# Statische Code-Analyse als effiziente Qualitätssicherungsmaßnahme im Umfeld eingebetteter Systeme

Dr. Daniel Simon (SQS AG), daniel.simon@sqs.de, Dr. Frank Simon (SQS AG), frank.simon@sqs.de

## 1 Einleitung

In diesem Papier werden Erfahrungen aus gemeinsamen Projekten des Kunden und der SQS bei der Durchführung proaktiver Qualitätssicherungsmaßnahmen im Umfeld von eingebetteten Telekommunikationssystemen erläutert. Dabei werden insbesondere die Vorteile von frühzeitig ergreifbaren Qualitätssicherungsmaßnahmen mit Hilfe statischer Code-Analysewerkzeuge sowie deren ungefähre Quantifizierung erkennbar.

## 2 Aufgabenstellung

An die Kommunikations- und Informationssysteme, die der Kunde entwickelt, werden besondere Ansprüche in Hinsicht auf Zuverlässigkeit und Verfügbarkeit bei zugleich limitierten Hardware-Ressourcen gestellt. Dabei reicht die Palette der Softwarearten von einfachen Hardwaretreibern für Standardbetriebssysteme bis hin zu komplexen Benutzerschnittstellen für Endanwender.

Durch den Einsatz von statischen Analysetechniken soll bereits im Vorfeld der durchzuführenden dynamischen Tests die Fehlerrate verringert werden, um damit den Ablauf der Tests durch geringere Fehlerraten zu beschleunigen. Die vor der Durchführung der Qualitätssicherungsmaßnahmen aufgestellte Hypothese lautet, dass auch in der Praxis durch Qualitätsaudits mittels statischer Code-Analysen Laufzeitfehler vorab identifiziert und beseitigt werden können, die andernfalls nur durch Testen gefunden werden. Damit sollten die Projekte insgesamt von einem reduzierten Aufwand bei der Testdurchführung profitieren und eine höhere Qualität der Gesamtsysteme für den Kunden erzielen.

## 3 Statische Analysen im Vergleich zu dynamischen Tests

Dynamisches Testen im Umfeld des Kunden ist in frühen Phasen der Software-Entwicklung sehr aufwendig, da beispielsweise Software von Kommunikationssystemen im Verbund mit der noch in der Entwicklung befindlichen Hardware arbeiten muss. In späteren Phasen muss die Software im Zusammenspiel mit vielen Umsystemen getestet werden, die am Standort der Entwicklung nicht oder noch nicht zur Verfügung stehen.

Die statischen Code-Analysen bieten im Vergleich zu dynamischen Tests u.a. die folgenden Vorteile:

- Abdeckung von zusätzlichen Qualitätsaspekten wie Wartbarkeit und Analysierbarkeit.
- Verwendbarkeit von technisch abhängigen und fachlich unabhängigen Prüfregeln zur Kommunikation und Homogenisierung in Entwicklergruppen (z.B. gewünschte Architekturen oder Designs).
- Weitgehende Automatisierbarkeit der Prüfungen, ohne signifikante Initialaufwände zu benötigen.
- Durchführbarkeit für einzelne Software-Fragmente lange vor der Integration zu einem lauffähigen Gesamtsystem.
- Durchführungszeitpunkt unabhängig vom Stand des Projekts (für die Prüfung von Regeln idealerweise schon zu Beginn, aus Werkzeugsicht später aber genauso möglich).
- Durchführung der statischen Analyse mit geringen Kenntnissen der Fachlichkeit möglich und daher minimal-invasiv für den Projektverlauf.
- Vollständigkeit der Analyse aufgrund der statischen Gesamtbetrachtung des Systems (der Abdeckungsgrad dynamischen Tests in aller Regel <80%).
- Einfache Vergleichbarkeit der Messergebnisse mit anderen Systemen/ Versionen durch Benchmarking.

Als Verfeinerung der allgemeinen Zielsetzung der Qualitätssicherung erfolgte beim Kunden eine Fokussierung auf Indikatoren mit großem Einfluss auf die ISO9126-Qualitätseigenschaften [1] Effizienz, Stabilität und Zuverlässigkeit. Auf den Aspekt der Wartbarkeit der Software wird darüber hinaus nur sekundär Wert gelegt, da der Kunde zum Teil erhebliche Wartungs- und Supportanforderungen seiner Kunden erfüllen muss.

Die aufgeführten Vorteile der statischen Analyse motivieren deren zusätzlichen Einsatz, können die dynamischen Tests aber keinesfalls vollständig ersetzen. Der erwartete wirtschaftliche Vorteil der zusätzlichen statischen Analyse liegt neben der erreichbaren Ganzheitlichkeit (z.B. Wartbarkeit) darin, dass die Probanden der dynamischen Analyse eine deutlich bessere Eingangsqualität haben und die dynamischen Tests nur noch solche Abweichungen finden, für deren Identifikation dynamische Tests tatsächlich notwendig sind. Letztlich dient die statische Code-Analyse damit der Effizienzsteigerung dynamischer Tests.

