Quality Control in Action*

Nils Göde

CQSE GmbH
Lichtenbergstr. 8, 85748 Garching bei München
goede@cqse.eu

Zusammenfassung

Trotz der großen Zahl statischer Analysewerkzeuge, die Qualitätsdefizite im Quelltext aufzeigen, nimmt die Qualität in den meisten Systemen kontinuierlich ab. Der Schritt von der reinen Beobachtung zu einer echten Verbesserung stellt in der Praxis immer noch eine große Hürde dar. Dieser Beitrag beschreibt unsere Erfahrungen bei der schrittweisen Einführung eines Quality-Control-Prozesses für ein Java-System in der Versicherungsbranche. Unsere Ergebnisse zeigen, dass allein durch den Einsatz von statischer Analyse noch keine Qualitätsverbesserung erzielt wird. Durch einen begleitenden Quality-Control-Prozess lässt sich die Qualität allerdings kontinuierlich steigern.

1 Quality Control

Das Ziel von Quality Control ist die Reduktion der Qualitätsdefizite („Findings“) im Quelltext und die damit einhergehende langfristige Sicherung der Wartbarkeit. *Quality Control* fasst dabei alle Maßnahmen zusammen, die für die Verbesserung der Qualität erforderlich sind. Eine ausführlichere Beschreibung des Vorgehens findet sich in [5].

2 Setup

Bei dem beobachteten Java-System handelt es sich um ein betriebliches Informationssystem aus der Versicherungsbranche mit einer Historie von mehreren Jahren. Der Quelltext wird kontinuierlich durch das Analysewerkzeug *Teamscale*[3, 4] auf Qualitätsdefizite untersucht. Die Ergebnisse stehen den Entwicklern durch ein Web-Interface sowie direkt in der IDE zur Verfügung.

Zusätzlich finden monatliche Retrospektiven statt in denen wir als externe Experten der CQSE die Entwicklung der Qualität mit den Entwicklern diskutieren. Auf dieser Basis erstellen wir für ausgewählte Findings konkrete Arbeitsaufträge („Tasks“), die im Issue-Tracker eingestellt und bei der Sprintplanung berücksichtigt werden.

*Das diesem Artikel zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen **Q-Effekt, 01IS15003A** gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

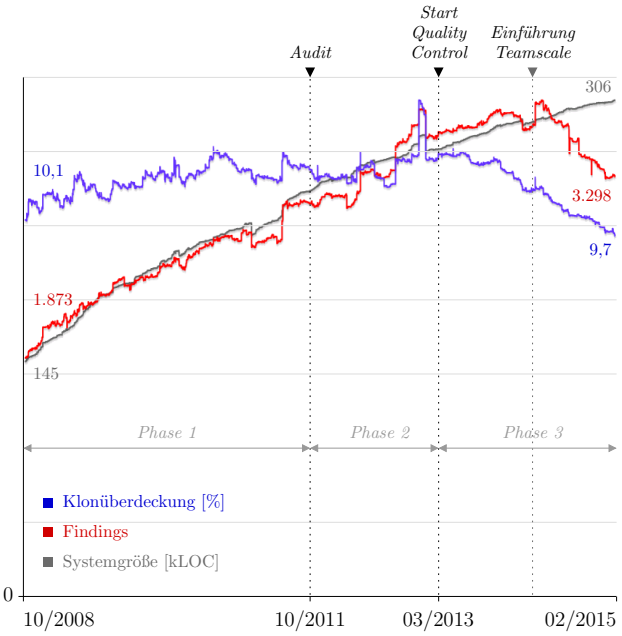

Abbildung 1: Trend zentraler Metiken

3 Evolution

Abbildung 1 zeigt die Entwicklung der zentralen Metriken vom Beginn der Versionsverwaltung bis heute. Diese beinhalten die Größe des Systems, die Klonüberdeckung und die Zahl der Findings. Die Abbildung zeigt, dass sich die Systemgröße im beobachteten Zeitraum etwa verdoppelt hat. Die Zahl der Findings ist auch deutlich gestiegen, während die Klonüberdeckung leicht gesunken ist.

Die Evolution lässt sich grob in drei Phasen unterteilen. Der Übergang von der ersten zur zweiten Phase besteht aus einem umfassenden Audit des Quelltextes durch die CQSE und der Einführung von *ConQAT*[1] zur kontinuierlichen Analyse des Quelltextes. Der Übergang zur dritten Phase ist charakterisiert durch den Beginn des Quality-Control-Prozesses. Im Laufe der dritten Phase wurde *ConQAT* zudem durch das Werkzeug *Teamscale* abgelöst.

Während die Systemgröße in jeder Phase etwa gleichbleibend zunimmt, nimmt die Zahl der Findings und die Klonüberdeckung in den ersten Phase deutlich zu. Auch in der zweiten Phase nimmt die Qua-

lität weiter ab – trotz der Tatsache, dass der Audit den Entwicklern einen umfassenden Überblick über die Qualitätsdefizite gegeben hat und der Quelltext kontinuierlich durch *ConQAT* analysiert wurde. Erst in der dritten Phase, nach der Einführung des Quality-Control-Prozesses, ist eine Trendwende erkennbar, die durch die Einführung von *Teamscale* noch einmal deutlich verstärkt wird.

