen, wie oft Methoden überschrieben werden. Wird eine Methode besonders häufig überschrieben, so hat deren Änderung in der Superklasse vermutlich stärkere Auswirkungen als eine weniger häufig überschriebene. Entwickler könnten zudem den Rückschluss bilden, dass diese Methode zukünftig als abstrakte Methode definiert werden sollte.

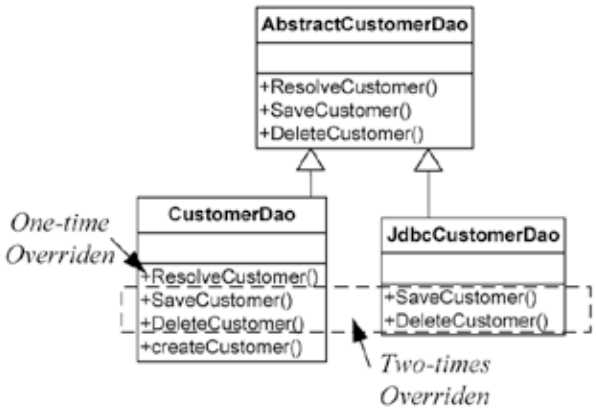

Abb. 1. Überschreiben als Strukturkennzahl

Sogenannte *Cockpits* erlauben es, Projekte zu untersuchen. Ein Cockpit, häufig auch Dashboard (dt. etwa Instrumententafel) genannt, ist die technische Umsetzung und Visualisierung eines Kennzahlensystems [Cockpit]. Darin werden verschiedene Projektinformationen und deren Visualisierung übersichtlich an einem zentralen Punkt dargestellt. Führungskräfte benutzen solche Cockpits, um einen Überblick über Projekte zu erhalten und diese zu steuern.

Bisherige Cockpits ermöglichten jedoch nur die Kontrolle dieser projektbezogenen Kennzahlen. Es ist bisher nicht möglich, Cockpits zur Überprüfung der eigentlichen Quelltextstruktur einzusetzen (siehe Beispiel im Kasten „Strukturbezogene Kennzahlen“ für solche Kennzahlen). Der Grund, warum bisher keine Cockpits für die Codestruktur existieren, liegt vermutlich darin, dass Quelltext im Regelfall in Dateien und als Text strukturiert ist, was eine Auswertung nur unter erheblichem Aufwand möglich macht. Somit stellt sich die Herausforderung, solche Sachverhalte in Cockpits zu untersuchen.

Hypermodelling

Die Hypermodelling-Technologie befasst sich damit, die zuvor beschriebenen Probleme zu lösen. Hypermodelling unterstützt sowohl die Möglichkeit der Codesuche als auch die der strukturierten Analyse auf Basis einer einheitlichen und standardisierten Infrastruktur (s. Abb. 2). Hypermodelling setzt dabei auf Data-Warehouse-Technologie (s. Kasten „Data-Warehouse“), da diese ohnehin in größeren Unternehmen etabliert ist [Fr12a]. Dies ermöglicht es, der Komplexität und dem Umfang von modernen Codebasen gerecht zu werden. Zusätzlich verfügen Data-Warehouse-Systeme über eine Vielzahl von leistungssteigernden Mechanismen wie hauptspeicherresidente Technologie, Vorberechnungen, Subaggregate und Query Profiling, um nur einige wenige zu nennen. Zusätzlich stehen nativ mehrdimensionale Analysemöglichkeiten zur Verfügung. Dies bedeutet beim Einsatz der Technologie zur Quelltextanalyse, dass Code und die damit assoziierten Artefakte von verschiedenen Perspektiven aus untersucht werden können. Ferner stehen fertige Infrastrukturen mit Werkzeugen bereit, um mit geringem Aufwand Analysen und Cockpits für verschiedene Anwendungsfälle zu erstellen.

