der Attributwerte und zum Aufrufen der Methoden des Objektes. Attributwerte oder die Rückgabewerte von Methodenaufrufen können wiederum eine Referenz auf eine Instanz einer Komponente sein, welche wiederum über eine entsprechend generierte Oberfläche inspiziert werden kann. Die sich ergebenen Sequenzen von Benutzerinteraktionen werden durch das Werkzeug aufgezeichnet und können anschließend ebenfalls, in Form von *UML-Sequenzdiagrammen*, graphisch visualisiert werden.

15.4 Ausblick

Die auf die oben beschriebene Art gewonnenen statischen und dynamischen Informationen bilden die Basis für die Schnittstellen der technischen Wrapper. Zukünftige Arbeiten sollen klären, wie diese Informationen genutzt werden können, die Schnittstellen der semantischen Wrapper zu spezifizieren, z.B. die Komposition der originären Operationen der Schnittstelle zu semantisch höherwertigen Operationen.

Literaturverzeichnis

[NSW03] M. Nagl, R. Schneider, and B. Westfechtel. Synergetische Verschränkung bei der A-posteriori-Integration von Werkzeugen. In M. Nagl and B. Westfechtel, editors, *Modelle, Werkzeuge und Infrastrukturen zur Unterstützung von Entwicklungsprozessen*, pages 137–154. Wiley-VCH, 2003.

[NW99] M. Nagl and B. Westfechtel, editors. *Integration von Entwicklungssystemen in Ingenieuranwendungen: substantielle Verbesserungen der Entwicklungsprozesse*. Springer, 1999.

gemeinsame Sitzung mit dem Workshop der GI-Fachgruppe 2.1.4 Programmiersprachen und Rechenkonzepte

16 Werkzeuggestützte Qualitätsuntersuchungen: Ein Erfahrungsbericht

Markus Bauer, Olaf Seng
Forschungszentrum Informatik Karlsruhe (FZI), Haid-und-Neu-Straße 10-14, 76131 Karlsruhe,
{bauer|seng}@fzi.de

16.1 Einführung

Qualitätsuntersuchungen (Softwareassessments) werden durchgeführt, um unter Einsatz von ReverseEngineering Techniken die innere Qualität eines Systems zu untersuchen. Die innere Qualität eines Systems wird durch diejenigen Eigenschaften bestimmt, die der Entwickler wahrnimmt, wie z.B. Erweiterbarkeit, Wiederverwendbarkeit, etc. Für Softwareunternehmen ist es von großem Interesse, eine quantitative Aussage über die innere Qualität ihrer Systeme machen zu können, da diese Entwicklungsaufwand und Wartbarkeit der Systeme beeinflusst und somit die anfallenden Kosten bestimmt. Zudem können kritische Stellen identifiziert werden. Solche Stellen können für Restrukturierungsmaßnahmen, Testfälle und Dokumentation bilden.

16.2 Softwareuntersuchungen – Techniken und Vorgehensweise

Zur Durchführung der Qualitätsuntersuchung werden aus dem Bereich Software-Reengineering bekannte Techniken eingesetzt [FAM99] [BC01]. Dazu gehört an erster Stelle die sogenannte *Faktenextraktion*. Hierbei wir mit Hilfe von Compilerbautechnolgien und graphtheoretischen Konzepten ein Designdatenbank des Systems erstellt, auf der weitere Analysetechniken aufsetzen. *Visualisierungstechniken* können das Verständnis von Architekturen und Strukturen erleichtern und erlauben z.B. eine einfache Überprüfung geschichteter Architekturen. Mit Hilfe von *Abstraktionstechniken* kann die Information in der Designdatenbank verfeinert werden, in dem z.B. bei prozeduralen System Methoden mit Datentypen zu einer Einheit gruppiert werden. Mit Hilfe von *Software-Heuristiken und -Metriken* können z.B. Schwachstellen gesucht und Entwurfsrichtlinien überprüft werden [Ciu99].

Der Ablauf werkzeuggestützter Qualitätsuntersuchungen gliedert sich in die folgenden Schritte, für die insgesamt mindestens fünf Arbeitstage veranschlagt werden müssen:

1. *Zieldefinition:* Zuerst muss mit den Entwicklern zusammen geklärt werden, welche Ziele das Assessment verfolgen soll und welche Analysen überhaupt durchgeführt werden sollen.
2. *Faktenextraktion und Analyse:* Zu Beginn der Analyse steht die (oft von Problemen begleitete) Faktenextraktion, zudem können einfach und schnell durchzuführende Analysen erste Einschätzungen

über das System vermitteln.

