

QBench¹ – Sicherung der inneren Qualität bei der Evolution objektorientierter Softwaresysteme

Christoph Andriessens
FZI Forschungszentrum Informatik
Forschungsbereich Programmstrukturen
Haid-und-Neu-Str. 10-14
76131 Karlsruhe

Kurzfassung

Die Kosten für Softwareevolution, auf die sich im Vergleich zur Ersterstellung die größten Kosten verteilen, werden maßgeblich durch die innere Qualität der Software bestimmt – die Softwarequalität, die vom Softwareingenieur in Form technischer Strukturen und Merkmale wahrgenommen wird. Das Projekt QBench wird die nötigen Methoden und Werkzeuge bereitstellen, um die innere Qualität objektorientierter Software im Rahmen eines ganzheitlichen und integrierten Ansatzes konstruktions- und evolutionsbegleitend zu beurteilen und zu verbessern. Im Fokus liegen dabei besonders die konzeptionelle Lücke zwischen Problemerkennung und -behebung sowie der Nachweis, dass sich die Sicherung innerer Qualität betriebswirtschaftlich lohnt. QBench hat die Schaffung von Grundlagen, die für den folgenden Durchstich benötigt werden, im Sommer 2004 abgeschlossen. Dieser Beitrag gibt einen Überblick über das Projekt und die bisher erzielten Ergebnisse.

1. Einleitung und Vorstellung des Themenkomplexes

Ausufernde Softwareprojekte, die Termine nicht einhalten können und Budgets sprengen sowie darauf folgend hohe Wartungskosten sind ständige Begleiter in der Softwareentwicklung. Seit 1968, als das Nato Science Committee den Begriff „Softwarekrise“ prägte, belegen ständig neue Studien diese Problematik: Im Jahr 2002 kam die Standish Group in dem jährlich aktualisierten „Chaos-Report“ zu dem Ergebnis, dass nur 34% aller Softwareprojekte erfolgreich waren. Der Trend dieser Werte stagniert über die letzten Jahre, d.h. auch neue Techniken und Prozesse lösen das Problem nur unzureichend.

Eine Ursache für dieses Problem ist, dass Softwareentwicklung kein linearer Prozess ist, der dem schon sprichwörtlichen Wasserfall folgt. Software wird evolutionär entwickelt: Vorhandene Software wird an neue Anforderungen angepasst oder in neue Systeme integriert. In der Gesamtsicht verteilen sich die weitaus größten Kosten auf Softwareevolution: In den achtziger Jahren war Wartung, Pflege und Betrieb von Software bereits doppelt so teuer wie die initiale Entwicklung. In den neunziger Jahren wuchsen diese Kosten nochmals kräftig und werden im Jahr 2000 teilweise bei 90 Prozent der Gesamtkosten gesehen.

Die Kosten für Softwareevolution werden maßgeblich durch die *innere Qualität*² der Software bestimmt: Darunter verstehen wir die Softwarequalität, die vom Softwareingenieur

¹ <http://www.qbench.de>

wahrgenommen wird und die sich in Kriterien wie Erweiterbarkeit, Wiederverwendbarkeit, Verständlichkeit, Modifizierbarkeit, Testbarkeit, Kompatibilität oder Anpassbarkeit an verschiedene Umgebungen ausdrückt. Wir unterscheiden sie von der *äußeren Qualität*, die die Softwarequalität beschreibt, die vom Benutzer der Software wahrgenommen wird. Mit ihren Eigenschaften bestimmt nun innere Qualität – oder ihr Fehlen – direkt, wie leicht die Software an neue Anforderungen und Systemumgebungen angepasst, wiederverwendet und weiterentwickelt werden kann.

Um die hohen Evolutionskosten zu senken, ist es also erforderlich, stets eine hohe innere Qualität eines Softwaresystems bei der Entwicklung zu gewährleisten. Um Qualität zu sichern, muss man Aussagen zum aktuellen qualitativen Zustand machen können. Hier setzt QBench an: QBench wird die nötigen Methoden und Werkzeuge bereitstellen, um die innere Qualität objektorientierter Software im Rahmen eines ganzheitlichen³ und integrierten⁴ Ansatzes konstruktions- und evolutionsbegleitend zu beurteilen und zu verbessern. Die Werkzeuge sollen eine weitestgehend automatisierte Sicherung und Verbesserung innerer Qualität erlauben. Um sie in existierenden Softwareentwicklungsprozessen der industriellen Praxis einsetzen zu können, ist außerdem die Erstellung spezieller Leitfäden geplant.