Data-Warehouse

Data-Warehouse-Systeme werden im betriebswirtschaftlichen Kontext eingesetzt, um Berichte zu erstellen. Dabei werden die Daten aus den verschiedensten operativen Systemen, wie zum Beispiel CRM oder ERP, extrahiert, homogenisiert und in einer Datenbank zusammengeführt. Die Daten werden dann in sogenannte Data-Cubes geladen. Diese stellen mehrdimensionale Strukturen dar und ermöglichen es, Sachverhalte aus verschiedenen Blickwinkeln zu betrachten und zu analysieren. Beispielsweise können Hierarchien oder Verhältnisse sowie der Vertrieb von Produkten oder Produktgruppen für einzelne Länder oder Regionen betrachtet werden.

Grundlegend ist die Zielsetzung dieser Systeme, dass schnell und dynamisch große und firmenweite Datenbestände effizient untersucht werden können, um betriebswirtschaftliche Entscheidungen zu treffen. Data-Warehouse-Systeme stellen daher verschiedene mehrdimensionale Operationen und Infrastrukturen zur Untersuchung und Darstellung dieser großen Datenmengen bereit.

Teil dieser Infrastrukturen sind zahllose Mechanismen, um Performance-Optimierungen vorzunehmen. Zusätzlich existiert eine Vielzahl an Werkzeugen, die es schnell und automatisiert ermöglichen, Berichte zu erzeugen. Oftmals werden auch aufgrund der homogenisierten Daten Data-Mining-Techniken angewandt, um Muster in der Datenbasis aufzuspüren.

Funktionsweise von Hypermodelling

Grundlegend funktioniert der Hypermodelling-Ansatz wie in Abbildung 2 gezeigt. Quelltext wird zunächst in ein relationales Schema transformiert. Dies ermöglicht die Nutzung der Data-Warehouse-Infrastruktur. In diesem Schema sind die Beziehungen des Quelltextes selbst beschrieben. Solche Relationen beinhalten die Abbildung von Methodenaufrufen, Vererbungen, Methodenparametern und weiterer Verknüpfungen im Quelltext. Zusätzliche Daten, die bei der Softwareentwicklung auftreten, wie beispielsweise Qualitätseigenschaften, werden dann ebenfalls in dem Data-Warehouse gespeichert und verweisen auf die zugehörigen Quelltextartefakte.

Abbildung 2 zeigt exemplarisch, um welche Daten es sich hierbei handeln kann: Informationen über Style Issues oder Bugs wie aus findbugs oder Bugzilla. Aber auch Daten aus Testergebnissen wie Junit und sämtliche weitere Daten, die im Quelltext assoziiert sind, können auf diesen verweisen. Daher deutet die Grafik an, dass sämtliche Daten, die sich auf den Quelltext beziehen, in das Data-Warehouse transformiert und geladen werden. Es ist dabei zu sehen, dass das Schema des Quelltextes selbst im Zentrum der Abbildung angeordnet ist. Durch diese zentrale Stellung ist somit eine Erweiterbarkeit gewährleistet, sodass die anderen bei der Entwicklung auftreten-

