

CBTesten

Verifikation und Validation in objekt-orientierten und komponenten-basierten Software-Entwicklungen

Dr. Axel Kalenborn
Ralf Heid
Stefan Maratzki
ICTeam Internet Consulting AG
Paulinstraße 27
D-54292 Trier

Michael Ochs
Björn Snoek
Dr. Gerd Groß
Fraunhofer-IESE
Sauerwiesen 6
D-67661 Kaiserslautern

Dr. Martin Verlage
Ulrich Maurer
MARKET MAKER Software AG
Karl-Marx-Straße 13
D-67655 Kaiserslautern

Kurzfassung

Komponenten-Technologien und insbesondere die Möglichkeit der Mehrfachnutzung von Komponenten bieten für KMUs hervorragende Möglichkeiten. Zum einen können auf Basis existierender Softwarekomponenten neue Produkte für Individualprojekte oder Nischenmärkte schnell und mit hoher Qualität entwickelt werden, auf der anderen Seite ermöglicht die Vermarktung von Softwarekomponenten für größere Systeme eine Chance auf Mehrfachnutzung von vorhandenem Know-how. Aber gerade die Mehrfachverwendung stellt auch die besondere Schwierigkeit bei der Entwicklung und Verwendung von Komponenten dar. Variabilitäten und Komplexitäten beim Einsatz der Komponenten erfordern ein sorgfältiges Ausführen von Tests, die bei manueller Durchführung zeitaufwendig und teuer sind. Ein Problem liegt im Fehlen automatisierbarer Test-Verfahren. Mit diesem Problem beschäftigt sich das Projekt CBTesten. Ein Methodenansatzes für eine effiziente Durchführung von Komponenten- und Systemtests soll entwickelt werden. Dazu werden OpenSource-Ansätze mit neuesten Forschungsergebnissen, dem „Built-in Testen“, vereint und eine innovative Plattform für verschiedenartige Tests bis hin zu automatisierten Regressionstests umgesetzt.

1 Einleitung und Vorstellung des Themenkomplexes

1.1 Anforderungen moderner Entwicklungs- und Testkonzepte

Große komponenten-basierte Applikationen dominieren in zunehmendem Maße die Software-Entwicklung [At+01]. Eine Konsequenz daraus, die bereits heute beobachtet werden kann, ist die verstärkte Nachfrage nach Komplettlösungen Dritter, die in eigene Applikationen eingebettet werden können. Dies ist eine Erweiterung des sog. „Sub-Contracting“-Prinzips, bei dem Komponenten-Anbieter typische domänen-spezifische Lösungen implementieren und anbieten. Dieses Vorgehen wirft insbesondere die Frage auf, wie Applikationen integriert werden können, die auf unterschiedlichen Technologien basieren, unter gleichzeitiger Qualitätsverbesserung, die diese Technologien versprechen. Insbesondere KMUs¹ sind von einfach einzuführenden und leicht zu beherrschenden Entwicklungs- und Qualitätssicherungsmethoden abhängig, denn

¹ KMUs = Kleine und mittelständische Unternehmen

- die Kräfte werden im Wesentlichen auf die Produktentwicklung und deren Vermarktung konzentriert,
- die Abhängigkeit von wenigen (Haupt-)Auftraggebern ist hoch, so dass Verzögerungen in der Produkt- bzw. Projektabnahme kritisch sind,
- die Vorgaben der Auftraggeber hinsichtlich der Entwicklungsprozesse, entwicklungsbegleitenden Dokumentation und Produktions- und Testumgebung stellen kleinere Unternehmen vor große Herausforderungen.

Die Komplexität heutiger Software-Systeme kann nur dann beherrscht werden, wenn durch ingenieurmäßige Ansätze die Struktur der Systeme im Rahmen der Möglichkeiten einfach gehalten wird. Objektorientierte und komponentenbasierte Entwicklungsmethoden und Programmiersprachen haben ihre Eignung zur Beherrschung der Komplexität gezeigt. Vor allem die Komponentenorientierung verspricht signifikante Verbesserungen bei der Wiederverwendung bereits existierender Lösungen, sowie bei der Verkürzung der Entwicklungszyklen.

Wenn nun Applikationen verstärkt aus existierenden Teilen erstellt werden, verschiebt dies den Entwicklungsaufwand von der Erstellung der Funktionalität hin zu Integration existierender Funktionalität. Auf der einen Seite wird also Entwicklungsaufwand gespart, diese Einsparung wird aber auf der anderen Seite durch notwendige Arbeiten bei der Integration, zum Beispiel beim Testen, wieder aufgezehrt.

