

TestBalance: Methoden und Werkzeuge zur Optimierung des Erfolgs von QS-Maßnahmen

Heinrich Schettler
imbus AG
Kleinseebacher Straße 9
D-91096 Möhrendorf

Kurzfassung

Softwareproduzierende Unternehmen benötigen eine Antwort auf die Frage:

Wie muss die Qualitätssicherung einer zu erstellenden Software geplant und durchgeführt werden, um ein betriebswirtschaftlich optimales Produkt zu liefern?

Betriebswirtschaftlich optimal bedeutet dabei, dass der Nutzen der Qualitätssicherung (QS) die dafür aufzuwendenden Kosten möglichst weit übersteigt. Kann dieses Optimum schon bei der Planung von QS-Maßnahmen gezielt angesteuert werden, wird vermieden, dass zu viel Qualitätssicherung mehr Kosten verursacht als die Folgen der in der Software verbleibenden Fehler. Zugleich werden zu hohe Qualitätsrisiken infolge eines zu niedrigen QS-Umfangs vermieden. Insgesamt wird so die optimale Balance zwischen Nutzen und Kosten hergestellt.

Zwar gibt es schon einzelne Modelle zur Prognose von Einflussgrößen auf die Wirtschaftlichkeit von QS-Maßnahmen, es fehlen jedoch der Ausbau und die Integration zu einer lückenlosen und einfach anwendbaren Methode.

Ziel von TestBalance ist, ein einfach anwendbares Gesamtverfahren zu entwickeln, das es dem Projekt- oder QS-Manager in einem SW-Entwicklungsprojekt ermöglicht, das wirtschaftliche Optimum bei der QS-Planung gezielt anzusteuern und auch alternative, nicht optimale Planungen quantitativ zu beurteilen. Überdies soll TestBalance als Kombination von Tools, Methoden und zugrundeliegenden Modellen möglichst universell in Entwicklungsprojekten sowohl kleiner als auch großer softwareproduzierender Unternehmen eingesetzt werden können.

Aufgrund des operativen Projektstarts zum 1.6.2006 konnten zum Zeitpunkt der BMBF-Statuskonferenz vom 26.-28.6.2006 noch keine Projektergebnisse vorliegen. Dieser Artikel faßt daher die Vorhabensbeschreibung des Projekts „TestBalance“ zusammen [TB2006].

Einleitung

Die Anforderungen an die Software-Qualitätssicherung sind wesentlich durch folgende allgemeine Entwicklungen beeinflusst:

- Umfang, Komplexität, Kopplung und Vielfalt von IT-Systemen und Systementwicklungen nehmen zu.
- Die Abhängigkeit des wirtschaftlichen, gesellschaftlichen und persönlichen Lebens von IT-Systemen wächst.

Beide Entwicklungen führen zu einer Zunahme der Risiken aus Software-Fehlern und stellen im Zusammenwirken mit einer gleichzeitigen absoluten oder relativen Verknappung der Mittel für Qualitätssicherung eine große Herausforderung dar.

So wurden beispielsweise Fälle beobachtet, in denen QS-Anforderungen um jährlich 20% steigen, ohne daß eine entsprechende Zunahme der QS-Budgets erfolgt, oder in denen Testbudgets um bis zu 50% reduziert wurden.

Die erforderliche Effizienzsteigerung der QS kann nur teilweise durch verbesserte Prüfverfahren und -techniken (z.B. Automatisierung) erreicht werden. Eine wachsende Bedeutung für Wirtschaftlichkeit, Effektivität und Effizienz der Qualitätssicherung erhält daher der gezieltere Einsatz von verfügbaren Mitteln und Zeit mittels besserer planerischer Methoden.

Das dadurch zugängliche Potential soll mit folgendem Beispiel illustriert werden: Weit verbreitet sind Funktionstests, mit denen die einzelnen Funktionalitäten eines Anwendungssystems (z.B. Masken, Fenster) auf Korrektheit überprüft werden. Je nach Größe des Fehlerrisikos kann dabei eine Funktion mit einem von beispielsweise drei, verschieden intensiven Testverfahren bzw. gar nicht getestet werden. Für ein System aus 100 Anwendungsfunktionen gibt es dann schon etwa 175.000 verschiedene mögliche priorisierende Testpläne. Dieses Testplanspektrum reicht vom Maximaltest, der alle Funktionen mit dem Testverfahren größter Intensität prüft, bis hin zum vollständigen Verzicht auf Test. D.h.: Die möglichen Testpläne unterscheiden sich substantiell in Nutzen und Kosten. Ohne geeignete Planungsinstrumente bleibt dem QS-Planer jedoch nichts übrig, als sich durch Intuition oder Tradition leiten zu lassen und einen dieser Testpläne quasi zufällig herauszugreifen. Die Chance dabei in die Nähe des optimalen Testplans zu kommen, ist jedoch recht gering.