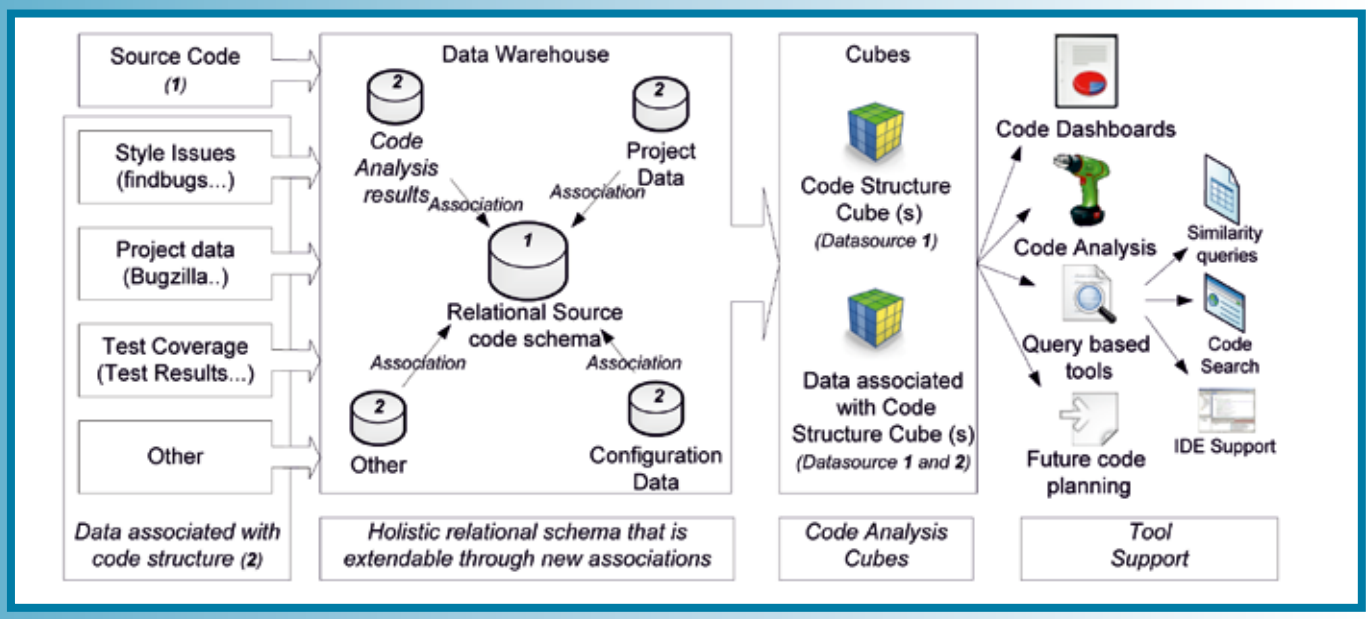

Abb. 2: Die Hypermodelling-Infrastruktur

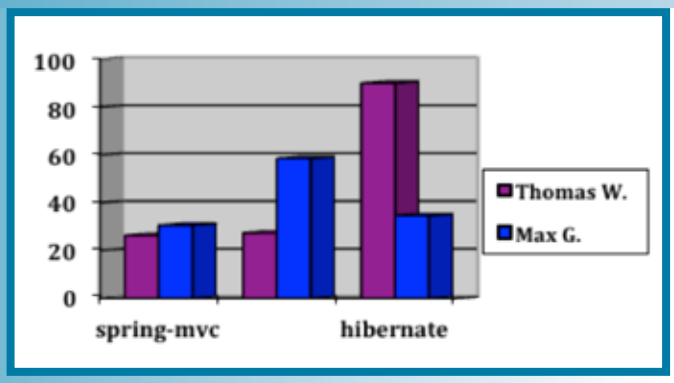

Abb. 3: Intensität der Frameworknutzung von Entwicklern: Selbstverständlich können auch Abfragen darüber Aufschluss geben, welche Entwickler mit ihrem Quelltext welche Bibliotheken nutzen, um Indikatoren über die Wissensverteilung im Entwicklungsteam zu erhalten

den Artefakte sich auch auf den Quelltext beziehen. Durch das relationale Quelltextschema sind indirekt sämtliche Informationen von verschiedenen Werkzeugen, welche mit Quelltext assoziiert werden, miteinander verknüpft.

Daten-Cubes zur Quelltextabbildung

Durch die Ausrichtung aller Daten auf eine zentrale Quelltextstruktur können somit Analyse-Cubes von Quelltext erstellt werden, die alle vorhandenen Beziehungen einschließen. Ein Cube zeigt dabei einen mehrdimensionalen Blickwinkel auf die relationalen Sachverhalte und bildet die vorkommenden Hierarchien im Quelltext und assoziierten Fragmenten ab.

Solche Cubes können zum Beispiel die Codestrukur und die damit verbundenen Qualitätsinformationen beinhalten. Der Quelltext selbst stellt dabei die Verbindung zu den Daten aus den verschiedenen Werkzeugen dar. Dies ermöglicht die Beantwortung verschiedener Fragestellungen durch Abfragen auf die mehrdimensionale Datenbasis. (s. Beispiel in Abb. 3). Eine Möglichkeit ist es, Style Issues aus findbugs im Zusammenhang mit genutzten Bibliotheken zu untersuchen, um zu bestimmen, ob hier Zusammenhänge bestehen.