Ein tatsächlich erreichter Vorteil bei der Anwendung von Komponententechnologie hängt also von zwei Faktoren ab: (1) eingesparter Entwicklungsaufwand und (2) verbrauchter Integrationsaufwand. Moderne Software-Entwicklungsansätze konzentrieren sich auf die Bereitstellung von Komponenten, die für verschiedene Produkte und Produktvarianten eingesetzt werden können und unterstützen somit die Einsparung von Entwicklungsaufwand. Produkte entstehen dabei durch Konfiguration und Anpassung der Komponenten. Durch die späte Integration und die parallele Nutzung in verschiedenen Systemkontexten ergeben sich für die Validierung hierbei allerdings folgende Probleme:

- Für den Komponententest müssen zahlreiche potenzielle Einsatzszenarien überprüft werden, wodurch der Testaufwand überproportional ansteigt. Eine vollständige Validierung ist selten möglich, da durch die potentiellen Konfigurationen der Testraum sich potenziert.
- Für den Systemtest, der auf den konfigurierten und integrierten Komponenten beruht, müssen bei jeder Änderung einer Komponente die Tests wiederholt werden, auch dann, wenn die Änderung die entsprechende Systemvariante nicht betreffen sollte. Hier sind effiziente Methoden bei Regressionstests auf Systemebene gefragt.

Diese Probleme werden zusätzlich verstärkt, wenn Komponenten Dritter oder Web-Dienste in bestehende komponenten-basierte Systeme integriert werden sollen. Dabei muss zunächst ein Benutzungsprofil zur Testfallreduzierung erstellt werden, und teilweise müssen interne Zustände, die sich je nach Klient ändern können, beim Integrationstest und Regressionstest berücksichtigt werden. Dabei stellen sich diese besonderen Herausforderungen bzgl. der Validierung von komponenten-basierten Systemen für die beteiligten Industriepartner wie im Folgenden beschrieben dar.

1.2 Problemlage bei den Industriepartnern im Konsortium

1.2.1 ICTeam Internet Consulting AG

Komponententests werden derzeit bei ICTeam auf verschiedene Weise durchgeführt. Es werden dabei Tests der Frontends und Backends unterschieden, da dort jeweils unterschiedliche

Technologien zum Einsatz kommen, die aufgrund von differenzierten Anforderungen verschiedene Testszenarien erfordern. Die Backend-Systeme basieren vollständig auf Java und werden hauptsächlich beim täglichen Product-Build zur Compilezeit vollständig überprüft. Dabei wird ein Staging durchgeführt, wobei jede Komponente mehrfach überprüft wird, bevor sie zum Kunden ausgeliefert wird. Logische Tests werden über die Frontends durchgeführt, die die Backend Komponenten nutzen. Hier werden hauptsächlich manuelle Tests durchgeführt, die diese in regelmäßigen Abständen anhand fest definierter Fälle durchführen. Schwierigkeiten bereiten hier insbesondere Seiteneffekte in der Programmierung, die Fehler in Softwarebereichen verursachen, die nicht direkt von Änderungen oder Erweiterungen betroffen sind.

1.2.2 MARKET MAKER Software AG

Momentan werden bei der MARKET MAKER Software AG Komponententests auf der Basis der Frameworks „JUnit“ (siehe <http://www.junit.org>) eingesetzt. Die Erfahrungen für die Eigenentwicklung von Komponenten sind durchaus positiv, so dass diese Ansätze für die Weiterentwicklung verwendet werden sollen. Der Code wird um Testmethoden erweitert, um einzelne Klassen oder Komponenten zu testen. Diese Tests können umfangreich ausgearbeitet werden, etwa wenn Datenbanken mit Kundendaten erschaffen, gefüllt und dann auf Konsistenz hin überprüft werden müssen. Problematisch ist bei den Komponententests die Prüfung auf Überdeckung sowie die Anwendbarkeit der Tests für verschiedene Varianten des Systems. Da die Komponenten von MARKET MAKER auch in Fremdsystemen auf externen Plattformen zum Einsatz kommen, ist die Überprüfung der Eigenschaften vor Ort durch den Kunden ebenfalls ein Thema. Im Falle einer Störung dient dies zur schnellen Diagnose. Bei der Installation des Systems kann hier eine Prüfung auf Konformität mit den Annahmen bezüglich der spezifischen Umgebung für die Komponente stattfinden.

1.3 Zielsetzung des Projektes

Ziel dieses Forschungsprojektes ist die Entwicklung und der Transfer einfacher und KMU-tauglicher Konzepte zur Validierung von komponenten-basierten Applikationen. Dabei werden sowohl funktionale als auch temporale Eigenschaften betrachtet. Im Projekt ist das Hauptziel die Ermöglichung eines modularen Regressionstests zeitkritischer Informationssysteme, d.h. eine Überprüfung sowohl zeitlicher als auch funktionaler Eigenschaften. Dabei hat der modulare Regressionstest zum Ziel, die Ausführung von Testfällen unter gegebenen Randbedingungen (z.B. Abhängigkeiten im System) zu minimieren, um das Testen möglichst kostengünstig und Ressourcen schonend zu gestalten. Dabei werden sowohl methodische wie auch werkzeugtechnische Gesichtspunkte betrachtet und ein möglichst hoher Automatisierungsgrad bei der Analyse des Systems und der Auswahl der Testfälle angestrebt. Teilaspekte des modularen Regressionstests sind u.a.