Eine der besonderen Herausforderungen, denen sich QBench gegenüber sieht, ist die konzeptionelle Lücke zwischen Verfahren zur Problemidentifikation und Verfahren zur Problembhebung, z.B. Quelltexttransformationsverfahren. Genauer besteht die Schwierigkeit darin, wie von einem gegebenen Entwurfsproblem ausgehend automatisch eine Sequenz von Quelltexttransformationen abgeleitet werden kann, die das Problem beseitigen und gleichzeitig zu verbesserten vorgegebenen Qualitätseigenschaften führen. Liegen mehrere Entwurfsprobleme vor, muss auch hier eine Folge von Transformationen gefunden werden, welche die vorgegebenen Qualitätseigenschaften insgesamt verbessern. Der Beitrag zur Schließung dieser Lücke eingebettet in einen ganzheitlichen Gesamtansatz, der Modelle und Quelltextabstraktionen zulässt, ist ein Teil des besonderen innovativen Gehalts von QBench. Hieran schließen sich Formalisierung und Klassifizierung von Problemstrukturen und Lösungsstrategien an. Gerade letztere sind beispielsweise in Form der Fowlerschen Refactorings weit bekannt, aber wenig formalisiert.

Zur Sicherstellung praxisorientierter Ergebnisse werden die Ergebnisse anhand vier industrieller Fallstudien ständig evaluiert und auch in die industrielle Praxis transferiert. Hier ist gleichzeitig ein weiterer Teil des besonders innovativen Gehalts von QBench zu sehen: Durch eine Erfassung der Qualitätsdaten der vier Fallstudien zu Projektbeginn und eine gleichzeitig einsetzende Aufzeichnung relevanter Entwicklungsdaten bei allen Industriepartnern soll in QBench der Nachweis geführt werden, dass sich die Sicherung innerer Qualität auch betriebswirtschaftlich lohnt. Damit würde die Vermutung, dass innere Qualität einen Einfluss auf äußere Qualität hat (im Sinne des bekannten „Axioms des Software-Engineerings“), aus wirtschaftlicher Sicht gestärkt.

2. Projektstatus

In diesem Abschnitt beschreiben wir zunächst die beiden zentralen inhaltlichen Konzepte, bevor wir auf den Projektrahmen und die bisher erzielten Ergebnisse eingehen.

² auch *interne* Qualität (im Gegensatz zu *externer* Qualität) genannt

³ Ganzheitlich: Alle relevanten Softwareentwicklungsartefakte (Quelltexte und darauf abbildbare Abstraktionen wie Modelle, Konfigurationen etc.) werden berücksichtigt.

⁴ Integriert: Der Ansatz unterstützt durchgängig alle Schritte von der Problemerkennung bis zur Behebung.

Zentrale inhaltliche Konzepte

Erkennungs- und Lösungsstrategien sind die zentralen inhaltlichen Konzepte im wissenschaftlichen Herz des Projektes.

- *Erkennungsstrategien* dienen der automatisierten Identifikation von im Sinne innerer Qualität mangelhafter Softwaremerkmale. Sie sollen in einem erweiterbaren Katalog abgelegt werden, der von Werkzeugen zur Problemerkennung genutzt werden kann. Grundlage für ihre Entwicklung sind existierende und evtl. zu entwickelnde Metriken und Heuristiken. Eine wichtige Fragestellung bei der Entwicklung der Erkennungsstrategien ist, wie das Rauschen vermindert werden kann, das normalerweise durch Metriken und Heuristiken erzeugt wird: Eine meist kleinere Zahl wichtiger Ergebnisse geht in einer Flut eher unwichtiger Ergebnisse (häufig auch „Falsch-Positiver“ Ergebnisse) unter.
- *Lösungsstrategien* bilden (semi-)automatisch eine erkannte mangelhafte Struktur kontextabhängig auf eine Lösungsstruktur für das erkannte Qualitätsproblem ab. Diese Lösungsstruktur kann dann (semi-)automatisch durch Programmtransformation umgesetzt werden. Auch die Lösungsstrategien sollen in einem erweiterbaren Katalog abgelegt werden, der von Werkzeugen zur Problembehebung genutzt werden kann. Ein wichtiges Stichwort bei der Problembehebung ist natürlich die Verhaltensbewahrung, die sicherzustellen ist.