Vor diesem Hintergrund konzentriert sich TestBalance auf anwendungsnahe Forschung und Lösungsentwicklung zur Verbesserung der QS-Planungsprozesse. In Abgrenzung davon ist es nicht Aufgabe von TestBalance, die operativen QS-Prozesse selbst oder spezielle Prüfmethoden zu entwickeln oder zu verbessern. Im Interesse einer praxistauglichen Lösung ist es hingegen erforderlich, daß sich die zu entwickelnden TestBalance-Lösungen möglichst reibungsfrei in die bei Anwendern vorhandenen Prozesse einfügen lassen. Hierbei sollen auch die Bedürfnisse kleinerer und mittlerer Unternehmen berücksichtigt werden.

Ziel

Ziel von TestBalance ist, Lösungen für QS-Projekte zu entwickeln, die sie in die Lage versetzen, ihre Qualitätssicherung betriebswirtschaftlich optimal zu planen. Im Wirtschaftsleben ist generell üblich, Maßnahmen so zu planen, daß die Differenz zwischen dem erwarteten Nutzen der Maßnahmen und den dafür aufzuwendenden Kosten maximal wird.

Im Fall von QS-Maßnahmen liegt der Nutzen (QS-Nutzen) darin, dass das Risiko aus Fehlern eines SW-Systems nach Ausführung der QS-Maßnahme geringer ist als das Risiko ohne ihre Ausführung. Der QS-Nutzen ist daher eine Risikosenkung mit dem Ergebnis eines geringeren Restrisikos, also einer besseren Produktqualität. Dem Nutzen gegenüber stehen die Kosten zur Durchführung der QS-Maßnahme (QS-Kosten).¹

TestBalance quantifiziert Risiken und Restrisiken durch die Fehlerfolgekosten. Fehlerfolgekosten sind die Kosten aus den möglichen Schadensfolgen der im System zu erwartenden,

¹ Vereinfachend und für viele praktische Zwecke ausreichend wird dabei meist davon ausgegangen, dass die durch die QS-Maßnahme entdeckten Fehler auch erfolgreich korrigiert werden. Die Kosten der Korrekturen werden dann bei Bedarf mit den QS-Kosten zu Kosten der Risikosenkung zusammengefaßt.

nicht entdeckten Fehler. Dabei müssen die Folgen für alle relevanten Stakeholder (Anwender, Hersteller,...) berücksichtigt werden.

Abbildung 1 skizziert den prinzipiellen Zusammenhang zwischen Fehlerfolgekosten, QS-Kosten und deren Summe, abhängig von einer geplanten Prüfintensität. Die Summe aus Fehlerfolgekosten und QS-Kosten (Gesamtkosten) ist genau dann minimal, wenn die Differenz aus QS-Nutzen und QS-Kosten maximal wird. Ein Projekt- oder QS-Manager strebt also ein wirtschaftlich sinnvolles Ziel an, wenn er eine Prüfintensität plant, die dem in Abbildung 1 dargestellten Minimum der Gesamtkosten entspricht.

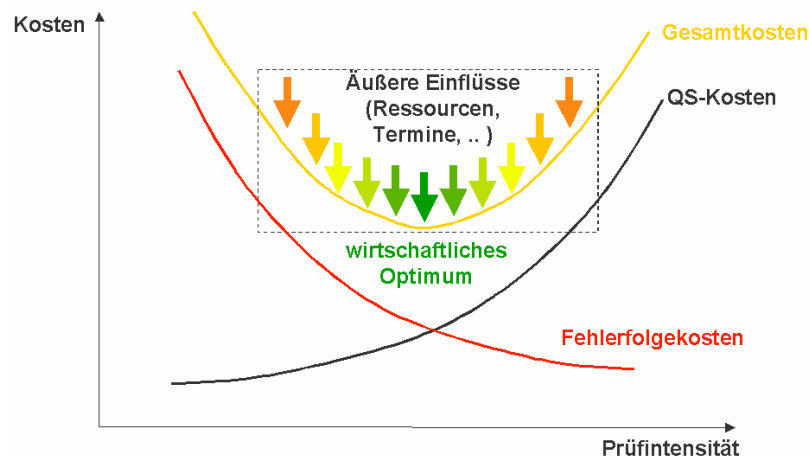


Abbildung 1: QS-Kosten und Fehlerfolgekosten

In praktischen Zusammenhängen wird die Suche nach dem optimalen QS-Plan häufig durch zusätzlich einzuhaltende Vorgaben eingeschränkt.