3. *Zwischenergebnisse:* Die ersten Analyseergebnisse sollten möglichst bald, idealerweise am zweiten Tag des Assessments gemeinsam mit einem oder mehreren Entwicklern diskutiert werden.
4. *Verfeinerte Faktenextraktion und Analyse:* Die Faktenextraktion kann verfeinert und auf weitere Systemteile ausgeweitet werden.
5. *Abschlussworkshop:* Die Ergebnisse sollten möglichst vielen Entwicklern z.B. in Form einer Präsentation gezeigt werden, um sie zu diskutieren und zu bewerten.

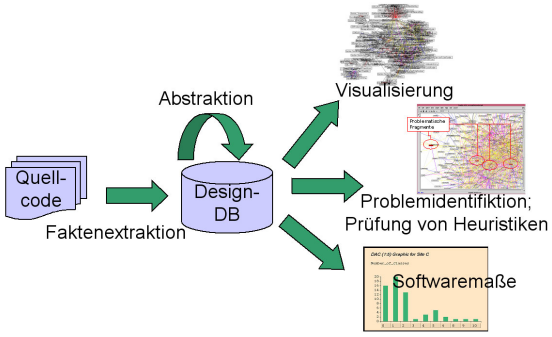

Abbildung 16.1. Techniken und Werkzeuge

16.3 Fallstudie und Ergebnisse

Bei der von uns untersuchten Fallstudie handelt es sich um ein komplett in C geschriebenes Datenbanksystem. Wir konzentrierten unsere Analysen auf den Server-Teil des Systems (Umfang: 1 Million LOC, 28 Subsystemen, 1347 Dateien).

Zuerst wurden die grobgranularen Einheiten des Systems – in diesem Fall durch Verzeichnisse gegebene Subsysteme – analysiert. Auf dieser Ebene wurden Architekturrichtlinien überprüft. Weiterhin konnten durch Kombination von Metriken wie LOC und McCabe die komplexesten Subsysteme identifiziert werden, die im weiteren Verlauf der Analysen besonders intensiv untersucht werden sollten. Einige der Subsysteme zeichnen sich durch eine besonders hohe Komplexität aus. Diese Subsysteme waren jedoch gut gekapselt (Analyse der Kopplung), so dass die hohe Komlexität kein Problem darstellt.

Auf der darunterliegenden Abstraktionsebene wurden in den besonders komplexen Subsystemen die vorhandenen Datentypen untersucht, indem Daten und zugehörige Funktionen zu abstrakten Datentypen gruppiert wurden. Dies war in diesem System über Heuristiken (Namenskonventionen, Parametertypen der Funktionen) möglich, da die Programmierer einen objektorientierten Programmierstil verwendet hatten. Auf diesen abstrakten Datentypen konnten dann ebenfalls Komplexitäts- und Kopplungsanalysen angewandt werden. In Abbildung 16.2 sind die Kopplungswerte für Paare von abstrakten Datentypen zu sehen.

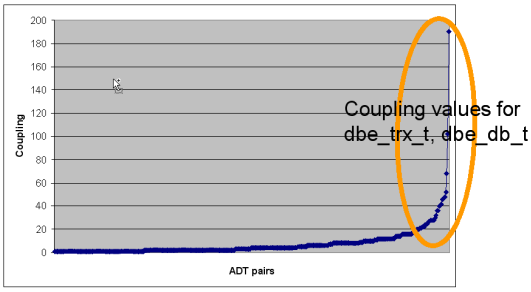

Abbildung 16.2. Kopplung abstrakter Datentypen

Manche der Datentypen sind äußerst komplex und zeichnen sich durch eine hohe Kopplung zu anderen Datentypen aus. Da sie aber zentrale und komplexe Konzepte wie Transaktionen repräsentieren, kann dies wahrscheinlich nicht vermieden werden.

Auf der untersten Abstraktionsebene können Analysen auf Methodenebene ausgeführt werden. Hierzu gehören z.B. die Suche nach unerwünschten Rekursionen oder Komplexitätsberechnungen. In dieser Fallstudie konnten – im Gegensatz zu Mozilla – keine langen Rekursionsketten gefunden werden.

16.4 Praxiserfahrungen

Im Verlaufe zahlreicher durchgeführter Softwareassessments konnten wir einige Erfahrungen sammeln, über die wir im Folgenden berichten wollen.

Hindernisse

Die Durchführung von Softwareassessments kann durch eine Reihe von Hindernissen erschwert werden.

