Ein iterativer, eigenmotivierter Regelkreis zur Einführung von Code-Quality-Management innerhalb der Raiffeisen Bausparkasse GmbH Wien

Dr. Frank Simon*, Dr. Daniel Simon**, Andreas Madjari***

(*SQS Research & Innovation| **SQS Competence Center Application Intelligence),
SQS AG Köln

***Raiffeisen Bausparkasse Gesellschaft m.b.H., Wiedner Hauptstrasse 94
A-1050 Wien
(Frank.Simon|Daniel.Simon)@sqs.de
Andreas.Madjari@raibau.at

Abstract: Die Einführung von Code-Quality-Management (CQM), d.h. der systematischen Transparenzschaffung und Optimierung technischer Aspekte großer IT-Systeme, wird aus wirtschaftlichen und risikobezogenen Gründen zunehmend relevant. Der Weg zu einem solchen erfolgreichen CQM ist allerdings in jedem neuen Projekt schwierig und aufwändig: Ein wesentliches Risiko besteht dabei in der fehlenden konsensorientierten Abstimmung der verwendeten Kriterien, entlang derer CQM vorangebracht wird, sowie einem vorzeitigen Involvieren des Managements bzgl. erreichter bzw. erhoffter Ergebnisse. In diesem Papier wird der gesteuerte und erfolgreiche Einsatz von CQM innerhalb der Raiffeisen Bausparkasse GmbH Wien beschrieben. Kern ist hierbei ein iterativer und eigenmotivierter Regelkreis, der die Zustimmung des Teams systematisch erarbeitet und damit gleichzeitig eine höhere Management-Beteiligung ermöglicht. Dieser ermöglicht heute ein effizientes CQM innerhalb der Raiffeisen Bausparkasse GmbH Wien. In diesem Papier können einige erreichten Ergebnisse sogar quantifiziert werden, wobei auch wertvolle, über das reine CQM hinausgehende Mehrwerte festgestellt werden konnten.

1 Einleitung

Der Kostentreiber Nummer Eins für Software-Systeme ist der Wartungsaufwand, der für jedes erfolgreiche Software-System sehr bald den Aufwand für die Erstentwicklung bei weitem übersteigt [Ko04]. Die bei der Software-Wartung entstehenden Kosten hängen dabei direkt von der inneren, technischen Qualität der Software ab [MB97], also von denjenigen Aspekten eines Systems, die weder für den Anwender noch für das Management direkt sichtbar sind. Die inneren Aspekte wie Prüfbarkeit, Wartbarkeit, Verstehbarkeit und ähnliche sind jedoch für die Software-Entwickler von entscheidender Bedeutung, da sie unmittelbar den Aufwand mitbestimmen, der bei der Korrektur, Anpassung oder Weiterentwicklung eines existierenden Systems verursacht wird. Im Zuge der Entwicklung eines auch technisch zukunftsfähigen Systems muss eine Qualitätssicherung folglich auch solche Reengineering-Tätigkeiten motivieren, die diese

innere Code-Qualität untersuchen und verbessern. Dieses Ziel wird durch Code Quality Management (CQM) verfolgt, das sich dabei ganz maßgeblich der Quellcode-Metriken bedient ([SSM06]): CQM dient der Kontrolle und der Überwachung laufender Projekte im Hinblick auf technische Qualitätsaspekte und ermöglicht damit eine Transparenz bezüglich technischer Risiken und Optimierungspotentiale.

In vielen Projekten hat sich gezeigt, dass bei kontinuierlicher Anwendung von CQM die Projekte deutlich einfacher und zuverlässiger zu warten sind, was sich u.a. in deutlich reduzierten Aufwänden während der Wartung äußert (zwischen 10% und 20% Einsparung, vgl. z.B. [SR04], [SSR05]).