Projektrahmen

Das Projekt ist auf einen Zeitraum von 30 Monaten ausgelegt. Es verfolgt einen iterativen Ansatz: Die 30 Monate sind in vier Iterationen aufgeteilt, von denen jede einem speziellen Ziel untergeordnet ist und in einer Rückbetrachtung Anforderungen für die jeweils nächste Iteration definiert.

1. Die erste Iteration setzt das Fundament: Grundlegende Elemente sind dabei eine Analyse der Anforderungen in der industriellen Praxis sowie eine Zusammenstellung des Standes der Technik, die existierende Lücken und Anpassungsbedarf untersucht und hervorhebt. Des Weiteren wurde der schon angesprochene Wirtschaftlichkeitsnachweis definiert und eingerichtet. Ein erstes Assessment der Fallstudien schafft ein tieferes Verständnis für die Herausforderungen, die die Sicherung innerer Qualität an die Industriepartner in ihrer jeweiligen Domäne stellt und ermöglicht eine spätere „Vorher-Nachher“-Qualitätsanalyse. Gleichzeitig werden mit einem initialen Qualitätsmodell und dem Systemmetamodell formale Grundlagen geschaffen, die dann in einer Sammlung und Klassifizierung von Problemstrukturen zur Realisierung von Erkennungsstrategien aufgegriffen werden.
2. Die zweite Iteration wird einen Durchstich erzielen: Auf der breiten Basis von Ergebnissen der ersten Iteration und den Anforderungen, die diese ergeben hat, soll erstmals über die Definition von Lösungsstrategien ein konzeptueller Durchbruch erzielt werden.
3. „In die Breite tragen“ ist das Thema der dritten Iteration: Die Erkenntnisse, die beim Durchstich gewonnen wurden, sollen nun genutzt werden, um den Ansatz in die Breite tragen zu können.
4. Die Stabilisierung der zuvor gewonnenen Ergebnisse ist schließlich Zweck der vierten Iteration.

Zum Ende jeder Iteration werden außerdem wieder die Qualitätsdaten der Fallstudien aufgenommen, um eine Entwicklung erkennen und einen Projekterfolg nachweisen zu können.

Der Verbund setzt sich folgendermaßen zusammen:

- Zwei Forschungseinrichtungen:
 - FZI Forschungszentrum Informatik, Karlsruhe (Projektkoordinator)

- Fraunhofer FIRST SWQlab, Cottbus
- Ein Beratungshaus: SQS Software Quality Systems AG, Köln
- Vier Industriepartner:
 - CAS Software AG, Karlsruhe (Fallstudie: CRM-System / Groupware / Intranetportal)
 - infor business solutions AG, Friedrichsthal (Fallstudie: ERP-System)
 - LogControl GmbH, Pforzheim (Fallstudie: Client-/Serversystem zur Lagerverwaltung)
 - PTV AG, Karlsruhe (Fallstudie: Plattform zur integrierten Behandlung transportlogistischer Problemstellungen auf Basis flexibler Workflows und Standardkomponenten)
- Zwei assoziierte Partner, die das Projekt unterstützen:
 - Software Tomography GmbH, Cottbus
 - Harman/Becker Automotive Systems GmbH, Karlsbad

Stand der Arbeiten

Das Projekt beendet die erste Iteration im Juli 2004. Die bisher erzielten Ergebnisse unterteilen wir in grundlegende, Konzept und Evaluierung des zu entwickelnden Ansatzes direkt betreffende sowie empirische Ergebnisse.