Grundlegend können durch die mehrdimensionale Abbildung von Quelltext sämtliche Beziehungen Untersuchungsgegenstand sein. Die Nutzung der Analyse-Cubes ermöglicht es dabei, die verschiedenen Blickwinkel schnell und effizient zu beleuchten [FrKöSa11].

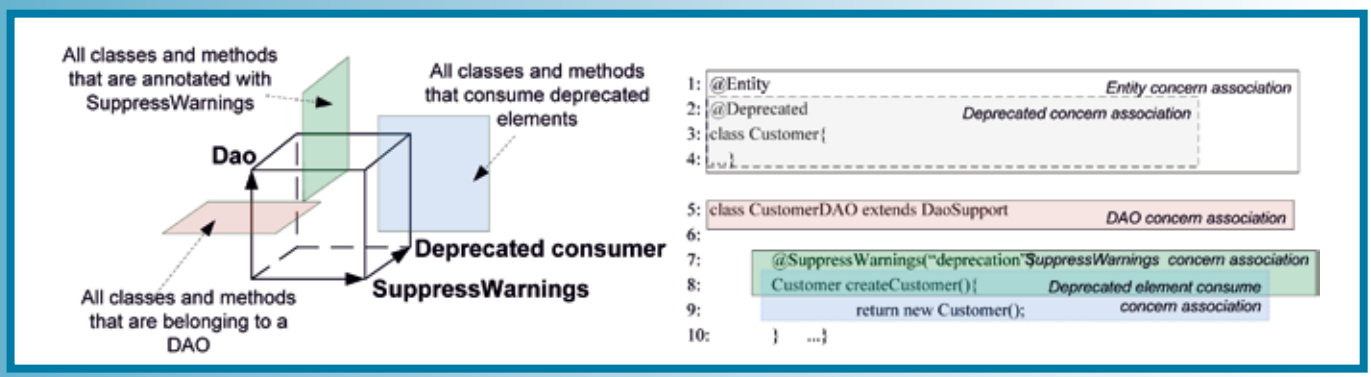

Abb. 4: Beispielhafte Einordnung von Quelltext in einen mehrdimensionalen Raum

Technik: Beispielhaftes Quelltext-Cube-Mapping

Abbildung 4 zeigt ein exemplarisches Mapping von Quelltext zu einem mehrdimensionalen Würfel. Aufgrund der Darstellbarkeit wurden dabei die Dimensionen auf drei (DAO, Deprecated Consumer, SuppressWarnings) begrenzt. Die verschiedenen Farben (rosa, blau, grün) ordnen dem Code die verschiedenen Dimensionen zu. Die Klasse `Customer` gehört zum einen zur Dimension aller mit `@Entity` markierten Klassen und zum anderen auch zu den veralteten Elementen (`@Deprecated`). Solche Zuordnungen können auch, wie in der Klasse `CustomerDAO` zu sehen, aufgrund von Superklassen (`extends DaoSupport`) geschehen. Zuletzt wird auch noch eine Verbindung zum Konsum veralteter Dokumente, die zum Beispiel durch einen Compiler aufgedeckt wurde, als Dimension verstanden.

Wie in Abbildung 4 zu sehen ist, sind einige der Dimensionen des Quelltextes im zweiten Teil der Abbildung angeordnet und stellen einen mehrdimensionalen Raum dar. Es gilt sich nun vorzustellen, wie dieser Raum in Scheiben geschnitten wird. Zum Beispiel, dass sämtliche Quelltextfragmente ausgewählt werden, die zu der DAO-Dimension gehören. Grundlegend ist eine solche für sämtliche Dimensionen möglich. Durch die Kombination verschiedener „Scheiben" können dann Quelltextfragmente aufgedeckt werden, die zu mehreren Dimensionen zugleich gehören.

So würde zum Beispiel die Anfrage nach sämtlichen Fragmenten, die Warnungen unterdrücken und veraltete Elemente konsumieren, in der `createCustomer`-Methode resultieren.