- (Automatisierte) Analyse der Abhängigkeiten zwischen Komponenten (z.B. mittels Source Code-Analyse) und Identifikation der zu testenden Interaktionen im System
- Integration von BIT-Artefakten über verschiedene Wege (z.B. direktes Encodieren, Byte Code-Injektion, Generierung von Stubs)
- Initialer Aufbau / Anpassung einer Testfalldatenbank
- Aktualisierung der Testfalldatenbank nach Modifikation
- „Filterung“ von Testfällen je nach Modifikationsart
- Logging der Testergebnisse
- Modularer Regressionstest für funktionales Testen und Performanzanalyse für die Domäne der *zeitkritischen Informationssysteme*

Dabei stehen drei Hauptziele im Vordergrund:

1. Modellbasierte Spezifikation des Systems als Basis für die Testfall-Ableitung
2. Systematische Ableitung von Testfällen und Aufbau eines Testfall-Repositories
3. Ausführung von Testfällen während der Entwicklung und im Zielsystem

2 Projektstatus

2.1 Problemlage

Bei den vom Projekt adressierten Themen müssen vor allem die Adaption der existierenden Ansätze entsprechend der besonderen Belange, die in kleinen und mittleren Unternehmen vorherrschen, betrachtet werden. Diese Konzepte müssen so einfach sein, dass sie inkrementell in existierende Prozesse eingebunden werden können, um möglichst wenig bestehende Arbeitsabläufe in solchen Unternehmen zu verändern. Es müssen spezielle Entwicklungstechnologien in den KMUs (z.B. JavaBeans, Web-Dienste) betrachtet werden, und im Idealfall eine optimale Mischung aus eigener KMU-spezifischer methodischer und technologischer Unterstützung für einen existierenden Ansatz gefunden werden.

Um die Steigerung der Entwicklungseffizienz bzw. der Qualität überprüfen zu können, werden darüber hinaus noch Kriterien (Messwerte) benötigt, die Aussagen über die Auswirkungen der geplanten Veränderungen machen können. Dies sind zum Beispiel:

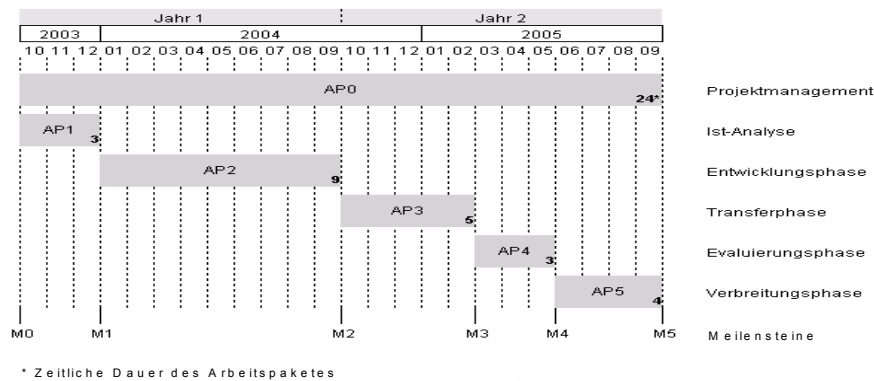
- Aufwand der Testentwicklung
- Aufwand der Komponenten-Integration vorher/nachher
- maximale Antwortzeit

Um Änderungen „verträglich“ in kleinen Unternehmen einführen zu können, ist es notwendig, möglichst kleine Technologieschritte anzustreben. Gerade KMUs müssen sich wegen ihrer begrenzten Ressourcen auf die Produkte konzentrieren. Prozesse werden daher und wegen der überschaubaren Größe der Teams im vollen Umfang kaum beachtet oder gepflegt. Bei der Einführung ist zu beachten, dass jeder einzelne Schritt auch in seinen Auswirkungen, soweit möglich und effektiv, überprüft werden kann (Messbarkeit). Ein weiterer wichtiger Schritt bei der Einführung von Änderungen ist die Durchführung von Machbarkeitsstudien für die Einführung der zuvor definierten Änderungspakete. Dabei müssen als sog. Eingangs-Kriterien mögliche Schwierigkeiten bei der Einführung, wie z.B. aktueller Kenntnisstand der Mitarbeiter und benötigter Schulungsaufwand, verwendete Implementierungstechnologien etc. berücksichtigt werden. Diese Studien sollen nach Möglichkeit Risiken bei der Einführung der neuen Technologien aufzeigen (z.B. unverhältnismäßiger Trainingsaufwand der Mitarbeiter oder notwendige Werkzeugunterstützung) und gegebenenfalls frühzeitig Alternativen zur Vermeidung dieser Risiken untersuchen und vorschlagen.

2.2 Projektplan

Der in Abbildung 1 dargestellte Projektplan stellt den zeitlichen Ablauf der verschiedenen Arbeitspakete Meilensteinen im Projekt CBTesten dar.