Grundlegendes

- Das zentrale Anliegen des Arbeitspaketes „Anforderungsanalyse“ war neben einer Präzisierung der Ziele für die weiteren Arbeitspakete die Analyse und Dokumentation der Anforderungen der Projektpartner. Um die Relevanz der Ergebnisse von QBench für die industrielle Praxis zu gewährleisten, standen ganz besonders die Anforderungen der Industriepartner im Vordergrund, die alle als Erstanwender für die Ergebnisse des Projektes QBench vorgesehen sind. Im Ergebnisdokument wurden die Anforderungen an den zu entwickelnden ganzheitlichen Ansatz zur Sicherung der inneren Qualität von Software gesammelt und in „Inhaltliche Anforderungen“, „Organisatorische Anforderungen“ sowie „Nicht berücksichtigte Anforderungen“ klassifiziert. Zu den festgehaltenen inhaltlichen Anforderungen gehören beispielsweise Verhaltensbewahrung durch Qualitätsverbesserung, prinzipielle Unterstützung der Sprachen C#, C++, Delphi und Java sowie Verbesserung der Schnittstellenstabilität. Ein Teil der organisatorischen Anforderungen ist die Durchführung des Wirtschaftlichkeitsnachweis (siehe unten). Mit den nicht berücksichtigten Anforderungen wurde unter anderem festgestellt, dass Performanceverbesserung oder die Untersuchung von Datenbankfunktionalitäten (wie Stored Procedures) kein Thema für QBench sind.
- Eine wichtige Aufgabe zu Beginn des Projektes war es, den relevanten Stand der Technik im Bereich der Sicherung innerer Qualität von Software zusammenzustellen. Dabei sollten auch die gefundenen Lücken klar herausgearbeitet werden. Ziel war es zum einen, das zusammengetragene Wissen kompakt und schnell im Verbund verbreiten zu können. Zum anderen sollten die Stärken und Schwächen eventuell bestehender Ansätze dokumentiert werden, um effizient in der weiteren Arbeit darauf aufbauen zu können. Das Ergebnis zeigt, dass das von QBench verfolgte Ziel, einen ganzheitlichen Ansatz zur konstruktions- und evolutionsbegleitenden Sicherung der inneren Qualität von objektorientierter Software zu entwickeln und einzusetzen, nach wie vor aktuell und ungelöst ist. Das gilt insbesondere für die Verbindung zwischen Problemerkennung und Problemlösung. Das Ergebnisdokument ist öffentlich.