- Beschränkungen des QS-Budgets, fest definierte Releasetermine oder limitierte QS-Kapazitäten begrenzen die möglichen Prüfintensitäten und können eine höhere Anzahl von Restfehlern und höhere Restrisiken im ausgelieferten System bedingen.
- Qualitätsstandards (z.B. Testabdeckungskriterien, Produktzertifizierungskriterien) erfordern einen Einsatz von QS-Kosten, der den im wirtschaftlichen Optimum übersteigt.

Solche Rahmenvorgaben müssen zusätzlich zum rein betriebswirtschaftlichen Optimierungsziel berücksichtigt werden können. Auch dann muß vorhergesagt werden, welche Auswirkungen Abweichungen vom betriebswirtschaftlichen Optimum auf Nutzen, Kosten und Produktqualität haben.

In realen Projektkontexten können auch andere Optimierungsfragestellungen auftreten. Beispiele dafür sind:

- Die Maximierung des QS-Nutzens bzw. die Minimierung der erwarteten Fehlerfolgekosten mit dem gegebenen Projektbudget und / oder im Rahmen der verfügbaren Zeit. Eine Frage, die sich oft bei der operativen Planung in Testprojekten stellt
- Die Maximierung des Nutzen-Kosten-Verhältnisses von QS-Maßnahmen (entsprechend einer Renditeoptimierung) unter Berücksichtigung von Budget- oder Zeitvorgaben z.B. bei der projektvorbereitenden Planung.

Diese und verwandte Optimierungsaufgaben lassen sich voraussichtlich auf ähnliche Weise wie die Optimierung der Nutzen-Kosten-Differenz bzw. der Summe aus Fehlerfolge- und QS-Kosten lösen.²

Die hier angesprochenen Optimierungsaufgaben können sich bei der Planung einzelner QS-Maßnahmen (z.B. des funktionalen Tests) wie auch bei der abgestimmten Planung aller QS-Maßnahmen eines Projekts (z.B. Reviews, Komponententests, funktionale Tests, Last- und Performance-Tests) stellen. Dabei ist die unterschiedliche Sensitivität verschiedener QS-Maßnahmen auf verschiedenartige Qualitätsrisiken zu berücksichtigen. Nicht zuletzt ist die in der Praxis vorhandene Vielfalt der Ausprägungen von Qualitätssicherung eine weitere Herausforderung.

Lösungskonzept

Die von TestBalance insgesamt angestrebte Lösung besteht aus einer Kombination von Methoden und prototypischen Werkzeugen, die auf quantifizierenden Prognose- und Kalkulationsmodellen für die relevanten Größen beruhen. Ihr Zweck ist, die Frage eines QS-Managers beantworten zu helfen, welche der verfügbaren QS-Maßnahmen in welcher Intensität für die konkrete Projektaufgabe optimal sind.

Teilmodelle

Das erforderliche System aus quantifizierenden Prognose- und Kalkulationsmodellen geht von Informationen und Daten über das zu realisierende System, den Entwicklungsprozeß, Schadensfolgen der zu erwartenden Fehler, die Wirksamkeit von QS-Maßnahmen und die dafür aufzuwendenden Kosten aus.

Bezogen auf die praktisch relevanten QS-Maßnahmen soll das System aus Modellen folgende Teilaspekte abdecken:

- **Initialer Fehlergehalt:** Der vor Durchführung der zu planenden QS-Maßnahmen zu erwartende Fehlergehalt kann durch Größen wie die voraussichtliche Fehleranzahl, die Dichte und Verteilung von Fehlern im Produkt abgeschätzt werden. Dieser Fehlergehalt hängt insgesamt von den Anforderungen an das Produkt, der Qualität vorhandener Produktbestandteile und Eigenschaften des Entwicklungsprozesses, wie z.B. seiner Reife, ab.
- **QS-Effektivität:** Die Wirkung zu planender QS-Maßnahmen beruht auf der Senkung des Fehlergehalts und dem Ausschluß möglicher Fehler und ist u.a. durch initialen Fehlergehalt und die Prüfintensität bestimmt. Zu diesem Teilaspekt gehören auch Modelle zur Prognose des Fehlergehalts im Verlauf von QS-Maßnahmen. Solche Modelle helfen dem QS-Manager ihre Durchführung zu verfolgen und bei Bedarf korrigierend einzugreifen. Nach Abschluß der QS-Maßnahme unterstützen solche Modelle den Nachweis der Produktqualität.
- **QS-Kosten:** Die Kosten für QS-Maßnahmen hängen vom Umfang des zu prüfenden Systems, initialem Fehlergehalt, QS-Effektivität und weiteren Eigenschaften der Prüf- und Entwicklungsprozesse ab.