Abbildung 1. Darstellung des Projektzeitplans mit Meilensteinen.



In der folgenden Tabelle 1 werden die in Abbildung 1 aufgeführten Arbeitspakete näher beschrieben.

Tabelle 1. Kurzbeschreibung der Arbeitspakete des Projektes.

AP	AP Name	AP Kurzbeschreibung
1	Ist-Analyse	Mit Unterstützung des Fraunhofer IESE wird eine Analyse des aktuellen Stands von Entwicklungs- und Qualitätssicherungsprozessen bei den industriellen Projektpartnern durchgeführt und Verbesserungsvorschläge, die in AP2 und AP3 umzusetzen sind, erarbeitet.
2	Entwicklung	In diesem Arbeitspaket werden unter der Leitung des Fraunhofer IESE geeignete Lösungsstrategien für die während AP1 identifizierten Probleme entwickelt. Diese umfassen bzgl. der zu verbessernden Testaktivitäten sowohl funktionale Aspekte wie auch Performanzanforderungen. Ein wichtiges Kriterium für die Entwicklung der Lösungen ist dabei die Machbarkeit, d.h. für jeden Ansatz werden die damit verbundenen Schwierigkeiten und Risiken bei der Einführung und während des Betriebs innerhalb der KMUs abgeschätzt, und schließlich geeignete, d.h. KMU-taugliche Ansätze ausgewählt. AP2 schließt neben den eigentlichen Forschungs- und Entwicklungstätigkeiten notwendige Weiterbildungsmaßnahmen der beteiligten KMUs mit ein. Dazu werden Schulungen je nach Bedarf durch das beteiligte Forschungsinstitut gehalten.
3	Transfer	Nachdem in AP2 Forschungsergebnisse in Hinblick auf die Verbesserung von Testverfahren für komponenten-basierte Applikationen erarbeitet worden sind, erfolgt nun deren Umsetzung in den KMUs. Dazu werden entwickelte Konzepte mit ausgewählten Technologien direkt bei den beteiligten Industrieunternehmen implementiert, wobei der Forschungspartner begleitend zur Seite steht. Implementierung bedeutet in diesem Zusammenhang, dass die entwickelten Konzepte innerhalb der KMUs eingeführt werden. Dies umfasst u.a. die Anpassung oder Umstellung existierender Prozesse, die Anpassung bzw. Neuentwicklung u.a. von Testfällen und Testcode, Einführung neuer Techniken, wie z.B. suchbasierte Testverfahren, Entwicklung von zusätzlichen Hilfsprogrammen
4	Evaluierung	In AP4 erfolgt ein Vergleich zwischen den in AP3 erzielten Verbesserungen und den in AP1 im Rahmen des Maßnahmen-Katalogs festgelegten Zielvorgaben, d.h. es wird ein „Vorher-Nachher Vergleich“ durchgeführt. Es wird also ermittelt, ob die eingeführten Methoden, Techniken und Technologien während des Einsatzes bei den beteiligten KMUs den gewünschten Erfolg gebracht haben, oder ob noch Verbesserungspotential besteht. Damit verbunden ist die Bewertung der entwickelten Konzepte und die Erfassung der Auswirkungen nach Einführung bei den KMUs.
5	Verbreitung	Im letzten Arbeitspaket erfolgt die Verbreitung der erzielten Ergebnisse in Form von z.B. Schulungsunterlagen, Konferenz- und Workshopbeiträgen oder wissenschaftlichen Artikeln.

2.3 Aktueller Projektstatus

Das Arbeitspaket 1 – „Ist-Analyse“ ist vom CBTesten Konsortium erfolgreich beendet worden [CBT04]. Aktuell befindet sich das Projekt in der Durchführung von Arbeitspaket 2 – „Entwicklungsphase“, in dem, wie in Tabelle 1 beschrieben, geeignete Lösungsstrategien für die Zielsetzung erarbeitet, initial erprobt und hinsichtlich ihrer Machbarkeit untersucht werden. Die wesentlichen Ergebnisse des ersten Arbeitspaketes „Ist-Analyse“ sind die Identifikation der Problemlage bei den Industriepartnern sowie eine Fokussierung und Präzisierung der Zielsetzung

des Projektes hinsichtlich Regressionstests². Die Hauptprobleme hinsichtlich Testen und Qualitätssicherung, die bei den Partnern identifiziert wurden, sind in Tabelle 2 anonymisiert dargestellt [CBT04].

Tabelle 2. In Arbeitspaket 1 identifizierte Hauptprobleme bei den Industriepartnern (anonymisiert).