1.1 Typische Risiken

Auch wenn die Vorteile eines technisch hochwertigen Codes primär auf Seiten der Entwicklung liegen (einfachere Planung, zuverlässigere Umsetzung, etc.), greift CQM in einen für die Entwickler sensiblen Bereich: Der Quellcode als geistige Schöpfung des einzelnen Entwicklers, der bei der Prüfung der externen Qualität von Fremden sonst nicht direkt betrachtet wird, ist plötzlich das Ziel einer Untersuchung, deren Resultate prinzipiell auch höheren Ebenen der Organisation zugänglich sind. Dieses Risiko wird häufig unterschätzt, was wiederum zu deutlichen CQM-Hindernissen führen kann. Typische Fehler bei der Einführung von CQM sind daher:

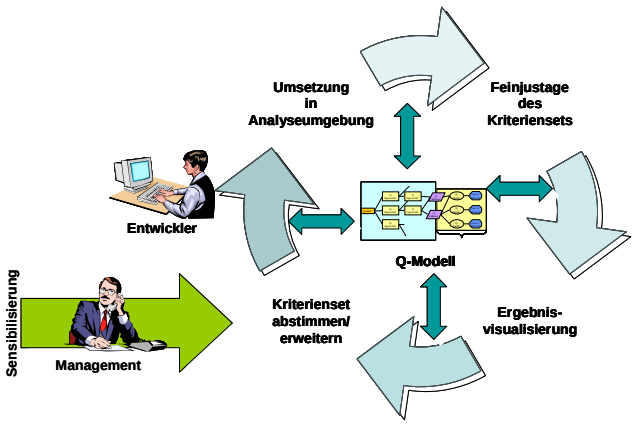
- Fehlende Entwicklereinbeziehung: Da CQM letztlich die von Entwickler erstellten Artefakte optimiert, müssen sie den Zweck von CQM verstanden haben und diesen mittragen. Neben einer sicherlich normativen Kraft, Software ingenieurmäßig erstellen und optimieren zu wollen, müssen dennoch projektspezifische Besonderheiten, historische Parameter und andere Vorbehalte angegangen werden. Werden Entwickler inhaltlich nicht integriert, existiert eine inhaltliche Lücke zwischen der Managementdarstellung auf dieses Thema und den tatsächlichen Aktivitäten dazu.

- Fehlende Managementunterstützung: CQM kann wirksam nur betrieben werden, wenn es ein strategisches Ziel darstellt. Anderenfalls können gerade initiale Aufwände zur Etablierung von CQM nicht entsprechend gestemmt werden. Bei der Umsetzung ist darüber hinaus wichtig, dass der aktuelle Stand von CQM sowie dessen Entwicklung kontinuierlich und häufig auch aggregiert dem Management bereitgestellt werden muss. Eine CQM-Maßnahme von Entwicklern für Entwickler unter Ausschluss des Managements funktioniert nicht.

Da CQM folglich die größten Potentiale nur in einer kooperativen Zusammenarbeit mit den Entwicklern bei gleichzeitiger Integration des Managements hat, wird in diesem Papier ein diese Anforderungen berücksichtigender Einführungsprozess vorgestellt, wie er bei der Raiffeisen Bausparkasse GmbH Wien erfolgreich eingesetzt wurde.

2 Regelkreis

Der in der folgenden Abbildung gezeigte Regelkreis mit einer Initialphase (Schritt 1: Sensibilisierung) und einigen iterativ durchzulaufenden Aktivitäten versucht, die beschriebenen Risiken durch ein gesteuertes, evolutionäres Vorgehen zu kompensieren. Die einzelnen Schritte werden im Folgenden jeweils kurz vorgestellt.



2.1 Schritt 1: Management-Sensibilisierung

In diesem Schritt wird vor dem Management eines Projektes die gesamte CQM-Vision mit dem Ziel vorgestellt, ein Mandat für die Etablierung zu erhalten. Der hier vorgestellte Prozess kann für diese Darstellung bereits effizient verwendet werden. Innerhalb der Raiffeisen Bausparkasse GmbH Wien wurde dieser Schritt in mehreren Workshops zusammen mit dem Management und den Chefarchitekten durchgeführt. Die Workshops bezogen sich hierbei auf die grundsätzlich möglichen Mehrwerte durch CQM, dem Aufzeigen und Demonstrieren von Referenzen aus anderen Projekten sowie das Erläutern der grundlegenden Methodik (vgl. z.B. [SSM06]). Die Automatisierbarkeit war hierbei besonders wichtig. Aufgrund der technischen Parameter der Raiffeisen Bausparkasse GmbH Wien wurde im konkreten Fall die Bauhaus-Suite von Axivion [Ax08] sehr erfolgreich pilotiert und im Folgenden auch weiter eingesetzt.