Modelle für QS-Effektivität und -Kosten bilden gemeinsam die wirtschaftlich relevanten Eigenschaften von QS-Maßnahmen ab und erlauben die Beurteilung ihrer Effizienz.

² Die hier beschriebenen Größen finden sich unter anderer Bezeichnung auch im Rahmen von „Total Cost of Quality“-Modellen wieder. Im „Cost of Quality“-Modell entsprechen die QS-Kosten den „Cost of Quality“ und die Restrisiken bzw. erwarteten Fehlerfolgekosten den erwarteten „Cost of Non Quality“.

- **Fehlerfolgekosten:** Nach Abschluß von QS-Maßnahmen sind Restfehler zu erwarten, deren Umfang, Verteilung und Folgen bereits bei der Planung vorhergesagt werden müssen. Die daraus resultierenden Fehlerfolgekosten hängen u.a. vom nach Ausführung der Prüfung erwarteten Fehlergehalt und den möglichen Schadensfolgen ab. Darunter sind die Schäden für Anwender, weitere Stakeholder und / oder ihre Rückwirkungen auf den Hersteller der Software inkl. der zugehörigen Fehlerkorrekturkosten zu berücksichtigen.
- **Modellintegration:** Die Modellintegration beschreibt das Zusammenspiel der oben skizzierten Teilaspekte und Modelle bei der planerischen Optimierung der Prüfintensität in einem Regelkreis (s. Abbildung 2) und stellt auch die erforderlichen Optimierungsverfahren und –algorithmen bereit. Die Optimierung geht von den für das Projekt relevanten Wirtschaftlichkeitszielen und Vorgaben zu Qualität, Kosten und ggf. Zeit aus.

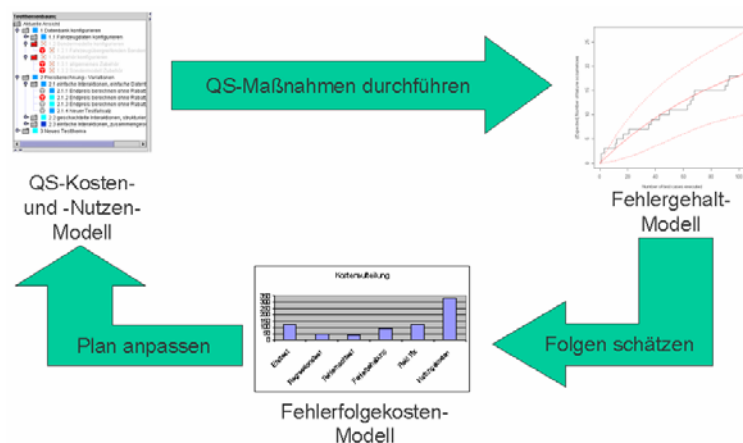


Abbildung 2: TestBalance-Regelkreis

Die skizzierten Prognose- und Kalkulationsmodelle müssen sich auf Basis von in der Praxis verfügbaren oder ermittelbaren Projekt- und Produktdaten bewähren. Speziell für die Anwendung in kleinen und mittleren Unternehmen kann damit gerechnet werden, dass wesentliche Basisdaten nur durch Expertenschätzungen erfaßt werden können.

TestBalance-Gesamtverfahren

Das TestBalance-Gesamtverfahren setzt auf den im vorangegangenen Kapitel skizzierten Modellen und ihrer Integration auf. Es beinhaltet Vorgehensweisen für die Planung und das Management von Prüfprozessen und eine für die Zwecke von TestBalance als Forschungsprojekt adäquate prototypische Tool-Unterstützung, die nach Projektabschluß den Ausgangspunkt für die Entwicklung von Werkzeugen in Produktqualität liefern wird.

Aus der Perspektive der anwendenden QS-Manager und –Planer soll das TestBalance-Gesamtverfahren in einem Projekt hauptsächlich folgende Unterstützung leisten:

- **QS-Strategie:** Zu Projektbeginn wird abhängig von den Qualitätsanforderungen an das System und den abzusehenden Qualitätsrisiken festgelegt, welche Arten von QS-Maßnahmen in welchem Ausmaß durchgeführt werden müssen, um das für das Projekt angemessene wirtschaftliche Optimum anzusteuern.
- **QS-Planung:** Bei der konkreten Planung jeder einzelnen QS-Maßnahme wird im Rahmen der vorgegebenen QS-Strategie im Detail festgelegt, welche Teile bzw. Teilaspekte des zu

prüfenden Systems wie intensiv geprüft werden müssen, um das wirtschaftliche Optimum bezogen auf die QS-Maßnahme und im Zusammenspiel zu erreichen.

