ROI-Modell für das Reengineering zur Optimierung technischer SW-Qualität

Dr. Frank Simon (SQS) *(Frank.Simon@sqs.de)*
Martin Studemund (Telekom) *(Martin.Studemund@telekom.de)*
Christian Rioth (T-Systems) *(Christian.Rioth@t-systems.de)*

1 Problemstellung

Die technische Qualität eines Software-Systems fokussiert diejenigen Eigenschaften, die von technischer Seite maßgeblich die Effektivität und die Effizienz der Software-Entwickler während der Erstellung und Pflege des Software-Systems bestimmen. Immer kürzer werdende Time-to-Market-Zyklen sowie die Notwendigkeit der Einarbeitung immer weit reichender Änderungsanforderungen an ein Software-System lassen die technische Qualität zunehmend als Schlüsselfaktor für erfolgreiche Projekte deutlich werden. Auch wenn die Sensibilisierung für gute Programmierung daher zunehmend wahrgenommen wird, leidet ihre konsequente Sicherstellung und Optimierung in Form von entsprechenden Reenginering-Aktivitäten aufgrund fehlender Möglichkeiten der Abschätzungen des möglichen Einsparpotentials. Obwohl die Zahl veröffentlichter Fallstudien, in denen für einen konkreten Projektkontext die *nach* Durchführung des Reengineering-Projektes erzielten Einsparungen beschrieben sind (z.B. 20% des gesamten Wartungsbudgets in [SiRi04]), sukzessive zunimmt, so genügt dies häufig nicht zur Motivation des initialen Hubs, der für ein Reengineering-Projekt notwendig ist, da das konkrete Einsparpotential *vor* Durchführung des Reengineerings unbekannt ist und z.B. unter der Annahme eines technisch perfekten Systems bei Null liegt.

2 Lösungsansatz

Mit dem in diesem Papier vorgeschlagenen und im Umfeld der Deutschen Telekom und T-Systems bereits eingesetzten ROI-(Return on Invest)-Modell wird die Möglichkeit geschaffen, auf Basis einer bestehenden Qualitäts-Ist-Analyse eines Software-Systems automatisch detaillierte und quantifizierte Hinweise im Vorfeld möglicher Reengineering-Tätigkeiten zu erhalten, welche Einsparpotentiale durch welche konkreten Reengineering-Tätigkeiten zur Optimierung der technischen Qualität gehoben werden können. Damit ist es dem Projekt-Management möglich, konkrete Aktivitäten, die ausschließlich technische Änderungen bewirken und daher in der Vergangenheit häufig eine niedrigere Priorität hatten, auch unter wirtschaftlichen Gesichtspunkten zu begründen und zu planen, um damit ein messbares Einsparpotential zu heben.

Die Grundidee dieses ROI-Modells besteht aus der kausalen Verknüpfung der beiden folgenden Modelle zu einem Gesamtmodell:

- Spezielle Qualitätsmodelle erlauben das Herunterbrechen allgemeiner Qualitätsanforderungen an Softwaresysteme in Form von Qualitätseigenschaften bis auf die Ebene konkret messbarer Code-Merkmale.
- Empirische Modelle der Wartungsphase erlauben eine Klassifizierung der anfallenden Tätigkeiten und eine prozentuale Verteilung bzgl. der jeweilig verbrauchten Aufwände.

Die Verknüpfung selbst geschieht unter Verwendung von drei bzgl. der ROI-Höhe defensiven Hypothesen, so dass ein robustes ROI-Modell als Ausgangsbasis weiterer Anpassung aufgebaut werden kann.

2.1 Bidirektionale Qualitätsmodelle

Die bidirektionalen Qualitätsmodelle sind ein Ergebnis des Forschungsprojektes QBench (vgl. [SeSiMo05] und ermöglichen als Erweiterung bisheriger Qualitätsmodelle nicht nur die unidirektionale, stufenweise Verfeinerung von geforderten Qualitätseigenschaften bis auf die Ebene messbarer Merkmale, sondern erlauben ebenfalls buttom-up die Auswirkungen existierender, durch verfügbare Werkzeuge überprüfbarer und messbarer Code-Merkmale auf höherwertige Qualitätseigenschaften zu explizieren. Beide Hauptstoßrichtungen (bidirektional) werden in mehreren Qualitätsindikatoren zusammengefasst, die als verbindendes Element zwischen technischen Code-Merkmalen und allgemeinen Qualitätseigenschaften folgende Informationen tragen:

- Gewichtete Abhängigkeiten zu Qualitätseigenschaften: Jeder Qualitätsindikator hat unterschiedlich starken Einfluss auf die Qualitätseigenschaften. Diese Eigenschaften sollten bereits möglichst weit operationalisiert sein, um die Gewichtung nachvollziehbar gestalten zu können. Eine solche Möglichkeit bietet die ISO9126 mit der Definition von Änderbarkeit (vgl. [ISO9126]). Qualitätsindikatoren gewichten demnach ihren Einfluss auf die Eigenschaften Analysierbarkeit, Prüfbarkeit, Modifizierbarkeit und Stabilität. Ein typisches Beispiel stellt der Indikator „Defensive Programmierung“ dar, der demnach hauptsächlich Auswirkungen auf die Stabilität besitzt.
- Menge von Qualitätsmerkmalen bestehend aus Regeln und Metriken, die einen negativen Einfluss auf den Qualitätsindikator haben können. Für die Automatisierbarkeit ist wichtig, sich auf solche Merkmale zu beschränken, die

grundsätzlich berechenbar sind und für die eine Werkzeuginfrastruktur zur Verfügung steht. Ein typisches Merkmal für den Qualitätsindikator „Defensive Programmierung“ betrachtet leere Exception-Handler, die die Stabilität des Systems negativ beeinflussen können.