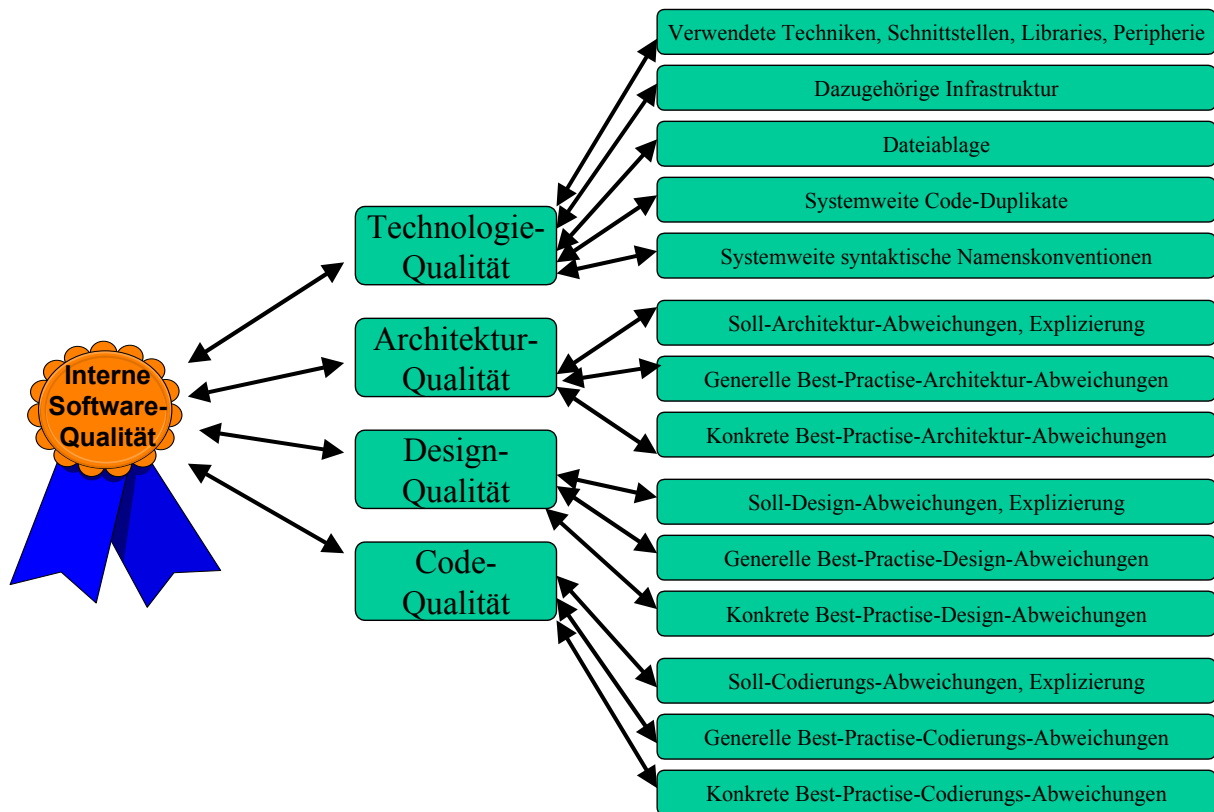


Abbildung 1: Das initiale Qualitätsmodell von QBench

Konzept und Evaluierung

- Um den Qualitätsbegriff für das Projekt zu operationalisieren, wurde ein initiales Qualitätsmodell entwickelt. Dabei hat sich QBench bewusst nicht an das bekannte, bereits bestehende Softwarequalitätsmodell nach der deutschen DIN 66272 bzw. der internationalen ISO 9126 Norm gehalten. Die Operationalisierung von Qualität findet dort auf einer relativ hohen bzw. abstrakten Ebene statt. Trotz einer übersichtlichen Modellstruktur fehlt damit eine echte Konkretisierung von Qualität, die direkt im Tagesgeschäft umsetzbar und für die Qualitätsbestimmung von Fallstudien anwendbar gewesen wäre. Die Definition einer festen Hierarchie von Qualitätsmerkmalen erschwert zudem die Anpassbarkeit des Qualitätsmodells. Da jedoch in der Praxis die fachlichen Anforderungen an ein Softwareprodukt stark variieren, ist es oft erforderlich, eine individuelle Gewichtung einzelner Qualitätsmerkmale vornehmen zu können. Für den täglichen Gebrauch werden Gewichtungen einzelner Merkmale z.B. auch benötigt, um in Hinsicht auf Zeit, Kosten und Qualität eine Priorisierung der Qualitätsmerkmale insbesondere aus betriebswirtschaftlicher Sicht vornehmen zu können. Die fehlende Anpassbarkeit des Qualitätsmodells nach ISO 9126 ist ein wesentlicher Grund dafür, dass dieses Modell nicht weiter im Projekt QBench verwendet wird. Auch wenn aus Sicht von QBench Projektspezifika einen ganz wesentlichen Einfluss für spezifische Qualitätsmodelle haben, wird für das erste initiale Qualitätsassessment dennoch versucht, eine einheitliche Qualitätsmodellvorstellung zu entwerfen. Die Basis des initialen Qualitätsmodells für QBench (Abbildung 1: Das initiale Qualitätsmodell von QBench) ist die Zerlegung eines Softwaresystems in vier verschiedene Schichten, die jeweils Abstraktionsebenen eines Softwareprodukts repräsentieren, nämlich Technologie, Architektur, Design und Code. Jeder Abstraktionsebene werden nun spezifi-

sche Qualitätsmerkmale zugeordnet: Auf der Technologieebene wird dazu eine High-Level-Sicht auf das gesamte System eingenommen. Betrachtet werden hier Aspekte, die Auswirkungen auf das System als Ganzes haben, und die – direkt oder indirekt – entscheidend für die Qualität des fertigen Produktes sind. Auf den Ebenen Architektur, Design und Code lassen sich die Qualitätsmerkmale in Soll-Vorschriften bzw. deren explizites Festhalten, die Bestimmung von Abweichungen dagegen sowie Abweichungen gegen generelle und konkrete, projektspezifische Best-Practises klassifizieren.