Code-Cockpits als Analysewerkzeug

Durch Analyse-Cubes ist es nun möglich, Werkzeuge aus dem Data-Warehouse-Bereich zu verwenden, die verschiedene Visualisierungen der Resultate von Abfragen ermöglichen. Hierbei können Cockpits zusammengestellt werden, die es ermöglichen, Untersuchungen zu zentralisieren und grafisch aufzubereiten.

Exemplarisch ist ein solches Dashboard zur Untersuchung der Erbschaftsbeziehungen in Abbildung 5. Dort ist zu sehen, dass Tortendiagramme, Tachos, Balkendiagramme und Verteilungsintensitätsdiagramme die Beziehungen im Quelltext grafisch beschreiben. Die Balkengrafik gibt Aufschluss darüber, welche Arten von Problemen im derzeitig untersuchten Projekt gefunden wurden. Die Tortendiagramme zeigen, von welchen Packages besonders häufig geerbt (*absolute Anzahl*) wird und wie viel *unterschiedliche* Supertypen (Superkassen und Interfaces) davon verwendet werden. Hierbei ist es möglich, jedes Element der Package-Hierarchie zu untersuchen und das Cockpit danach zu filtern. Der direkte Vergleich zwischen den unterschiedlichen Supertypen und der kompletten Anzahl findet sich im Tacho. Von dieser grafischen Übersicht aus können tiefer gehende Untersuchungen direkt im Cockpit durchgeführt werden. So können beispielsweise gezielt Packages mit Supertypen als Filter ausgewählt werden, um zu ermitteln, welche Methoden welcher Typen besonders häufig überschrieben werden. Im Beispiel in Abbildung 5 ist zu sehen, dass die `setUp`- und `tearDown`-Methoden besonders häufig überschrieben wurden. Dies ist äußerst logisch, da diese zur Initialisierung und Beendigung von Testfällen dienen.

Insgesamt ist anzumerken, dass durch die Nutzung von Data-Warehouse-Werkzeugen der Aufwand einer Cockpit-Erstellung zu einem überschaubaren Kleinprojekt wird, wofür zuvor Monate notwendig waren. Somit können solche Cockpits nun flexibel für spezielle Analysezwecke erstellt and angepasst werden.