Hauptprobleme bei den Industriepartnern
Testen bei der Backend-Entwicklung wird nicht durchgeführt (Integrationstests)
Backend-Tests erfolgen nur über Ausprobieren des GUI am Frontend (Regressionstests)
Anforderungen in Angebots- und Feinkonzeptdokumenten nicht ausreichend präzise formuliert
Schwierigkeiten beim Regressionstest der Systeme. (Durch einen hohen Aufwand für die Durchführung komplexer Regressionstests werden diese zum Teil nicht durchgeführt. Eine hohe Abdeckung der Tests sollte deshalb auf einem hohen Automatisierungsgrad beruhen)
Schwierigkeiten beim Testen von Komponenten in unterschiedlichen Umgebungen (Systemtest)
Unklare/aufwendige Verwaltung von Referenzergebnissen (Soll-Werte) und Testfällen

Damit war eine Fokussierung und Präzisierung der Zielsetzung des Projektes möglich (s.a. Abschnitt 1.3). Die vorliegenden Ergebnisse des aktuell bearbeiteten Arbeitspaketes 2 – „Entwicklungsphase“ sind durchgeführte Schulungen zu den in Arbeitspaket 1 identifizierten Basistechnologien, ein Grundlagendokument zum Thema Testen, ein methodischer Leitfaden zur Erstellung von Artefakten für Built-in Testen zum Zwecke von Performanz- und Regressionstests und eine Übersicht von unterstützenden Tools für die verschiedenen Zwecke (vgl. Beschreibung unten). Laufende Aktivitäten sind die Bewertung der verschiedenen Ansätze für Performanz- und Regressionstests hinsichtlich ihrer Machbarkeit anhand von Kriterien wie z.B. Erstellung-/Entwicklungsaufwand, Flexibilität, Toolsupport, etc.

2.4 Aktuelle Ergebnisse und Konzepte

Die zu den Basistechnologien durchgeführten Schulungen behandelten, gemäß den Ergebnissen von AP1 – „Ist-Analyse“, in jeweils eintägigen Schulungen die Themen

- Anforderungsentwicklung / Requirements Engineering
- Unified Modeling Language (UML)
- Built-in Testen (BIT)

Das erstellte Grundlagendokument zum Testen fasst die wichtigen Themen aus Sicht der Industriepartner in Form von Weiterbildungsbedarfen zusammen. Dazu gehören zur Unterstützung der Schulungen eine Übersicht über modellbasierte Spezifikationen mit der UML, die Ableitung von Testfällen für Black-Box und White-Box Testen, die Planung von Tests sowie das Thema Built-in Testen und eine Übersicht von verschiedenen Testwerkzeugen (siehe unter anderem [Bin00], [Bei90]). Der methodische Leitfaden zur Erstellung bzw. Entwicklung von Built-in Tests beinhaltet eine Folge von Aktivitäten, die entlang des allgemeinen Testprozesses [Spi03] orientiert ist. In Abbildung 2 sind die oben erläuterten acht Schritte entlang der Kette von Testaktivitäten von „Planung“ bis „Auswertung“ im allgemeinen Testprozess aufgereiht. Die einzelnen Schritte werden in Abbildung 2 entlang des allgemeinen Testprozesses dargestellt und werden in Tabelle 3 kurz erläutert.

Abbildung 2. Darstellung des methodischen Leitfadens zur Erstellung von Built-in Tests.

² Der Begriff „Regressionstest“ bedeutet in diesem Zusammenhang, dass nach der Modifikation oder dem Austausch einzelner Komponenten eine geeignete Auswahl aus den vorhandenen Testfällen getroffen werden muss, um das Gesamtsystem erneut im Hinblick auf Korrektheit und Zeitverhalten zu testen.

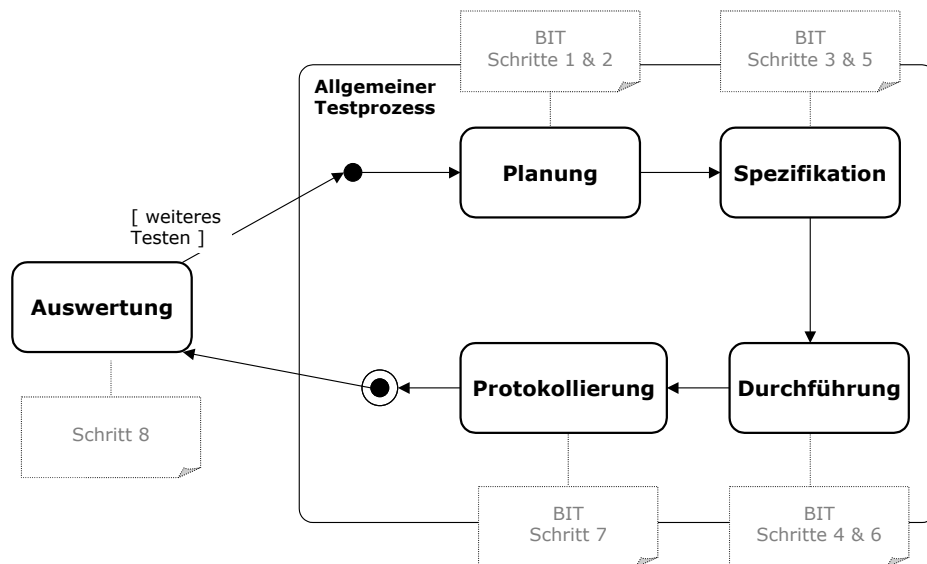


Tabelle 3. In Arbeitspaket 1 identifizierte Hauptprobleme bei den Industriepartnern (anonymisiert).