- Zur Entwicklung von Erkennungsstrategien wurden bisher über 70 sprachübergreifende Problemmuster gesammelt, benannt und klassifiziert sowie mit den verschiedenen Ebenen des Qualitätsmodells verbunden. Diese Problemmuster verfeinern das Qualitätsmodell.

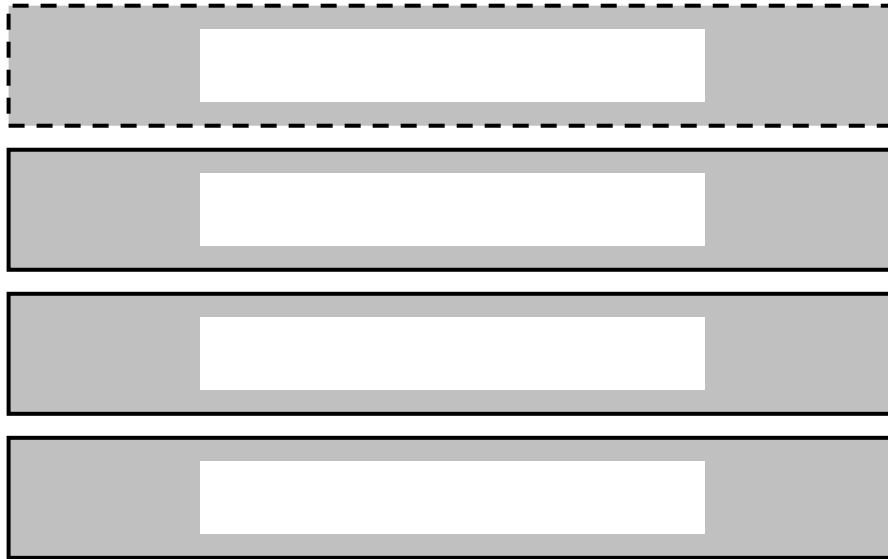


Abbildung 2: Das Systemmetamodell von QBench – schematischer Aufbau

- Das dreistufige, sprachübergreifende Systemmetamodell (Abbildung 2: Das Systemmetamodell von QBench – schematischer Aufbau) beschreibt die Modelle, die wiederum die mit den Werkzeugen von QBench zu untersuchenden Softwaresystemen beschreiben. Es besteht aus Quelltext auf der untersten Ebene, darüber einer Strukturbaumsicht und darauf aufbauend einem Analyse- und Transformationsmodell mit jeweils einem sprachübergreifenden Teil sowie einer sprachabhängigen Abbildung. Das Systemmetamodell ist mit seinem Analyseteil und einer Vorlage für das Transformationsmodell als Basis für die weitere Entwicklung verfügbar. Es wird im Weiteren auch noch um eine vierte Schicht, die Abstraktionen entsprechend Spezifikationen oder im Quelltext nur implizit vorhandenen Entwurfskonzepten beschreibt, erweitert werden. Damit wird berücksichtigt, dass generative Techniken (wie z.B. der Model Driven Architecture (MDA)-Ansatz der OMG) zunehmend wichtiger werden. Außerdem erlaubt es beispielsweise die Bewertung mehrerer Quelltextelemente zusammenfassenden Designkonzepte.
- Seit Projektbeginn laufen bereits kontinuierlich Arbeiten zur Entwicklung und dem Ausbau der Infrastruktur wie
 - einer Implementierung des Systemmetamodells zum Aufbau konkreter, persistenter Systemmodelle aus Quelltext,

- Werkzeugen zur Problemerkennung und
- Softwaretransformation.