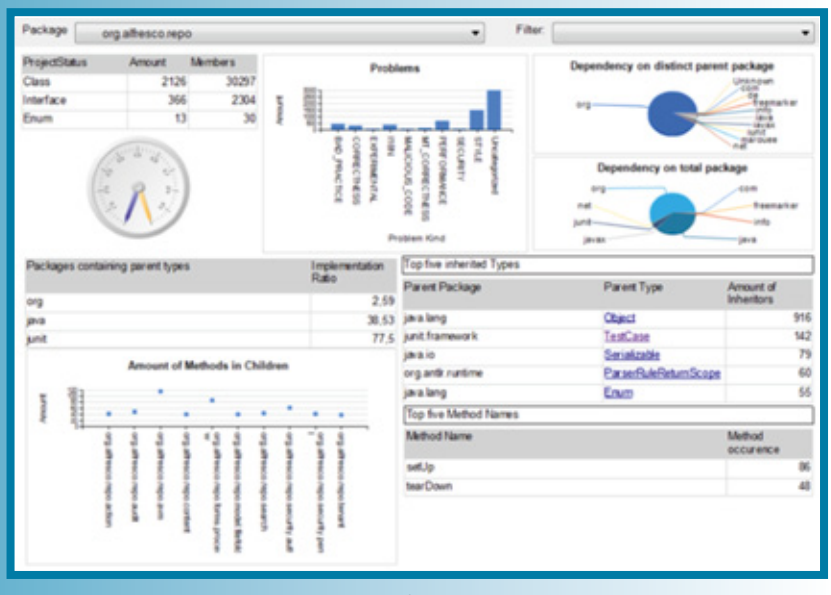

Abb. 5: Dashboard zur Untersuchung der Erbschaftsbeziehungen

Strukturierte Quelltext-Audits durch Abfragen

Ein weiterer Anwendungsfall ist die direkte Untersuchung der Quelltextstruktur mit Abfragen. Mit diesen können schnell und effizient Statistiken über die Struktur von Quelltext erstellt werden [FrKö12b]. Ein typisches Szenario ist die Eingliederung von extern gefertigtem Quelltext in die eigene Codebasis. Etwaige Schwachstellen, die die Anforderungen, wie die schon zuvor erwähnte Testüberdeckung, nicht erfüllen, können somit sukzessive durch Abfragen gezielt geortet und reklamiert werden. Die geforderten Qualitätsmerkmale können dann als zu erfüllende Kriterien, als Kennzahlen, spezifiziert und überprüft werden. (s. Kasten „Codeaudits und Nacherfüllung")

Anwendungsfall Codesuche

Zusätzlich können auf Basis von Abfragen mit geringem Aufwand spezialisierte Werkzeuge erstellt werden. Ein Anwendungsfall hier ist eine Codesuchmaschine, welche die in der Einleitung genannten Beschränkungen durch hochgranulare Abfragen und das Einbeziehen von Qualitätsfaktoren überwindet. Beispielsweise können Details wie die Supertypen von Methodenparametern gezielt angegeben werden, um gewünschte Methoden aufzudecken, da die komplexe mehrdimensionale Quelltextstruktur in Analyse-Cubes abgebildet ist [FrKö12a,Fr12b].

Durch das Einbeziehen von assoziierten Informationen des Quelltextes können zusätzlich noch Beschränkungen bei der Suche angewandt werden. Diese Details können dann bei Filterung oder Sortierung der Suchergebnisse einfließen. Eine solche Beschränkung oder Sortierung kann zum Beispiel darauf basieren, Methoden mit vollständiger Testabdeckung und der Erfüllung einer Mindestanzahl von Codierungsrichtlinien aufzudecken.

Codeaudits und Nacherfüllung

Die Reklamation und Nacherfüllung bei Codeaudits erfolgt in vier Schritten (s. Abb. 6):

- Der extern gefertigte Quelltext wird geliefert und durch Hypermodelling analysiert. Bei diesem Vorgang werden assoziierte Codierungsrichtlinien des Unternehmens direkt mit dem Quelltext zusammen in die Analyse-Cubes geladen.
- Danach untersuchen Qualitätsexperten die Codestruktur mit Abfragen und bestimmen Teile des gelieferten Quelltextes, die nicht den geforderten Qualitätskriterien entsprechen, und erstellen eine Reklamation sowie Zielkennzahlen für die betreffenden Quelltextstellen.
- Die Nacherfüllung führt zur Lieferung von verbessertem Quelltext, und dieser wird in Daten-Cubes geladen.
- Dessen Eigenschaften werden dann mit den zuvor spezifizierten Zielkennzahlen verglichen, um die Überarbeitung der Problemstellen zu kontrollieren und zu validieren.

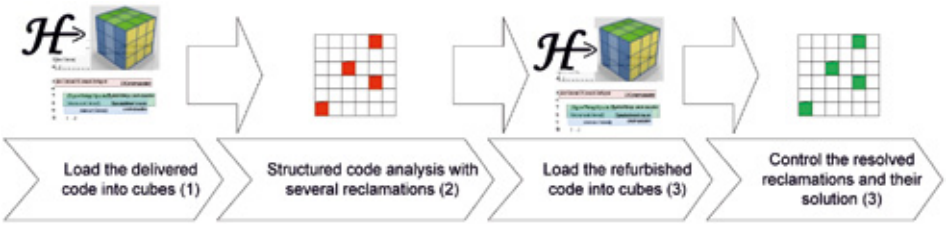

Abb. 6: Prozess der Reklamation und deren Nacherfüllung bei Codeaudits

Bewertung und Ausblick

Hypermodelling erweitert die Möglichkeiten zur Quelltextanalyse um viele realisierbare Anwendungen. Die Nutzung von Data-Warehouse-Technologie ermöglicht einen Einsatz innerhalb bestehender IT-Infrastrukturen von Unternehmen. Insbesondere anspruchsvolle Anwendungen, wie strukturierte Codeaudits von outgesourcten Projekten, versprechen dabei die Qualitätssicherung der Softwareentwicklung zu verbessern und zu beschleunigen.