## 4 Projektablauf

Initiiert wurde die Qualitätsoffensive seitens der Projektleitung, die als gesamtverantwortlich für die jeweils untersuchten Systeme zeichnet. Insgesamt wurde die Software dreier verschiedener Anwendungen in unterschiedlichen Technologien mit C-, C++- und C#-Anteilen untersucht. Das erklärte Ziel der Assessments ist die Identifikation von Risiken bzgl. der ISO9126 (vgl. oben). Da für die einzelnen Indikatoren eine Null-Fehlergrenze in der Praxis als nicht sinnvoll eingeschätzt wird, wird zur Bewertung der Indikatoren das allgemeine Risiko mit Hilfe von Benchmarking gegen das SQS-Benchmark-Repository ermittelt, das initiiert durch das Forschungsprojekt QBench mittlerweile Risikokennzahlen von knapp 200 Großprojekten aus der Industrie besitzt [6].

### 4.1 Quellcode-Übernahme

Vor Ort wurde die Codeübernahme und Erfassung der allgemeinen Projektparameter durchgeführt. Dazu gehören unter anderem die verwendeten Programmiersprachen, die eingesetzten Frameworks und ganz wesentlich die Inventarisierung der Fremdanteile. Zum Zeitpunkt des Kickoffs ist die Identifikation von generiertem Code von besonderer Bedeutung, da zumindest einige der untersuchten Indikatoren für generierten Code anders gewichtet werden. Die Beteiligung der Kunden-Entwickler ist für diesen Schritt unbedingt erforderlich.

### 4.2 Vermessung

Die Code-Analyse und die vorläufige Risikoidentifikation mit Hilfe des SQS-Benchmarking-Repository wurde im Anschluss ohne Mitwirkung des Kunden durch die SQS durchgeführt. Aus zwei wesentlichen Gründen kommt bei der Analyse ein ganzer Werkzeug-Zoo zum Einsatz, dessen Resultate im Anschluss dem Kunde in kondensierter und konsolidierter Form übergeben werden. Im konkreten Fall sind dies CAST [2], Bauhaus [3],

Software Tomograph [4], Splint [5] sowie eine Reihe von Perl-, Shell-, und sonstigen Skripten.

- Einerseits sind die Analysatoren primär technologiegetrieben und bieten etwa für bestimmte Sprachen besonders gute, für andere dagegen keine Unterstützung. Darüber hinaus besitzen die Werkzeuge für unterschiedliche Techniken unterschiedliche Detailtreue und Abstraktionsfähigkeit: Während manche Werkzeuge in einer Technik den vollständigen Kontrollfluss analysieren, abstrahieren andere davon und stellen lediglich gröbere, auf Modulebene verfügbare Informationen dar.
- Zweitens sind die Ergebnisse solcher Werkzeuge immer noch nur durch Experten effektiv und effizient anwendbar und zwischen den Werkzeugen untereinander vollkommen inkompatibel. Bereits an recht einfach anmutenden Metriken wie „Lines of Code“, die in allen Werkzeugen auch als solche bezeichnet werden, scheitert ohne entsprechende Kenntnisse die Vision, konsistente und objektive Kennzahlen zu erheben.

### 4.3 Ergebnisanpassung/-anreicherung

Die Ergebnispräsentation vor Entwicklern und die Anreicherung in Form einer Risikobewertung der vermessenen Indikatoren erfolgen nach der Konsolidierung der Analyseergebnisse. Um zu einer Gesamtbewertung der Projekte zu gelangen, wird auf der Grundlage bidirektionaler Qualitätsmodelle [6] eine Aggregation der Bewertung der Einzelindikatoren durchgeführt. Dabei kommt eine durch SQS vorgegebene Standardgewichtung der Indikatoren in Bezug auf die ISO-Eigenschaften zum Einsatz, die gegebenenfalls projektspezifisch angepasst werden kann. Damit sind quantitative Aussagen bzgl. der Ausprägung eines Systems entlang der ISO9126 möglich.

Die Ableitung von konkreten Maßnahmen in Bezug auf die konkreten Befunde der Software bedarf darüber hinaus kundenspezifischer Diskussion: So kann eine Liste hoch-priorisierter Risiken z.B. dadurch relativiert werden, dass ein Großteil der Symptome in Systemteilen liegen, die kurzfristig durch andere Module ersetzt werden. Eine weitere Dimension betrifft den Aufwand, der mit der Behebung gefundener Risiken verbunden ist. Hier existieren zwar Abschätzungen auf Basis langjähriger Erfahrungen; diese hängen aber in jedem Fall vom konkreten Kontext ab. Werden diese drei Dimensionen Benchmark-Risiko, Kunden-Risiko und Aufwand gleichzeitig betrachtet ergibt sich in Folge ein Entscheidungsraum, der Grundlage einer abgestimmten Priorisierung ist. Ziel ist die Identifikation der risikobehaftetsten und aufwandsminimalsten Maßnahmen.