Empirie

- Zur Definition der industriellen Fallstudien wurden Steckbriefe erstellt, die die wesentlichen bekannten Fakten der Fallstudie dokumentieren. Dazu gehören neben einer Beschreibung der Fallstudie technische Parameter, die wesentlichen Einfluss auf eine Qualitätsbetrachtung haben (generierter Code, Bibliotheken etc.). Darüber hinaus werden der organisatorische Kontext, etwaige Qualitätssicherungsmaßnahmen und Qualitätsziele beschrieben. Diese Steckbriefe werden im weiteren Verlauf des Projektes erweitert.
- Einer Erfassung der Qualitätsdaten und auch der Untersuchung der Qualität im Projektverlauf dienen Assessments der Fallstudien. Diese werden im Wechsel zwischen jeweils zwei Assessoren vorgenommen, um eine breitere Abdeckung der Fallstudien zu ermöglichen. Die dabei gewonnenen Erkenntnisse werden mit Entwicklern in kurzen Workshops diskutiert und dokumentiert, um etwaigen Anpassungsbedarf am Qualitätsmodell frühzeitig erkennen und für Folgeassessments berücksichtigen zu können. Die initialen Assessments sind abgeschlossen. Faktoren, die die innere Qualität der Fallstudien außerhalb von QBench beeinflussen könnten (wie neue Produktstrategien, Schulungen, neue strategische Managementziele), werden gegenwärtig identifiziert und können so dokumentiert werden.
- Für den Nachweis des betriebswirtschaftlichen Nutzens der Sicherung innerer Qualität wurde der Wirtschaftlichkeitsnachweis definiert und eingerichtet. Dazu wurden drei Hypothesen formuliert, die durch eine bei den Industriepartnern in den Fallstudien jeweils eingerichtete Sammlung entsprechender Daten wie Bearbeitungszeiten etc. und deren Auswertung nachgewiesen werden sollen.
 - Basis der *Hypothese zur internen Nutzungsanalyse* ist die Annahme, dass eine verbesserte innere Qualität zu einer stabileren Struktur und damit zu besserer Wartbarkeit einer Software führt. Wenn also die Wartbarkeit des Systems deutlich gestiegen ist, wird die Bearbeitungszeit für einzelne Wartungsaktivitäten vermutlich verringert. Hierdurch ist es möglich, die Wartungszyklen allgemein zu verkürzen, d.h. die Software schnell auf den neuesten Stand zu bringen und wieder produktiv einsetzen zu können. Zusammengefasst: Verbesserte innere Qualität verringert Bearbeitungszeiten und verkürzt Wartungszyklen.
 - Basis der *Hypothese zur externen Nutzungsanalyse* ist die Annahme, dass eine verbesserte interne Qualität die Wartbarkeit der betrachteten Software deutlich erhöht. Ist ein System leichter verständlich und stabiler, so ist zu erwarten, dass während der Implementierung von neuen oder geänderten Funktionalitäten weniger interne Fehler entstehen als zuvor. Sinkt also trotz unveränderter interner Maßnahmen zur Qualitätssicherung die absolute Anzahl an identifizierten Fehlern, können nicht nur Aufwände für umfangreiche Tests, Fehlersuche und Fehlerbehebung gespart werden. Eine geringere Fehleranzahl macht es auch wahrscheinlich, dass weniger externe Fehler durch Kunden identifiziert werden, da viele interne Fehler eine direkte Abhängigkeit zu externen Fehlern besitzen. Dies steigert die Kundenzufriedenheit, da die Software bei ihren Bedienern als stabil gilt. Insofern wäre ein direkter Zusammenhang zwischen interner und externer Qualität nachweisbar. Zusammengefasst: Verbesserte innere Qualität verringert die Anzahl interner und damit die Anzahl externer Fehler.
 - Basis der *Hypothese zur Wirtschaftlichkeitsanalyse* ist, dass das durch kürzere Wartungszyklen, geringere Fehlerrate und Imagegewinn beim Kunden erzielte Nutzen

größer ist als die für die Verbesserung der internen Qualität aufgewendeten Kosten und sich damit eine Qualitätsverbesserung auch aus betriebswirtschaftlicher Sicht lohnt. Grundlage für eine Bestätigung dieser Hypothese ist, dass mindestens eine der beiden zuvor beschriebenen Hypothesen bestätigt werden kann.

3. Erfahrungen, Bewertungen

In diesem Abschnitt erläutern wir Erfahrungen und Probleme außerhalb der Projektergebnisse, auf die wir im Projekt gestoßen sind.

Zu den typischen Probleme in der industriellen Softwareentwicklung, die sich in den Fallstudien zeigen, gehört Heterogenität der eingesetzten Technologien: Beispielsweise werden verschiedene Sprachen (C#, C++, Delphi, Java und andere) in einem System eingesetzt. Oder J2EE und .NET finden sich kombiniert in einem System wieder.

