Fraunhofer Institut
Experimentelles
Software Engineering

Research Lab Rheinland-Pfalz Testen und Testautomatisierung: Implementierung, Einführung und Evaluation der Lösungen

Ove Armbrust
Peter Dockweiler
Tobias Lindenblatt
Sebastian Scheffler
Clemens Schitter

Infrastruktur für eine Forschungs- und Transferplattform am Fraunhofer IESE in Kaiserslautern für regionale, software-entwickelnde KMUs (KMU Forschungslaborplattform)

Gefördert mit Mitteln des Europäischen Fonds für Regionalentwicklung (EFRE) und des Landes Rheinland-Pfalz (Ministerium für Wirtschaft, Verkehr, Landwirtschaft und Weinbau Rheinland-Pfalz: MWVLW RLP).

Förderkennzeichen: MWVLW;
Az.: 8315 38 51 04 IESE;
Kapitel 0877 Titel 892 02.

IESE-Report Nr. 047.06/D
Version 1.0
19. April 2006

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer-Gesellschaft.
Das Institut transferiert innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach (geschäftsführend)
Prof. Dr. Peter Liggesmeyer
Fraunhofer-Platz 1
67663 Kaiserslautern

Abstract

Dieses Dokument beschreibt Teile der Arbeiten, die bei den Projektpartnern im Rahmen des Research Lab Rheinland-Pfalz, Cluster „Testen und Testautomatisierung“ durchgeführt wurden. Beschrieben werden die Implementierung eines eigenen, spezialisierten Testwerkzeugs bei der market maker Software AG sowie die Anpassung und Einführung des ausgewählten Testwerkzeugs bei der WIKON Fernwirksysteme GmbH. Weiterhin wird die Integration in die bestehende Werkzeug- und Entwicklungslandschaft bei den Unternehmen und die notwendigen Schulungsmaßnahmen erläutert. Schließlich werden die Maßnahmen evaluiert.

Schlagworte: KMU Research Lab, built-in testing, computer software – testing, computer software - testing – automation, software test, test automation.

Inhaltsverzeichnis

1 Einführung

Dieses Dokument beschreibt die Arbeiten, die bei den Projektpartner im Rahmen des Research Lab Rheinland-Pfalz, Cluster „Testen und Testautomatisierung“ durchgeführt wurden. Im Projekt wurden zuerst der Stand der Forschung und Stand der Praxis festgestellt. Daraufhin wurde der tatsächliche Zustand der Testprozesse bei den Projektpartnern untersucht und Verbesserungsziele formuliert. Aus diesen Zielen wurden Anforderungen an Werkzeuge zur Unterstützung beim Testen, insbesondere auch für die Testautomatisierung, abgeleitet. Bei einer anschließenden Marktstudie wurden Kandidaten identifiziert, die (zumindest einige) Anforderungen erfüllen, bzw. es wurde festgestellt, dass der Markt keine entsprechende Werkzeugunterstützung anbietet.

Kapitel 2.2 beschreibt die Implementierung eines eigenen, spezialisierten Testwerkzeugs bei der market maker Software AG, Kapitel 2.3 die Anpassung und Einführung eines ausgewählten Testwerkzeugs bei der WIKON Kommunikationstechnik GmbH. In Kapitel 3 werden die Integration in die bestehende Werkzeug- und Entwicklungslandschaft bei den Unternehmen und die notwendigen Schulungsmaßnahmen erläutert. Schließlich evaluiert Kapitel 4 die Maßnahmen und zieht Schlüsse aus den Beobachtungen.

2 Anpassung und Implementierung der Werkzeuge

2.1 Einleitung

In diesem Kapitel wird beschrieben, welche Werkzeuge genutzt wurden, welche Anpassungen an den Werkzeugen vorgenommen wurden, und welche Werkzeuge selbst implementiert wurden.

2.2 market maker Software AG

Seit 1990 bietet die market maker Software AG privaten und institutionellen Anlegern hochwertige Lösungen für Portfoliomanagement und Börseninformationen.In diesem Projekt beteiligt war der Geschäftsbereich für Windows-Desktop-Software, der verschiedene Aspekte der Softwaretests wie zu spätes Finden von Fehlern, dynamische und Installationstests bearbeitete. Die Produktfamilie „Portfolio Manager“ des Geschäftsbereichs umfasst verschiedene Komponenten für Wertpapieranalyse und Portfoliomanagement für private Anwender über Vermögensverwalter bis hin zu Banken und Sparkassen. Derzeit pflegen und erweitern zwölf Entwickler die ca. 1,5 Millionen Zeilen Delphi- und C-Code, die in ca. 2.000 Modulen organisiert sind, die wiederum zu 23 Komponenten zusammengefasst sind.