2.2 Schritt 2: Kriterienset abstimmen / erweitern

In diesem ersten Schritt des iterativen Vorgehens geht es darum, das Kriterienset, anhand dessen die technische Qualität festgestellt und optimiert wird, im Konsens mit der Entwicklung zu erstellen. Neben etablierten Best-Practices werden hier idealerweise „Bauchgefühle“ der Entwickler selbst herangezogen, die Aussagen bzgl. der technischen Qualität ermöglichen: Die Kunst ist es, diese Bauchgefühle auf systematisch erhebbare Indikatoren abzubilden. Existiert z.B. das Bauchgefühl, dass bei durchgeführten Änderungen diese sehr häufig an einer Vielzahl anderer, vorher unbekannter Stellen nachgezogen werden müssen, so kann dies eine guter Indikator dafür sein, nach unterschiedlichen Formen der Code-Duplikation zu suchen.

Bei der Raiffeisen Bausparkasse GmbH Wien wurde das Kriterienset abgestimmt, bevor die eigentliche Werkzeugunterstützung vorhanden war; etwaig mit dem Werkzeug gelieferte Standard-Kennzahlen wurden global ausgeschaltet, wenn sie nicht Teil des abgestimmten Kriteriensets waren. Die Bauhaus-Suite von Axivion unterstützt dieses Vorgehen durch eine maximale Anpassbarkeit aller zu erhebenden Kennzahlen. Die Erfahrung zeigt, dass die Unterstützung durch die Entwickler für den organisatorischen Erfolg der Einführung wesentlich ist und den natürlichen Vorbehalten gegen CQM angemessen entgegen getreten werden muss [SS05].

2.3 Schritt 3: Umsetzung in Analyseumgebung

In diesem Schritt geht es darum, das abgestimmte Kriterienset für eine werkzeuggestützte Analyse bestmöglich innerhalb einer konkreten Analyseumgebung umzusetzen. Für diesen Zweck haben alle großen Analysewerkzeuge eigene Entwicklungsschnittstellen, die die Implementierung spezifischer Metriken und Indikatoren erlauben.

Innerhalb der Raiffeisen Bausparkasse GmbH Wien bestand dieser Schritt im Wesentlichen aus der Implementierung der Kriterien mittels Python, das zur Anbindung an das Bauhaus-API verwendet werden kann. Die Infrastruktur erlaubt einerseits die Gesamtvermessung von Releases und im Nachgang die Historisierung der erhobenen Daten in einer Datenbank. Andererseits lassen sich einzelne Metriken selektiv erheben und ermöglichen so auch die im nachfolgenden Schritt beschriebene Kalibrierung.

2.4 Schritt 4: Feinjustage des Kriteriensets

Dieser Schritt ist ganz wesentlich zur Sicherstellung des Erfolgs von CQM: Auch wenn das Kriterienset zu diesem Zeitpunkt abgestimmt ist, bedarf es nach erster Ergebnissichtung fast immer einer Feinjustage. Typische Parameter der Feinjustage sind:

- Ausblendung von Third-Party-Produkten und Nicht-Produktiv-Code für die Analyse (z.B. Datenzugriffschichten über die RogueWave-Bibliotheken).
- Bewusste Definition von Ausnahmen aufgrund technischer Constraints durch Supression-Listen für bestimmte Quellcodeanteile (erlaubte Verwendung von new/delete-Operatoren in der Speicherverwaltung, während überall sonst eben eine eigene Speicherverwaltungsabstraktion verwendet werden muss).
- Detaillierte Operationalisierung auf die Zieltechnologie (z.B. was ist eine Klasse in C++, wie zählen hier Templateinstanzen, etc.).
- Identifikation von Systemteilen, die nicht im Fokus der Vermessungen liegen (z.B. Teilsysteme, deren Ablösung kurz bevorsteht).

Innerhalb der Raiffeisen Bausparkasse GmbH Wien wurde dieser Schritt durchgeführt, in dem die konkreten Ergebnisse einer Bauhaus-Anwendung der Entwicklermannschaft

vorgestellt wurden. Ein Großteil des dort festgestellten Ergebnisrauschens konnte den oben vorgestellten vier Parametern zugesprochen werden. Die Folge waren entweder detailliertere Kriterien oder Anpassungen innerhalb des verwendeten Werkzeugs, das diese Festlegung von Messparametern unterstützt.