- **QS-Monitoring:** Während der Durchführung der QS-Maßnahmen werden abhängig vom Verlauf von Fehlerentdeckung und –ausschluß die Qualitäts- und Aufwandsprognosen präzisiert, als Statusinformation aber auch um frühzeitig Hinweise auf die Notwendigkeit steuernder Eingriffe zu erhalten.
- **Quality-Reporting:** Nach dem geplanten oder vorzeitigen Abschluß der einzelnen oder aller QS-Maßnahmen eines Projekts werden die über Kenngrößen, z.B. zu Restfehlergehalt und Restrisiken erstellten Prognosen zu einer quantifizierten Gesamtaussage über die Produktqualität und die QS-Effizienz konsolidiert. Diese unterstützt die Entscheidung über die Freigabe der Software und hilft den QS-Erfolg zu beurteilen.

Innovativer Gehalt

Die Entwicklung von Modellen und Instrumenten zur Unterstützung von Multi-Attribut-Entscheidungen unter Berücksichtigung ökonomischer Fragen und Faktoren ist eine Schlüsselrichtung der Software-Engineering-Forschung. Insbesondere sind Modelle und Methoden, die einem Unternehmen die optimale Nutzung knapper Ressourcen erlauben, eine zentrale offene Frage [BS2000]. So betont das Capability Maturity Model [CMM1993] die Bedeutung der Fähigkeit, den Software-Entwicklungsprozess quantitativ zu managen, als eine der Key Practice Areas des CMM-Level 4. Der Aspekt der optimalen Verteilung von Prüfleistung auf die unterschiedlichen Prüfprozesse eines Projekts ist ebenfalls noch nicht durchgängig gelöst, obwohl der Zusammenhang, dass je früher ein Fehler entdeckt wird, desto kostengünstiger seine Korrektur ist, seit langem bekannt ist [GT2002].

Zu QS-Kosten existieren zwar verschiedene Schätzmodelle (z.B. "Function Point Analysis" bzw. "Test Point Analysis" [FP2000], Software-Kostenmodell "CoCoMo II" [CO2000]). Diese Modelle machen jedoch nur globale Aussagen über die gesamten QS-Kosten, differenzieren nicht zwischen verschiedenen Prüfprozessen bzw. Qualitätsmerkmalen und machen insbesondere keine Aussagen zum QS-Nutzen bzw. Ergebnis.

Zur Vorhersage des Fehlergehalts existieren zwei wesentliche Ansätze: Vorhersage auf der Basis von Produkt- und Prozessmaßen und Vorhersage auf Basis von Ausfalldaten (Software Reliability Growth Models) [JM1998]. Software Reliability Growth Models gehen von Ist-Daten über Test- und Fehlerverlauf aus und liefern Prognosen über den weiteren Verlauf und die zu erwartende Zahl von Gesamt- und nicht entdeckten Restfehlern. Offen bleiben dabei Fragen nach der Verteilung und Bedeutung von Fehlern im Produkt und ihren Auswirkungen auf das Projekt und die spätere Anwendung.

Die vorhandenen Methoden werden überdies gar nicht oder nur sehr selten von KMUs eingesetzt [AOS2004]. Dies liegt vermutlich an Schwächen der Nachvollziehbarkeit von Modellen und Methodik sowie der fehlenden Integration in existierende SWE-Prozesse. TestBalance soll den Zugang erleichtern.

Die Projektpartner

Das Projekt TestBalance wird von den hier vorgestellten Partnern getragen:

imbus AG. Die imbus AG ist spezialisierter Lösungsanbieter für SW-Qualitätssicherung und –Test; ihre Kunden sind Softwareproduzenten verschiedenster Größen und Branchen. U.a. hat die imbus AG mit internationalen Partnern das Forschungsprojekts PETS durchgeführt und

das "Extended Partial Redundancy Model (EPR)" entwickelt, das auch zum Einsatz als Prognosewerkzeug im Umfeld systematischer Tests geeignet ist [PE2002].

SAP AG. SAP Research setzt auf eigenen Erfahrungen aus der Entwicklung pragmatischer Modelle für die Abschätzung von QS-Kosten auf und kann auf umfangreiche, zur Entwicklung und Validierung von Prognosemodellen geeignete Datensammlungen zugreifen. SAP arbeitet schon seit längerem an TestBalance zugrundeliegenden Aufgabenstellungen und hat handfeste Vorstellungen über Aufgabe und Lösungsanforderungen entwickelt.