Der Einsatz von Data-Warehouse-Technologie zur Quelltextanalyse eröffnet zusätzlich weitere Chancen. So können spezialisierte Anwendungen auf Basis der vorgestellten Architektur erstellt werden. Insbesondere die Integration der vielfältigen und verteilten Daten, die im Softwareentwicklungsprozess erzeugt werden, kann zukünftig von den vorhandenen Werkzeugketten und Vorgehensweisen im Data-Warehouse-Bereich profitieren. Zentrale Dashboards können Entscheidungsträger unterstützen, systematisch abstrahierte Einblicke in relevante Informationen über den Entwicklungsfortschritt zu erlangen, zu deuten und zu planen. Zusätzlich erlauben die bestehenden Werkzeuge vielfältige Projektionen der verschiedenen Sachverhalte auf unterschiedlichen Abstraktionsebenen. Dies ermöglicht es, zukünftig spezialisierte Dashboards für verschiedene Rollen zu entwickeln.

Aufgrund des fortgeschrittenen Reifegrades der Hypermodelling-Technologie und der zugehörigen Implementierungen ist es ein wichtiges und ausdrückliches Ziel, diese in industrielle Anwendungen zu überführen. Daher ist Hypermodelling neben der allgemeinen Betrachtung dieses Artikels ein konkretes Projekt, das sich mit der Umsetzung und dem Einsatz der Technologie beschäftigt. Das zugehörige Team hinter der Hypermodelling-Technologie ist auf der Suche nach industriellen Partnern, um Hypermodelling in Lösungen oder Produkte überführen zu können. Um die „most desired needs“ der Anwendungsmöglichkeiten bewerten zu können, ist das Feedback der Entwicklercommunity zu den verschiedenen Anwendungsmöglichkeiten und Einsatzgebieten sehr erwünscht und willkommen. Bei Interesse an einer Technologieüberführung freut sich das Hypermodelling-Team über die Kontaktaufnahme.

Literatur und Links

[Cockpit] `http://de.wikipedia.org/wiki/Kennzahlen-Cockpit`

[FrKö12a] T. Frey, V. Köppen, Hypermodelling Live – OLAP for Code Clone Recommendation, in: Tenth International Baltic Conference on Databases and Information Systems, July 8-11, 2012 (Baltic DB & IS 2012), Vilnius, Lithuania

[FrKö12b] T. Frey, V. Köppen, Exploring Software Variance with Hypermodelling – An exemplary approach, in: 5. Arbeitstagung Programmiersprachen (ATPS'12) im Rahmen der Software Engineering 2012 (SE2012), Gesellschaft für Informatik (GI), Berlin, 2012

[FrKöSa11] T. Frey, V. Köppen, G. Saake, Hypermodelling Introducing Multi-dimensional Concern Reverse Engineering, in: 2nd International ACM/GI Workshop on Digital Engineering (IWDE), Magdeburg, Germany, 2011

[Fr12a] T. Frey, Hypermodelling – A Data Warehouse Approach for Software Analysis, Magdeburger Informatik Tage (MIT) 2012

[Fr12b] T. Frey, Hypermodelling for Drag and Drop Concern Queries, in: Proceedings of Software Engineering 2010 (SE2012), Gesellschaft für Informatik (GI) (Berlin), 2012

Weiterführende Information

http://hypermodelling.com

Tim Frey ist freiberuflicher Berater für IT-Strategie und IT-bezogene Konzepte. Zusätzlich forscht er als Doktorand für die Uni Magdeburg im Bereich Software-Engineering.
E-Mail: tim.frey@tim-frey.com

Matthias Gräf ist Berater im Bereich Business Intelligence. Hierbei gilt sein Interesse stets neuen Entwicklungen und Einsatzgebieten der Data-Warehousing-Technologie.
E-Mail: matthias.graef@gmail.com