2.5 Schritt 5: Ergebnisvisualisierung und –auswertung

Die konkreten Befunde in der Software (sowohl punktuell als auch über die Zeit), die entlang der im Schritt 2 abgestimmten Kriterien erhoben werden, müssen den unterschiedlichen beteiligten Rollen an zentraler Stelle bereitgestellt werden. Besonders geeignet sind hierbei Darstellungen, die rollenspezifische Aggregationen anbieten. Nicht alle Ergebnisdetails müssen hierbei allen Rollen zur Verfügung stehen, sondern es können geeignete personen- oder gruppenbezogene Sichten gewählt werden.

Das Monitoring über die Zeit ermöglicht innovative Ansätze der Qualitätsverbesserung: Im konkreten Kontext wird ein einmal erreichtes Minimum als künftiges Maximum für Folgevermessungen herangezogen. Diese Qualitätsvorgabe wird häufig als „Bestandsschutz“-Konzept bezeichnet.

Die derzeit umgesetzten Visualisierungen zielen auf die Verwendung durch das Management in tabellarischer Form ab. Aus den erzeugten Tabellen werden hierfür eigene Statistiken, Auswertungen und managementtaugliche Visualisierungen je nach Bedarf generiert. Für die Entwickler selbst werden je Befundkategorie die Fundstellen für die einzelnen Abweichungen derart aufbereitet, dass bis in den Sourcecode des vermessenen Releasestandes navigiert werden kann.

Ab diesem Schritt beginnt der Regelkreis: Die bis hierhin festgestellten Justierungsnotwendigkeiten sowie Visualisierungsergebnisse werden für die nachfolgende Kriterienset-Modifikation bzw. Erweiterung verwendet. Durch das wiederholte und nachhaltige Durchlaufen dieser Iterationen kann – wenn die Schritte nicht zu grobgranular gewählt sind – sichergestellt werden, dass alle Beteiligten frühestmöglich interagieren können, wenn CQM-Parameter nicht optimal eingestellt sind. Innerhalb der Raiffeisen Bausparkasse GmbH Wien war die zeitliche Granularität so gewählt, dass ein derartiger Prozessdurchlauf etwa 4 Wochen benötigt.

3 Ergebnisse

Der Fokus der Primär-Ergebnisse der ersten Prozessdurchläufe lag auf Qualitätseigenschaften, die die Analysierbarkeit und Portierbarkeit der Probanden adressieren. Durch den Einsatz der Bauhaus-Suite wurden die entsprechenden Indikatoren anschließend implementiert; typische Indikatoren hierfür sind:

- Ausschluss der Verwendung von „char *“. Zur Vermeidung der üblichen Schwierigkeiten mit „char *“ sollen alle davon betroffenen Stellen langfristig eliminiert werden, in jedem Fall aber kurzfristig keine neuen hinzukommen.

- Ausschluss der uneingeschränkten Verwendung von new/delete-Operatoren. In der Anwendung der Raiffeisen Bausparkasse GmbH Wien ist ein Smartpointer-Konzept realisiert und die dort angebotenen Möglichkeiten zur Speicherverwaltung sind obligatorisch von den Entwicklern zu nutzen.

- Vermeidung der Verwendung bestimmter Konstrukte der Datenzugriffsschicht-API. Hier werden aus der Erfahrung der Entwickler heraus bekannte performancekritische oder in ihren Auswirkungen problematische Verwendungen in der weiteren Entwicklung untersagt.

- Einhaltung von Namens- und Dateiinhaltskonventionen durch die Entwicklung. Verstöße werden als Metrik geführt und nach Erkennen durch die Entwicklung bereinigt.

- Verfügbarkeit eines globalen Dependency-Modells über das System. Damit kann der Grad der Auswirkung einer Änderung auf Methodenebene im Gesamtsystem leichter bestimmt werden.

Die ersten Ergebnisse der Entwicklung über die Zeit sind sehr positiv und lassen sich punktuell bereits quantifizieren:

- Durch die Anwendung von CQM konnte ca. 10% des Codes identifiziert werden, der statt in allen ca. 230 Executables nur noch in 10 Executables benötigt wird. Die damit eingesparten Compile- und Testzeiten sind offensichtlich. Die Umsetzung dieser Optimierung hätte ohne den implementierten Automatismus ca. 200 Personenstunden gekostet; nun war sie nach 40 Personenstunden realisiert.