4 Beobachtungen

Dieser Abschnitt fasst unsere im Laufe des Projektes gesammelten Beobachtungen zusammen, die gleichzeitig die Ausgangsbasis für zukünftige Verbesserungen des Quality-Control-Prozesses bilden.

4.1 Externer Quality Engineer (QE)

Es hat sich bewährt die Rolle des QE außerhalb des Entwicklungsteams zu besetzen. Anderenfalls besteht die große Gefahr, dass das Thema Qualität permanent zugunsten des Tagesgeschäftes vernachlässigt wird. Durch die externe Besetzung ist sichergestellt, dass das Team regelmäßig mit dem Thema Qualität konfrontiert wird.

4.2 Interner Ansprechpartner

Zusätzlich zum QE hat es sich bewährt, einen Ansprechpartner aus dem Team zu ernennen, der unter anderem die Kommunikation mit dem QE führt. Die Hauptaufgabe dieser Person ist dafür zu sorgen, dass die vom QE erstellten Tasks und Themen Einzug in die Team-interne Sprintplanung erhalten.

4.3 Tasks in Sprintplanung

Um die Bearbeitung der vom QE erstellten Tasks zu gewährleisten, müssen diese gleichberechtigt mit anderen Änderungen in der Entwicklungsplanung berücksichtigt werden. Hierfür sollte der interne Ansprechpartner aus dem Team verantwortlich sein. Bevor dies praktiziert wurde, sind Qualitätsverbesserungen häufiger im Zuge vieler anderer Änderungen „vergessen“ worden.

4.4 Pfadfinderregel

Das Befolgen der Pfadfinderregel – den Code qualitativ besser oder zumindest gleichwertig zu hinterlassen wie man ihn vorgefunden hat – wird in der Praxis leider noch nicht konsequent befolgt. Das Problem an dieser Stelle ist zum einen die Frequenz der Analyse (erst nach dem Commit stehen die neuen Findings zur Verfügung). Zum anderen stellt sich die Behebung mancher Findings als sehr aufwändig heraus, so dass das Verhältnis des Aufwands zwischen der eigentlichen Änderung und der Qualitätsverbesserung nicht gerechtfertigt scheint.

4.5 IDE-Integration

Ohne die Integration der Findings in die IDE ist eine Qualitätsverbesserung kaum zu erzielen. Der Kontextwechsel von der IDE in z.B. eine Web-basierte Ansicht ist nicht praktikabel. Erst mit der Einführung des *Teamscale*-Plugins ist die Zahl der Findings, die im Rahmen anderer Änderungen entfernt wurden spürbar gestiegen.

4.6 Web-Dashboard

Auch wenn die Web-basierte Ansicht im Vergleich zur einfachen Ansicht der Findings in der IDE eine Fülle weiterer Informationen bietet, so wird diese von Entwicklern fast nicht berücksichtigt. Andererseits ist die Web-basierte Ansicht für den QE zwingend erforderlich, da nur hier die notwendigen Delta-Informationen [2] und die Evolution der Findings sichtbar ist.

4.7 Know-How Transfer

Durch die monatlichen Retrospektiven, bei denen ausgewählte Qualitätsdefizite im ganzen Team diskutiert werden, entsteht ein nicht zu unterschätzender Know-How-Tansfer. Dies betrifft sowohl Wissen über die Java-Programmierung im Allgemeinen als auch Wissen über das System im Speziellen. Zudem werden unter Umständen übergreifende Probleme diskutiert, die sich aus der lokalen Sicht einzelner Entwickler nicht lösen lassen.

4.8 Ressourcenbedarf

Um Qualitätsverbesserung in der Praxis zu erreichen, müssen dem Thema entsprechende Ressourcen zur Verfügung gestellt werden. In diesem konkreten Projekt wird in jedem Sprint ein festgelegter Prozentsatz des Entwicklungsbudgets für nicht-fachliche technische Qualitätsverbesserungen reserviert. Dadurch wird vermieden, dass Qualitätsverbesserungen dem Tagesgeschäft zum Opfer fallen.

5 Zusammenfassung

Der Trend in Abbildung 1 zeigt, dass durch Quality Control eine kontinuierliche Qualitätsverbesserung erzielt wird. Aufgrund der überwiegend positiven Erfahrungen und Ergebnisse wird das Verfahren zukünftig auf weitere Projekte ausgeweitet.

Literatur

[1] CQSE GmbH. ConQAT. `www.conqat.org`.

[2] N. Göde and F. Deissenboeck. Delta analysis. *Softwaretechnik-Trends*, 32(2), 2012.

[3] N. Göde, L. Heinemann, B. Hummel, and D. Steidl. Qualität in Echtzeit mit Teamscale. *Softwaretechnik-Trends*, 34(2), 2014.

[4] L. Heinemann, B. Hummel, and D. Steidl. Teamscale: Software quality control in real-time. In *Proceedings of the 36th International Conference on Software Engineering*, 2014.

[5] D. Steidl, F. Deissenboeck, M. Poehlmann, R. Heinke, and B. Uhink-Mergenthaler. Continuous software quality control in practice. In *Proceedings of the 30th International Conference on Software Maintenance and Evolution*, 2014.