## 5 Erfahrungen und Ergebnisse

Die Methoden und Konzepte der statischen Codeanalyse lassen sich in der Praxis mit Aufwänden zu einem beliebigen Zeitpunkt in die Projekte einbringen, die im Vergleich zu den Aufwänden des Blackboxtestens sehr gering sind (im vorliegenden Projekt knapp 10%). Allerdings gilt auch hier: je früher desto besser, da bereits durch die Einführung derartiger Qualitätssicherungsmaßnahmen das Bewusstsein der einzelnen Entwickler geschärft wird und die Entstehung von problematischem Code von vorneherein unterbleibt.

Die statische Codeanalyse steht frühzeitig bereit, nach Schwächen und Risiken im Quellcode der Applikationen zu suchen - lange bevor der erste Blackboxtest durchgeführt werden kann. Da die Fehler sehr techniknah identifiziert werden, ist das Lösungsvorgehen meist direkt ableitbar und zügig umsetzbar. Im konkreten Fall konnten durch die statische Analyse der Projekte aus dem Hause des Kunden Codestellen identifiziert werden, die im Falle einer Ausführung des Programms mit Sicherheit zu Fehlverhalten des System und schließlich zum Absturz der Software geführt hätten. Da diese vor aufwendigen dynamischen Tests identifiziert und behoben werden konnten, waren die dynamischen Tests insgesamt weniger aufwendig.

Im vorliegenden Projekt dominierten von den Vorteilen der statischen Analyse (vgl. oben) vor allen Dingen folgende Punkte:

- Vollständigkeit: Etwa 20% der identifizierten Fehler wären vermutlich kaum systematisch durch dynamische Tests identifiziert worden. Hierzu zählen insbesondere die Bereiche Speicherverwaltung und Initialisierungsanomalien.
- Erhöhung der Eingangsqualität des dynamischen Tests für deren Aufwandsreduktion: Auch wenn in der Praxis kein sauberer empirisch tauglicher Versuchsaufbau existierte, so konnte im vorliegenden Projekt zumindest im Vergleich zu ähnlichen Systemen des gleichen Unternehmens eine Aufwandsreduktion des dynamischen Tests bis zum Erreichen von Test-Ende-Kriterien um ca. 10-15% erreicht werden.
- Ganzheitlichkeit: Auch wenn die Eigenschaft Wartbarkeit nicht im direkten Fokus der Analyse stand (vgl. oben) so wurde sie dennoch optimiert, da auch diese Abweichungen Teil des Maßnahmenpaketes waren. Quantifizierungen dieser Wartbarkeitseinsparung aus anderen Projekten belegen hier Werte zwischen 10% und 30% (vgl. z.B. [7], [8], [9]).

Der frühe Zeitpunkt einer statischen Analyse hat den Vorteil, das Thema Qualität frühzeitig bis ins Management hinein zu kommunizieren. Damit schafft sie durchgehend und kontinuierlich (Automatisierbarkeit!) Transparenz im Hinblick auf die Qualität der Software. Auch die Möglichkeit, damit Lieferantensteuerung zu einer aktiven und aufgrund der Objektivität partnerschaftlichen Tätigkeit werden zu lassen, wird beim Kunden als sehr positiv angesehen. Somit wird ein nachhaltiger Beitrag zur Verbesserung der Produkte sowohl entwicklungsintern als auch extern für Kunden geleistet.

## 6 Referenzen


[1] ISO/IEC 9126-1:2001

[2] CAST APM, http://www.castsoftware.com/

[3] Bauhaus Suite, Axivion GmbH, http://www.axivion.de/

[4] Software Tomograph, Software Tomographie GmbH, http://www.software-tomography.de/

[5] Splint, http://www.splint.org/

[6] Simon, F., Seng, O., Mohaupt, Th., *Code Quality Management* dpunkt-Verlag, Mai 2006

[7] Studemund, M., Rioth, C., Simon, F. „ROI-Modell für das Reengineering zur Optimierung technischer SW-Qualität“, Proceedings der WSR2005

[8] Richter, U., Simon, F.: „Mit Code-Metriken Wartungskosten senken: Controlling technischer Qualität“, Proceedings der Metrikon 2005

[9] Conte, A., Simon, F.: „Partnerschaftliche Lieferantensteuerung mittels technischer Anforderungen“, Proceedings der SQM2007