- Das Identifizieren von performance-relevanten Stellen, eine immer wiederkehrende Tätigkeit in derart großen Systemen, konnte mit dem gewählten Verfahren auf etwa 70% des ursprünglichen Aufwandes beschleunigt werden.

- Selbst bei akuten Produktionsproblemen kann der gewählte Ansatz unterstützen: So konnte alleine durch die Verfügbarkeit entsprechender Metriken die Dauer eines Produktionsstopps mindestens halbiert werden und der technische Impact fundiert beurteilt werden.

Bei den Sekundär-Ergebnissen der kontinuierlichen Vermessung der Software-Releases ist die innerhalb der Entwicklung gestiegene Sensibilisierung für die technische Qualität der Software erreicht worden. Die durch den Regelkreis angestoßene kontinuierliche Abstimmung und Diskussion im Team führt letztlich zu einer Loslösung von Kopfmonopolen hin zur Entwicklung von Quellcode, der mehr als nur einem Individuum verständlich ist. Damit hat sich das grundsätzliche Qualitätsverständnis der Entwicklung verbessert und ein wesentlicher Schritt hin zu einem wirklich ingenieurmäßigen Vorgehen der Softwareentwicklung ist möglich. Dieses hilft u.a. dabei, den Wert einer lokalen Software-Entwicklungsabteilung zu belegen, indem durch CQM die Qualität

ihrer Arbeit systematisch und quantitativ belegt werden kann. Zwar dienen weiterhin die fachlichen Anforderungen an die Software als primärer Treiber für die Weiterentwicklung. Durch den erzielten Konsens innerhalb des Entwicklerteams können nun aber auch technische Reengineering-Tätigkeiten dem Management transparent gemacht und der Erfolg belegt werden.

4 Nächste Schritte und Ausblick

Der eingeschlagene Weg innerhalb der Raiffeisen Bausparkasse GmbH Wien wird weiter beschritten. Hierbei hilft der vorgeschlagene Prozess, den Weg der kleinen Schritte kontinuierlich abgestimmt und gesteuert voranzukommen. Neben den nächsten Schritten, das Kriterienset nach und nach ganzheitlicher zu gestalten, ist ein weiterer wesentlicher Schritt die Ausweitung von CQM innerhalb der Raiffeisen Bausparkasse GmbH Wien. Dies bedeutet die Anwendung des Prozesses für weitere Applikationen sowie die Einbeziehung weiterer Managementebenen. Letzteres kann interessant sein, um CQM z.B. auch für etwaig auswärts erstellte Applikationen anzuwenden, in dem das Kriterienset vertragsrelevant postuliert wird (vgl. z.B. [CS07]).

5 Referenzen

[Ko04] J. Koskinen, „Software Maintenance Costs.“, im Internet verfügbar unter http://www.cs.jyu.fi/~koskinen/smcosts.htm, September 2004

[MB97] G. Marliss, M. Ben-Menachem: „Software Quality: Producing Practical, Consistent Software”, Thomson Publishing, 1997

[SR04] F. Simon, U. Richter: „Mit Code-Metriken Wartungskosten senken: Controlling technischer Qualität“, Metrikon 2004, Berlin

[SSR05] F. Simon, M. Studemund, Ch. Rioth: „ROI-Modell für das Reengineering zur Optimierung technischer SW-Qualität“, Proceedings der WSR2005, Universität Koblenz-Landau, 2005

[SS05] D. Simon, F. Simon, „Das wundersame Verhalten von Entwicklern beim Einsatz von Quellcode-Metriken“, Metrikon 2005, Kaiserslautern, November 2005

[SSM06] F. Simon, O. Seng, Th. Mohaupt: „Code Quality Management”, Dpunkt Verlag; Mai 2006

[CS07] Antonio Conte, Frank Simon: „Partnerschaftliche Lieferantensteuerung“, Proceedings der Software-Quality-Conferences 2007, Düsseldorf, Mai 2007

[Ax08] Axivion: Axivion GmbH, Stuttgart, http://www.axivion.de