BIT Entwicklungsschritt	Kurzbeschreibung des Entwicklungsschrittes (ohne Input, Output, Rollen, Templates, Tools, Guidelines)
Schritt 1	Identifikation der zu testenden Komponenten-Interaktionen. Hierzu wird ein Modell der Systemarchitektur benötigt, und Kriterien, anhand derer entschieden werden kann, welche Interaktionen auf welche Weise (Szenarien) durch Tests überprüft werden.
Schritt 2	Definition und Modellierung der Testarchitektur. Hierbei wird entschieden, welche Komponenten Tester-Komponenten und Test-Schnittstellen erhalten und wie diese zusätzlichen Artefakte miteinander verbunden werden. Die Testarchitektur kann beim Built-in Testen als Teil der eigentlichen Systemarchitektur verstanden werden.
Schritt 3	Spezifikation der Test-Schnittstellen der identifizierten Komponenten in der gewählten Spezifikationsnotation (z.B. UML), falls diese Komponenten erweitert werden können (Server-Artefakte). Hier werden die externen Zugriffsmechanismen, die das Testen unterstützen sollen für jede Komponente separat definiert.
Schritt 4	Implementierung der Test-Schnittstellen in der gewählten Implementierungstechnologie (z.B. Java), falls dies möglich ist.
Schritt 5	Spezifikation der Tester-Komponenten in der Spezifikationsnotation (Client-Artefakte). Hier werden die Tester-Komponenten entsprechend der verwendeten Testkriterien geplant und modelliert.
Schritt 6	Implementierung der Tester-Komponenten in der gewählten Implementierungstechnologie.
Schritt 7	Integration der Komponenten, und Ausführen und Protokollieren der Tests. Dieser Schritt beinhaltet die eigentliche Ausführung der entwickelten Tests, als Teil der Komponenten-Integration (Deployment).
Schritt 8	Auswertung der Testergebnisse und Entscheidung über die weitere Vorgehensweise im Testprozess.

3 Erfahrungen und Bewertungen

Innerhalb des oben beschriebenen methodischen Leitfadens sind verschiedene grundsätzliche Szenarien je nach Zweck und Ziel der zu entwickelnden Tests umsetzbar. Von den jeweiligen Szenarien und Zielen der Tests hängt auch die Wahl der anzuwendenden Testtechnologie ab. Hier existieren verschiedene Möglichkeiten, die (speziell bei Web-basierten Systemen) grundsätzlich nach den Kriterien „Ort“ (Frontend/GUI vs Backend), „Art“ (funktional vs. Performanz), „Flexibilitätsbedarf bei der Testerstellung“ (hohe vs. niedrige Flexibilität), „Implementierungsaufwand“ (verfügbarer Aufwand „niedrig“ bis „hoch“), „Zielkomponenten“ (Frontend vs. Backend) klassifiziert werden können. Dabei besteht die Möglichkeit, die Built-in

Tests mittels verschiedener Implementierungstechnologien umzusetzen. Einige der Technologien sind in Tabelle 4 hinsichtlich der oben genannten Kriterien vergleichend dargestellt.

Tabelle 4. Übersicht verwendbarer Technologien und deren Ausprägungen bzgl. Der Kriterien.

Kriterium	HTTPUnit	JMeter	Byte-Code Injektion	Built-in Tests (im Code)	Stubs
Ort des Tests	Frontend	Frontend	Backend	Backend	Backend
Art des Tests	Funktionalität	Performanz (z.B. kombiniert mit Stubs)	Funktionalität & Performanz	Funktionalität & Performanz	Funktionalität & Performanz
Flexibilität bei Testerstellung	Hoch	Mittel	Niedrig – Mittel	Hoch	Niedrig
Aufwand für Testerstellung	Mittel – Hoch	Niedrig	Niedrig – Mittel	Niedrig – Hoch	Niedrig
Ziel-komponenten	Frontend & Backend	Frontend & Backend	Backend	Backend	Backend & Frontend