T-Mobile. Der Unternehmensbereich „Customer Focused Product Integration“ der T-Mobile International AG &Co. KG verantwortet die Integration und den Systemtest von Komponenten, Services und Endgeräten vor Einführung in das T-Mobile-Wirknetz. Auch T-Mobile verfügt über umfangreiche Datensammlungen und hat sich anhand von Prozeßbewertungen klare Vorstellungen zur Aufgabe von TestBalance und Lösungsanforderungen erarbeitet.

FhG IESE. Das Fraunhofer Institut Experimentelles Software Engineering, ist ein Institut der angewandten Forschung, das einsatzreife Methoden und Techniken in die Software-Entwicklung industrieller Partner transferiert. Wichtige Arbeitsgebiete sind, Prozeßanalyse, -verbesserung und Coaching laufender Projekte. FhG IESE ist Experte für Qualitätssicherung und Produkt- und Prozeßbewertung mit Hilfe von Fehlerstrommodellen und Fehlerklassifikationen, die in vielen Industrie- und Forschungsprojekten eingesetzt und weiterentwickelt wurden (z.B., QUASAR, EMPRESS, CB-Testen, s. auch [BF2004]).

FhG ITWM. Das Fraunhofer Institut für Techno- und Wirtschaftsmathematik begreift Mathematik als Schlüsseltechnologie und entwickelt mathematische Modelle und Methoden für ein weites Aufgabenspektrum, das auch umfangreiche Kompetenzen in datenbasierten Ansätzen erfordert. U.a. hat FhG ITWM solche Techniken in Forschungsprojekten zur quantitativen Modellierung von Software-Entwicklungsprozessen und Fehlergehaltsschätzungen eingesetzt [SEV 2005], [SEV2003].

Das Projekt „TestBalance“

Die Genehmigung der Fördermittel für TestBalance durch den Projektträger erfolgte am 22. Mai 2006, so daß TestBalance zum 1. Juni die Projektarbeit operativ aufnehmen konnte. Das Kick-off-Meeting fand am 14. Juli 2006 in den Räumen der imbus AG statt, die das Projektkonsortium leitet. Danach wurde mit dem Aufbau der Projektinfrastruktur und der operativen Vorbereitung und Detailplanung begonnen.

Die beiden folgenden Kapitel geben zunächst eine Übersicht über das geplante Vorgehen im Projekt „TestBalance“ und skizzieren die Arbeitsteilung im Projekt.

Planung

Abbildung 3 stellt den geplanten Projektverlauf von TestBalance in der Übersicht dar.

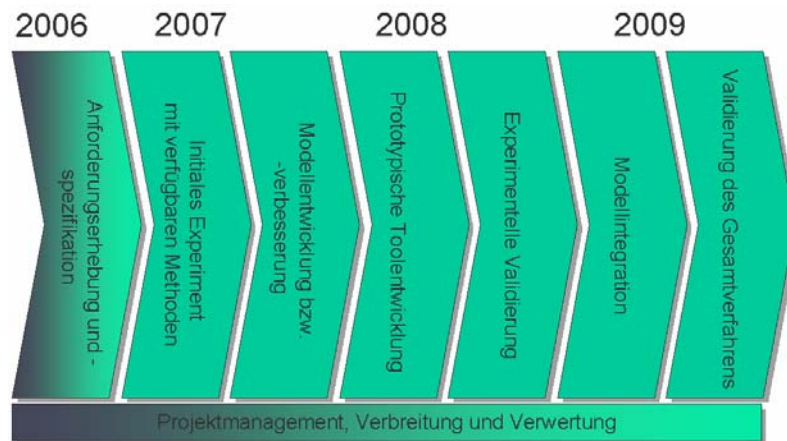


Abbildung 3: Umsetzungsplan

Im folgenden eine kurze Beschreibung der abgebildeten Aktivitäten:

Anforderungserhebung und –spezifikation: Wesentliche Ziele dieser Aktivität sind die systematische und detaillierte Erhebung der Anforderungen an TestBalance aus Sicht der Nutzer in Test- und Qualitätsmanagementprozessen und die praxismgerechte Priorisierung und Spezifikation der Leistungen der zu entwickelnden TestBalance Lösungen für die Nutzer. In diesem Zusammenhang wird auch der aktuelle Stand von Wissenschaft und Technik als Ausgangspunkt der Entwicklungsarbeit erfaßt und ausgewertet.

Initiales Experiment mit verfügbaren Methoden: Anschließend werden initiale Experimente mit im Stand von Forschung und Technik verfügbaren Methoden durchgeführt. Sie dienen dazu, die Leistungsfähigkeit und die Leistungsgrenzen dieser Methoden zu ermitteln und den erforderlichen Entwicklungsbedarf in TestBalance zu konkretisieren. In diesem Zusammenhang werden u.a. auch Fragen der Verfügbarkeit nutzbarer Daten und Informationen im operativen Umfeld geklärt.