Die praktische Erfahrung beim Einsatz derartiger Qualitätsindikatoren hat gezeigt, dass nur die Qualitätsmerkmale von technischen Details abhängig sind. Aktuell wird daher eine fixe Menge von Qualitätsindikatoren für die Sprachen C/C++, JAVA und Cobol jeweils angepasst verwendet.

2.2 Empirische Wartungsmodelle

Für ein ROI-Modell ist es essentiell, Transparenz bzgl. derjenigen Aufwände und damit Kosten zu erhalten, die im Fokus der Optimierung durch Reengineering-Maßnahmen stehen. Eine insbesondere im Bereich des Reengineerings häufig verwendete Klassifizierung mit einer Verteilung der Aufwände (vgl. [MaBe96]) geht davon aus, dass während der Wartung 47% des Gesamt-Aufwands für das reine Code-Verstehen, 28% für das Re-Testen, 19% für die Codierung der Änderung sowie 6% für die Re-Dokumentation benötigt werden. Da eine Optimierung der technischen Qualität sich positiv auf das Code-Verstehen (z.B. durch eine bessere Kommentierung), das Re-Testen (z.B. durch eine bessere Modularität) und die Codierung (z.B. durch bessere Erweiterbarkeit) auswirkt, stehen 94% des Wartungsbudgets im Fokus der Optimierung.

In einer weiteren Verfeinerung werden die Teilaktivitäten in eine kausale Relation zu den verfeinerten Qualitätseigenschaften gebracht. Diese Abschätzung geht für die Tätigkeit „Code-Verstehen“ z.B. davon aus, dass diese zu 2/3 durch die Analysierbarkeit, die explizit durch die Eignung definiert ist, im Code gesuchte Punkte diagnostizieren zu können (vgl. [ISO9126]), und zu 1/3 durch die Stabilität, da instabile Konstrukte i.d.R. auch missverständlicher sind, bestimmt wird.

2.3 Drei Kernhypothesen

Die mit den empirischen Modellen erstmals gegebene Möglichkeit, konkrete Aufwände für Aktivitäten (z.B. 47% Code-Verstehen) unter Berücksichtigung von Gewichtungen auf konkrete Code-Merkmale abzubilden und damit Veränderungen der technischen Qualität (und damit der Code-Merkmale) auf ihre monetäre Wirkung im Voraus abschätzen zu können, bedarf noch dreier wichtiger, numerisch diskutabler Hypothesen:

- Die Qualitätseigenschaften sind nur zu 50% durch die Qualitätsindikatoren bestimmt. Die Analysierbarkeit hängt z.B. zusätzlich von den verwendeten Analysewerkzeugen sowie den individuellen Entwickler-Fähigkeiten ab.
- Die Qualitätsindikatoren sind nur zu 50% durch die betrachteten Code-Merkmale bestimmt. So kann der Qualitätsindikator Kommentierung z.B. durch eine Vielzahl von Metriken und Analysen bestimmt werden (z.B. JAVADoc-Konformität); Aussagen über die Semantik sind auf Merkmalsebene jedoch nicht möglich.
- Die Priorisierung der zu optimierenden Merkmale eines Systems basiert auf einem Benchmarking mit anderen ähnlichen Systemen. Die SQS besitzt für die drei Sprachen C/C++, JAVA und Cobol ein derartiges Benchmark-Repository, in dem insgesamt fast 100 Projekte abgelegt sind. Die dritte Hypothese besagt, dass ein neues Projekt bzgl. derjenigen Code-Merkmale die größten Potentiale birgt, bei denen das System schlechter als der Durchschnitt ähnlicher Systeme ist. In einem solchen Fall kann der vollständige Kostenbereich, der für das Merkmal metrikbasiert feststellbar ist, nach einer Optimierung eingespart werden.

3 Erfahrungen

Das hier vorgeschlagene ROI-Modell ist Grundlage vieler aktueller Projekte zwischen der Telekom und T-Systems. Die jeweiligen Code-Merkmalsausprägungen können vollständig durch eine Code-Analyseumgebung ermittelt werden. Steht für ein Merkmal das Einsparpotential, das mittels eines Excel-Sheets vollständig automatisch berechnet wird, fest (s.o.), so kann nach Abschätzung des Aufwands für das Reengineering eine maximal transparente Entscheidung für oder wider die Maßnahme getroffen werden. In der Praxis hat sich für ein Projekt z.B. ein grundsätzliches Einsparpotential von ca. 10% ergeben. Nach Streichen derjenigen Merkmale, deren Aufwände zur Behebung über der erwarteten Reduzierung lagen, ergab sich eine Einsparung von knapp 7% mit einem einjährigen positiven ROI. Die durch dieses ROI-Modell erzeugte Transparenz ermöglichte die Verständigung auf diese Zieleinsparung, die aktuell in das System eingearbeitet wird.

4 Referenzen

[MaBe96] G. Marliss, M. Ben-Menachem: “*Slaying the Software Dragon: A Practical Guide to Software Quality*”, Thomson Publishing, 1997

[ISO9126] Internationaler Standard ISO/IEC 9126: „*Software product evaluation -- Quality characteristics and guidelines for their use*“, International Organization for Standardization

[SeSiMo05] Olaf Seng, Frank Simon, Thomas Mohaupt: „*Bi-directional Quality Models: How to Assess Technical Quality with Existing Tool Infrastructure*“, in Proceedings der ICSTest2005, Düsseldorf 2005

[SiRi04] Frank Simon, Ute Richter: "*Mit Code-Metriken Wartungskosten senken: Controlling technischer Qualität*“, Metrikon 2004, in Proceedings des internationalen Workshops für Software-Metriken und der DASMA Software-Metrik-Konferenz Metrikon 2004