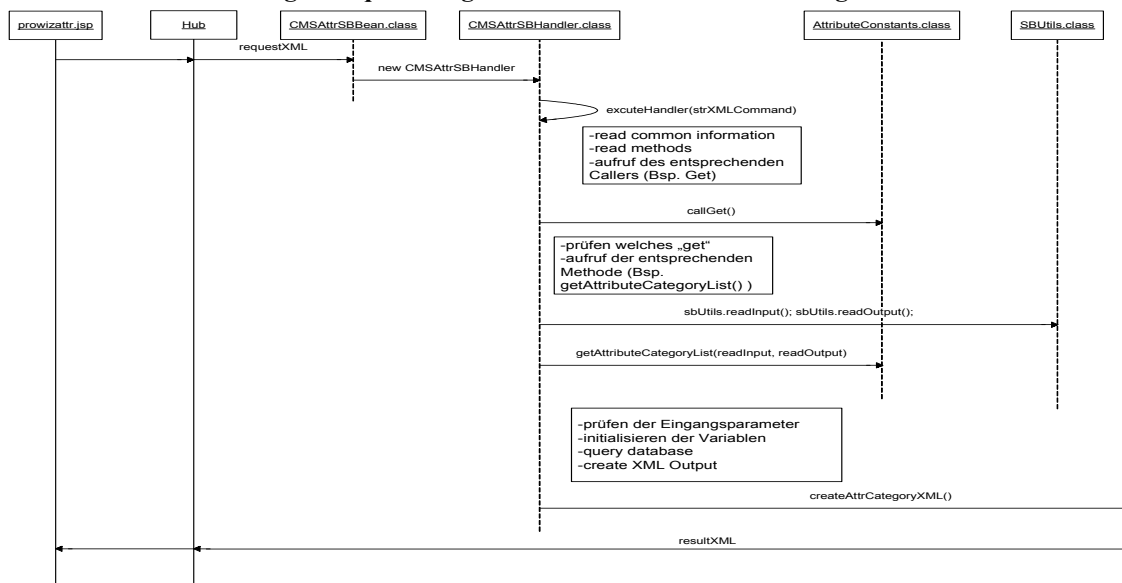
Tabelle 4 zeigt, dass sowohl Frontend als auch Backend von zeitkritischen Informationssystemen als Ansatzpunkt für Regressionstests verschiedener Art technologisch abgedeckt werden können. HTTPUnit, als Vertreter der Unit-Test Tools (wie auch DUnit, JUnit, etc.), ermöglicht die Erstellung und Durchführung von Tests für Web-basierte Systeme auf HTML-Basis hinsichtlich deren Funktionalität. Grundsätzlich ist aber auch ein Einsatz für Performanztests denkbar. JMeter dagegen zielt primär auf die Überprüfung der Performanz von Systemen am Frontend ab. JMeter liefert eine Umgebung zur Eingabe, Ausführung und Protokollierung, d.h. Aufzeichnung von Zeitverhalten wie z.B. Antwortzeiten unter Last des Systems oder Anzahl abgearbeiteter Anfragen pro Zeit. Eine Byte-Code Injektion (z.B. mittels des Tools BCEL) ermöglicht es Tests in das System zu injizieren. Dabei wird der injizierte Code in Form eines Wrappers bei Java im JAR-File integriert und steht dann zur Testlaufzeit des Systems zur Verfügung. Der Weg des Built-in Testens [Gro02a], [Gro02b] erfordert die Codierung von Tests in die als Testsystem instanziierten zu testenden Komponenten bzw. Klassen. Zum Beispiel stellen bei ICTeam insbesondere Funktionstestes unter Last sowie Systemtests in unterschiedlichen Umgebungen die beiden dominanten Problemklassen dar. Im ersten Bereich gilt es, einen kombinierten Test zu definieren, der funktionale Tests bei einer gleichzeitigen Dauerlast auf dem System möglich macht und damit funktionale Anomalien unter Stress aufdeckt. Im zweiten Bereich gilt es ein möglichst automatisiertes Testverfahren zu entwickeln, das ein System in einer kundenspezifischen Umgebung auf Anomalien und funktionale Fehler hin überprüft. Letzteres Szenario gilt auch für den Partner MARKET MAKER, da auch hier die reibungslose Lauffähigkeit der Systeme in der Zielumgebung des Kunden gewährleistet und dies überprüfbar sein muss. Für beide Bereiche gilt, dass nur eine weitgehende Automatisierung eine effizientere Umsetzung ermöglicht und damit Kosten einspart. Ein Beispielszenario für Regressionstests des Content Management Systems (CMS) des Partners ICTeam wird im Folgenden kurz dargestellt. Dabei besteht das System aus drei wesentlichen Architektur-Layers: (1) HTML-basiertes Frontend als Benutzerschnittstelle, (2) JSP-basierte Zwischenschicht, die die Kommunikation zwischen Front- und Backend regelt sowie (3) Java-basiertes Backend.

Eines der Testsznarien setzt im Bereich Attribut-Handling in der zweiten Schicht an. Das Attribut-Handling besitzt dabei die im UML-Sequenzdiagramm beschriebenen Interaktionen. Das gewählte Beispieltestszenario (Built-in Testen über encodierte Tests) setzt an einem JSP-Artefakt (z.B. „prodwizattr.jsp“, vgl. Sequenzdiagramm in Abbildung 3) an und hat die in Tabelle 5 in Kurzform beschriebene Spezifikation (Angaben als UseCase entsprechend der Spezifikation in Abschnitt 7.1.4 [CBT04]).