- Zunächst behindert die Skepsis (“Hilfe, meine Leistung wird begutachtet!”) der Entwickler die Durchführung der Softwareassessments. Deren Know-How ist für erfolgreiche Assessments jedoch von großer Bedeutung (Zieldefinition, Interpretation der Analyseergebnisse). Abhilfe schaffen hier Kurzpräsentationen von Zwischenergebnissen, in denen den Entwicklern Potenzial und Nutzen der Untersuchungen demonstriert werden.
- Defizite auf Werkzeugebene: Werkzeuge zur Faktenextraktion kommen oft mit industriellem Code nicht zurecht. Sie scheitern häufig an unvollständigem Code, Makros und speziellen Programmiersprachendialekten. Zudem bereitet die große Menge an Rohdaten (Fakten) Probleme. Die meisten Analysewerkezeuge verarbeiten große Datenmengen nur unzuverlässig, zudem bieten nur wenige Werkzeuge Abstraktionsmechanismen zur Reduktion der Datenflut, die die Analysten auswerten müssen.

- Verfahren zur automatisierten Identifikation von Qualitätsmängeln können nur Hinweise auf potenzielle Schwachstellen liefern. Diese müssen dann von Analysten anhand des Quelltextes des Systems untersucht und bewertet werden. Schwierigkeiten bereiten in diesem Zusammenhang der Zeitmangel während der Assessments und ggf. die mangelnde Erfahrung der Analysten in der Anwendungsdomäne des Systems.

Nutzen

Insbesondere die Workshops zur Präsentation und Diskussion der Analyseergebnisse mit den Entwicklern zeigen jedoch, dass Softwareassessments von Auftraggebern und Entwicklern überwiegend positiv bewertet werden. Neben dem eigentlichen Ziel, objektive Aussagen über die innere Qualität der Software zu bekommen, lassen sich einige weitere positive Effekte identifizieren:

- Schon im Verlaufe der Abschlussworkshops beginnt in der Regel eine ausführliche Diskussion der Entwickler über Softwarequalität und über möglicherweise verbesserungsbedürftige Teile des Systems. Häufig entsteht so, motiviert durch die Analyseergebnisse, ein verbessertes Qualitätsbewusstsein.
- Entwickler interessiert oft der Vergleich mit anderen Softwaresystemen. Externe Analysten können hierzu leichter Aussagen machen, obgleich die Vergleichbarkeit von Softwaremaßen nur schwer zu gewährleisten ist: Anwendungsdomäne, Programmiersprache und -stil beeinflussen die Messwerte maßgeblich.

16.5 Schlussbemerkung

In diesem Papier haben wir gezeigt, dass mit werkzeuggestützten Qualitätsuntersuchungen quantitive Aussagen über die innere Qualität eines Softwaresystems gemacht werden können. Wir haben in diesem Papier Nutzen und Hindernisse solcher Assessments aufgezeigt. Hinsichtlich der Hindernisse erhoffen wir uns insbesondere für den Bereich der Werkzeugunterstützung in Zukunft einige Verbesserungen. Das FZI arbeitet mit verschiedenen Partnern im Rahmen der Forschungsprojekte COMPOBENCH und QBENCH an diesen Themen.

Literaturverzeichnis

[BC01] BAUER, MARKUS und OLIVER CIUPKE: *Objektorientierte Systeme unter der Lupe*. in *Proceedings of the third German Workshop on Software-Reengineering*. Universität Koblenz-Landau, Institut für Informatik, 2001.

[Ciu99] CIUPKE, OLIVER: *Automatic Detection of Design Problems in Object-Oriented Reengineering*. in *Technology of Object-Oriented Languages and Systems - TOOLS 30*. IEEE Computer Society, 1999.

[FAM99] FAMOOS PROJEKTTEAM: *The FAMOOS Object-Oriented Reengineering Handbook*. Technischer Bericht, Forschungszentrum Informatik, Karlsruhe, Software Composition Group, University of Berne, 1999.

---

17 (Linear) Algebra for Program Analysis

Markus Müller-Olm
FernUniversität Hagen, LG Praktische Informatik V, 58084 Hagen, Germany
On leave from Universität Dortmund, mmo@ls5.cs.uni-dortmund.de

Helmut Seidl
Universität Trier, FB 4-Informatik, 54286 Trier, Germany, seidl@uni-trier.de

This talk reported on onging research in which we use techniques from algebra and linear algebra to construct highly precise analysis routines for imperative programs or program fragments. We are interested in programs that work on variables taking values in some fixed ring or field $\mathbb{F}$, e.g., the integers or the rationals. Our analyses precisely interpret assignment statements with affine or polynomial right hand side and treat other assignment statements as well as guarded branching statements conservatively as non-deterministic statements. More specifically, we have constructed

1. an interprocedural analysis that determines for every program point of an affine program all valid affine relations[1];

[1] An *affine relation* is a condition of the form $a_0 + \sum_{i=1}^{n} a_i x_i = 0$, where $a_0, \ldots, a_n \in \mathbb{F}$ are constants from the underlying field and $x_1, \ldots, x_n$ are the program variables. A relation is *valid* at a program point, if it holds whenever control reaches that program point.