Eine weitere Schwierigkeit sind kundenspezifische Anpassungen von Softwaresystemen, die sauber vom Rest des Systems getrennt werden müssen und im Laufe der Zeit zu harten und vor allen Dingen komplizierten Randbedingungen für Änderung und Weiterentwicklung des Systems heranwachsen können.

Die Qualitätssicherung ist besonders für Firmen ohne eigene Qualitätssicherungsabteilung eine Herausforderung: Hier muss die Qualitätssicherung vom Entwickler oder Projektleiter selber betrieben werden, denen hierzu die Unterstützung durch geeignete, in die Entwicklungsumgebung integrierte Werkzeuge fehlt.

Generative Techniken halten immer stärkeren Einzug in der Softwareentwicklung: Dies reicht von der eher einfachere Erzeugung von Datenbankzugriffsschichten oder Interfaces bis hin zu vollständigen MDA-Ansätzen. Zu den QBench-Fallstudien gehört eine Anwendung, deren Quelltext zu ungefähr 85% aus Spezifikationen generiert wird, und keine der Fallstudien ist frei von generiertem Code. Bei der Untersuchung innerer Qualität muss dies berücksichtigt werden, da die Qualität des generierten Codes eine andere Relevanz besitzt als direkt manuell geschriebener Code. Insbesondere stellt sich hier die Frage nach einer geeigneten Bewertung der Spezifikationen, die Grundlage der Codeerzeugung sind. Ebenso ist die Frage interessant, ob manuell nachimplementierter Code in geschützten Bereichen generierten Codes genauso bewertet werden kann wie manuell implementierter Code, der keinen Randbedingungen durch Generate unterliegt.

Eine notwendige Sammlung von Daten wie beispielsweise Bearbeitungszeiten, die zur Beurteilung von Qualitätssicherungsmaßnahmen wie der in QBench zu entwickelnden Methoden und Werkzeuge oder für die Wirtschaftlichkeitsbetrachtung der Sicherung innerer Qualität (vgl. den Wirtschaftlichkeitsnachweis) unabdingbar ist, stößt schnell auf Bedenken der Entwickler und eines evtl. vorhandenen Betriebsrates. Zum einen gilt es hier natürlich, durch Darlegung der Projektziele zu überzeugen. Gleichzeitig kann zum anderen versucht werden, entsprechende Daten so zu erheben, dass entsprechende Bedenken zerstreut und trotzdem sinnvolle Beurteilungen von Qualitätssicherungsmaßnahmen vorgenommen werden können.

4. Ausblick

Zur Jahresmitte ist die erste Iteration beendet. Sie wird in einer Rückbetrachtung mit ihren Ergebnissen Anforderungen für die zweite Iteration definieren. Der bisherige Projektverlauf bestätigt die bisherige Planung und Ausrichtung des Projektes. Insbesondere die Aufnahme und Katalogisierung der Vielzahl bisher existierender Techniken und Konzepte hat eine sehr gute Grundlage für höherwertige Ergebnisse geliefert. Die Einbindung in das QBench-

Qualitätsmodell wird hier frühzeitig zu wertvollen, allgemein wiederverwendbaren Ergebnissen führen. Dies umfasst insbesondere eine deutlich gewachsene Transparenz bezüglich des ansonsten häufig unscharfen Konzeptes der inneren Qualität. Die zu erstellenden Werkzeugprototypen versprechen hier wertvolle Einblicke, die den Projekterfolg sichern helfen. Die Einrichtung des Wirtschaftlichkeitsnachweises verspricht interessante Erkenntnisse über den Einfluss der inneren Qualität. Insbesondere werden hiervon Daten erwartet, die helfen, den Transfer der Ergebnisse von QBench in die Praxis zu vereinfachen, da hier als Einstiegshürde meistens rein wirtschaftliche Gründe hervorgebracht werden.

Mit der genannten größeren Transparenz wird es auch möglich sein, Problembhebungen (semi-)automatisch vorzuschlagen. Erste Ergebnisse zur Verbindung von Problemerkennung und -behebung erwarten wir bis zum Jahresende.