Tabelle 5. Use Case / Spezifikation des Szenarios für prodwizattr.jsp.

UseCase 2	Neuer Bibliotheksordner
Beschreibung	Ein neuer Ordner wird im Bereich Bibliothek erzeugt
Scope / Level	Administrationsfunktion
Aktoren	Administratoren und andere Benutzer mit Administrator-Rechten
Trigger	Aufruf einer URL
Stakeholder	Kunde und Dienstleister
Precondition	Installiertes ICCContent-System und System gestartet und als Administrator angemeldet
Postcondition Success	Erscheinen eines neuen Ordners mit der Bezeichnung 'new Folder' unterhalb der Root-Ordners im Menü (links) und Editierfunktionsansicht im Content-Bereich der Website
Postcondition Failure	-der Eintrag wurde nicht angelegt (erscheint nicht im Menü) -fehlende Editierfunktionsansicht (rechts) -fehlende Teile der Editierfunktionsansicht (rechts)
Basic Course	Kontextmenü durch rechte Maustaste im Bereich des Root-Ordners und darin Auswahl des Menüpunktes 'neu'
Alternative Course	-
Exceptions	Fehlermeldungen 404 und 500
Associations	Usecase 1 (= 'Login')
Non Functional Req's	-

Abbildung 3. Sequenzdiagramm für das Attribut-Handling des CMS.



4 Ausblick

Ein erwartetes Ergebnis des Vorhabens CBTesten ist die methodische und technologische Befähigung von KMUs zu einer zielgerichteten Erstellung und Durchführung von Regressionstests ihrer Systeme hinsichtlich Funktionalität und Performanz. Dabei sind verschiedene technologische Wege zur Realisierung der Tests möglich, wie z.B. (1) Frontend-to-Backend Tests mittels JMeter und ggf. eine Schicht von Stubs um bestimmte temporale oder funktionale Eigenschaften des Systems auch außerhalb der Produktionsumgebung zu überprüfen, (2) Simulation von Fehlbenutzungen der Software zur Sicherstellung von Laufzeitstabilität, (3) Simulation von wiederkehrenden häufigen Benutzungsprofilen zur Qualitätssicherung von erweiterter oder geänderter Funktionalität oder (4) Lastverhalten des Systems und Überprüfung der Performanzanforderungen. Ein weiteres Ergebnis wird sein, dass die verschiedenen Wege

Built-in Tests zu implementieren hinsichtlich ihres Umsetzungsaufwandes und ihrer Komplexität transparent werden. Diese Erfahrungen können in Vergleichsform andere KMUs bei der Auswahl der geeigneten Technologien unterstützen. Ein wesentlicher Bestandteil bei den Tests wird auch die Überprüfung nicht-funktionaler Eigenschaften der Komponenten sein. Bezieht sich der Lasttest auf die Performanz des Systems, so müssen weitere Eigenschaften geprüft werden. Hierzu zählt vor allem die Analyse struktureller Eigenschaften der Komponentenarchitektur, wie z.B. geringe Kohäsion und hohe Kopplung. Die Strukturinformationen werden auf Basis von Laufzeitinformationen während der Regressionstests erhoben. Hierzu ist bereits bei MARKET MAKER die Software MSystem des Fraunhofer-IESE im Einsatz, welches aber für die Aspekte von Tests noch erweitert werden muss. Vor allem die automatische Ableitung von Stubs/Mocks ist ein wesentlicher Bestandteil bei der Kopplung des Ansatzes mit der Methode aus CBTesten.

Referenzen

- [At+01] Atkinson, C. et al.: Component-based Product Line Engineering with UML. London: Addison-Wesley, 2001. (Component Software Series)
- [Bei90] Beizer, B.: "Software Testing Techniques." Van Nostrand-Reinhold, New York, 1990.
- [Bin00] Binder, R.V.: "Testing Object-Oriented Systems: Models, Patterns and Tools" Addison-Wesley, Boston, 2000.
- [CBT04] Schmitt, T.; Heid, R.; Groß, Dr. G.; Mayer, N.; Ochs, M.; Verlage, Dr. M.; Schmidt, G.W.; Maurer, U.: Ist-Analyse: Entwicklungs- & Testprozesse – Deliverable D1 im Projekt CBTesten", CBTesten Konsortium, Kaiserslautern: Januar 2004.
- [Gro02a] Groß, H.G.: Built-in Contract Testing in Model-driven, Component-based Development. 7th Intl. Conference on Software Reuse, WS CBSR, Austin, April 2002.
- [Gro02b] Groß, H.G.: Built-in Contract Testing in Component-based Application Engineering. Workshop on Component-based Development, Madrid, September 2002.
- [Spi03] Spillner, A.; Linz, T.: „Basiswissen Softwaretest. Aus- und Weiterbildung zum Certified Tester. Foundation Level nach ASQF- und ISEB-Standard.“ Heidelberg: dpunkt.Verlag, 2002.