Modellentwicklung und –verbesserung: Entwicklung bzw. Verbesserung der oben beschriebenen quantifizierenden Modelle, Anwendungsverfahren und Hilfsmittel zur Prognose von Fehlergehalt, QS-Effektivität, QS-Kosten und Fehlerfolgekosten auf Basis des Stands der Wissenschaft, Technik und Praxis.

Prototypische Toolentwicklung: Zur Vorbereitung auf den Einsatz der TestBalance Lösungen im industriellen Umfeld, werden je Einzelmodell prototypische Tools entwickelt, die zunächst an die Einsatzbedingungen bei den TestBalance-Anwendungspartnern angepaßt sind.

Experimentelle Validierung: Die den Einzelmodellen entsprechenden Prototypen, Methoden und Modelle werden experimentell validiert. Dazu werden die historischen Datenbestände der Anwendungspartner herangezogen und die TestBalance-Lösungskomponenten von ihnen in Projekten eingesetzt.

Modellintegration: Die Einzelkomponenten von TestBalance werden zunächst paarweise und dann zu dem vollständigen TestBalance-Gesamtverfahren integriert. Dabei werden Einzelmodelle und -komponenten bei Bedarf weiter aufeinander abgestimmt.

Validierung des Gesamtverfahrens: Als inhaltlicher Projektabschluß wird das TestBalance-Gesamtverfahren bei den Anwendungspartnern in Praxisprojekten validiert und konsolidiert. Die Validierung dient der Überprüfung der Erreichung der industriellen Ziele, der Erarbeitung von Demonstrationsmaterial für wissenschaftliche Zwecke und für die spätere interne und

externe Vermarktung der Projektergebnisse und zur Quantifizierung von Nutzen und Kosten von TestBalance.

Arbeitsteilung im Projekt

Der Umfang und die Heterogenität der Aufgabenstellung von TestBalance erfordern die Zusammenarbeit von Partnern unterschiedlicher Schwerpunkte und Kompetenzen in einem Projektkonsortium. Die beteiligten Partner nehmen im Projekt unterschiedliche Rollen ein:

Anwendungspartner liefern die Anforderungen und Hintergrundwissen aus der industriellen Anwendung, validieren Projektergebnisse, integrieren sie in die eigenen Prozesse und verbreiten sie im eigenen Unternehmen. Darüberhinaus stellen sie Praxisdaten für die Modellentwicklung und -evaluation bereit.

Forschungspartner steuern ihre umfassenden Kenntnisse über den aktuellen Stand von Forschung und Technik in der Modell- und Methodenentwicklung bei und entwickeln diesen weiter, um die komplexen Anwendungsaufgaben zu lösen.

Ein **Integrationspartner** im Projekt hat die Aufgabe, Gesamtvision und Gesamtlösung im Auge zu behalten und die Verwendbarkeit der Projektergebnisse, auch über den speziellen Bedarf der am Projekt beteiligten Partner hinaus, zu sichern.

Die **imbus AG** übernimmt als **Integrationspartner** die Gesamtkoordination des Projekts und ist als **Anwendungspartner** in der Rolle eines typischen KMU und **Forschungspartner** u.a. bei der Modellierung und Quantifizierung von Risiken sowie die der risiko- und anforderungsbasierten QS-Planung beteiligt.

FhG IESE übernimmt als **Forschungspartner** u.a. die Aufgaben der Entwicklung der Prognosemodelle für Fehlergehalt und QS-Effektivität und koordiniert die Modellintegration.

FhG ITWM übernimmt als **Forschungspartner** u.a. die Entwicklung statistischer, metrikbasierter Schätzungen von Softwarezuverlässigkeit und die der mathematischen Optimierung.

Die **SAP AG** repräsentiert als einer der beiden **Anwendungspartner** die Domäne von Unternehmenssoftware-Systemen und wirkt bei der Lösungsentwicklung mit. Als **Forschungspartner** verantwortet SAP das Modells zur Abschätzung des QS-Aufwands.

T-Mobile repräsentiert als **Anwendungspartner** das Gebiet der Qualitätssicherung sehr großer, komplexer und dynamischer IT-gestützter Systeme und wirkt bei der Lösungsentwicklung mit. T-Mobile übernimmt zudem die Qualitätssicherung der Tool-Implementierung von TestBalance.

Ausblick

Die an TestBalance beteiligten Partner planen, die Projektergebnisse auf vielfältige Weise zu nutzen:

- Die Anwendungspartner, SAP AG, T-Mobile und imbus AG werden TestBalance zum Eigenbedarf einsetzen. Dazu werden sie die Ergebnisse bedarfsgerecht weiter spezialisieren und durch Schulungen und Beratungsangebote für die hausinterne Verbreitung sorgen. Angestrebt wird dadurch auch die Verbesserung der Planbarkeit und Kalkulierbarkeit der hauseigenen Entwicklungsprozesse.
- Die beratungsorientierten Partner imbus AG und FhG IESE werden TestBalance-Ergebnisse nutzen, um ihr Beratungs- und Schulungsangebot zu Projekt- und Qualitätsmanagement für ihre zahlreichen Kunden in vielen Branchen im industriellen aber auch im

KMU-Bereich auszuweiten und dieses Angebot zukunftssicher zu gestalten. Insbesondere die imbus AG wird die prototypischen TestBalance-Tools zu marktfähigen Lösungen weiterentwickeln.

- Die beteiligten Forschungseinrichtungen FhG IESE und FhG ITWM werden mit Hilfe von TestBalance ihre fachliche Kompetenz weiter vertiefen und ihre Schwerpunkte sowohl im Software- und Quality-Engineering als auch die entsprechenden mathematisch-statistischen Kompetenzen weiter ausprägen.

Angesichts des Umfangs und des Variantenreichtums des Aufgabengebiets werden sowohl die Einzelkomponenten von TestBalance als auch das entwickelte Gesamtverfahren nach Abschluß des Vorhabens zwar einen bedeutenden Schritt jedoch keine insgesamt abgeschlossene Entwicklung darstellen.

Darüberhinaus wird der in der Einleitung beschriebene generelle Trend der Zunahme von Risiken aus Software-Fehlern kombiniert mit Mittelbeschränkungen für Qualitätssicherung weiterhin dazu beitragen, daß die Anforderungen an die Wirtschaftlichkeit der Qualitätssicherung weiter steigen. Daher kann generell damit gerechnet werden, dass der Bedarf nach auf TestBalance basierenden Lösungen auch nach Projektabschluß zunimmt.

Literaturhinweise

- [ASQF2002] Arbeitskreis Software-Qualität Franken e.V.: <http://www.asqf.de>
- [AOS2004] Stand der Praxis von Software-Tests und deren Automatisierung, Armbrust, Ove; Ochs, Michael A.; Snoek, Björn, IESE-Report, 093.04/D
- [BF2004] A Measurement Framework for Software Inspections in the Quasar Context, Bernd Freimut IESE-Report, 118.03/E,
- [BS2000] Software economics: a roadmap, Barry W. Boehm and Kevin J. Sullivan, ICSE '00: Proceedings of the Conference on The Future of Software Engineering, 2000
- [CMM1993] Key Practices of the Capability Maturity Model, Technical Report, CMU/SEI-93-TR-025, 1993
- [CO2000] Software Cost Estimation with Cocomo II, Barry Boehm et. al, 2000, Prentice Hall PTR
- [FP2000] Function Point Analysis: Measurement Practices for Successful Software Projects, David Garmus, David Herron 2000, Addison-Wesley Professional
- [GM1978] A Controlled Experiment in Program Testing and Code Walk-throughs/Inspections. Glenford J. Myers, Communications of the ACM, 21(9):760–768, September 1978
- [GT2002] The Economic Impacts of Inadequate Infrastructure for Software Testing, Gregory Tassej, NIST-Report 03/2002
- [JM1999] Software Reliability Engineering, John Musa, McGraw-Hill 1999
- [LM2001] Quantifying the Reliability of Embedded Systems by Automated Analysis. Peter Liggesmeyer, Oliver Maeckel: DSN 2001: 89-96,

- [PE2002] Prediction of software Error rates based on Test and Software maturity results (<http://www.ist-pets.com>)
- [SEV2003] Blocked neural networks for knowledge extraction in the software development process. P. Lang, A. Sarishvili, A. Wirsen, 2003, ITWM Report Nr. 56.
- [SEV2005] Simulation based software process modelling and evaluation. J. Münch, P. Lang, et. al. in publishing, erscheint in “Advances in SE&KE” 2005.
- [SS2002] An Empirical Evaluation of Six Methods to Detect Faults in Software, S. S. So, S. D. Cha, T. J. Shimeall, Y. R. Kwon. Software Testing, Verification and Reliability, 2002
- [TB2006] Projektantrag zum Vorhaben “TestBalance – Methoden und Werkzeuge zur Optimierung des Erfolgs von Qualitätssicherungsmaßnahmen”, Version 23.5.2006
- [WH1976] An Experimental Analysis of Program Verication Methods. W.C. Hetzel. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 1976