2.2.1 Ausgangssituation

Beim Projektpartner market maker Software AG sind unter den beteiligten Entwicklern die Kenntnisse über Konzepte, Prinzipien und Methoden des Testens in den verschiedenen Phasen des Entwicklungsprozesses unterschiedlich und vor allem nicht durchgängig akzeptiert. Bei der Testspezifikation waren vor allem neue Anforderungen oft nicht genau genug bekannt oder nicht eindeutig genug, andere waren veraltet. Manche Anforderungen wurden daher im Verhältnis zu wenig, manche zu intensiv getestet.

Weiterhin ließen sich komplexere Units, Komponenten und Subsysteme (von einfachen Unit-Tests abgesehen) nur schwer, oft sogar gar nicht isoliert testen. Tests erforderten generell eine komplexe Testumgebung (Verzeichnisse, Benutzer, Systemkonfiguration, Netzwerk, Datenbank, Datenstände, ...), die nur mit hohem Aufwand bereitgestellt werden konnte. Eine automatische Testdurchführung fiel deshalb sehr schwer. Diese Tests wurden daher üblicherweise zeitaufwändig manuell oder sogar im schlimmsten Fall gar nicht ausgeführt. Letzteres betraf vor allem

Performance- und Lasttests auf Hardware, wie sie bei bestimmten Kunden im Einsatz ist. In der Folge traten Fehler, die eigentlich noch vor dem Release bei der market maker Software AG im Hause gefunden werden sollten, erst beim Kunden zutage.

Darüber hinaus gab es, bedingt durch den komplexen Aufbau der Software und der Testinfrastruktur, so gut wie keine automatischen Regressionstests. Nach der Korrektur eines Fehlers hätten eigentlich alle Tests noch einmal durchgeführt werden sollen, um sicherstellen zu können, dass sich durch die Korrektur keine neuen Fehler eingeschlichen haben. Dies geschah jedoch nur selten, da für die aufwendigen manuellen Tests keine Zeit blieb.

Generell prüfen die meisten Tests dynamische Aspekte der Software, sind also ausführungsbasiert und verlangen daher nach einer entsprechenden Ausführungsumgebung samt zeitaufwändiger Installation des Produktes und der Produktdatenbank. Der Aufwand, um diese Testumgebung bereitzustellen, überstieg oftmals den Aufwand für die eigentliche Ausführung der Tests. Auch statische Tests (zum Beispiel zur Überprüfung der Einhaltung von Designrichtlinien) wurden jedoch wenig genutzt.

Alle Tests müssen nicht nur einmalig angelegt werden, sondern müssen zusammen mit der Anwendung von Release zu Release gewartet und an neue/geänderte Anforderungen angepasst werden. Dieser Aufwand ist nicht unerheblich. Zudem sind ausführbare Tests auch wieder Software und unterliegen damit den gleichen Effekten wie die Nutzsoftware selber, also Anfälligkeit für Fehler erster und zweiter Art, Änderung von Anforderungen, Wartung, Dokumentation, ...

Hinzu kommt: Je umfangreicher Tests angelegt sind, desto größer ist der Wartungsaufwand im Falle einer Änderung der Anwendung, weil die Tests an die geänderte Anwendung angepasst werden müssen. Zum Aufwand der Änderung der Anwendung kommt also immer noch der Aufwand für die Anpassung der Tests hinzu. Die Möglichkeit, schnell auf geänderte Anforderungen zu reagieren, sinkt dadurch, was wiederum zu Unverständnis bei Kunden führt.

Der hohe Aufwand für die Erstellung bzw. Änderung der Tests führt oft dazu, dass nicht alle Testphasen so ausführlich wie eigentlich sinnvoll durchgeführt werden. Spätestens beim Systemtest werden dann aber wertvolle Ressourcen vergeudet, indem Fehler gefunden werden, die von ihrer Art her eigentlich bei den Unit- und Integrationstests hätten gefunden werden können. Fehler auf Unit- und Integrationsebene werden dadurch zwar gefunden, aber zu spät. Als abgeschlossen angesehene Phasen müssen zur Fehlerkorrektur erneut aufwändig durchlaufen werden, bis hin zum (wiederholten) Systemtest. Dabei sind im schlimmsten Fall nicht nur lokale Änderungen notwendig, sondern globale Änderungen am Programmcode oder gar am Design. Durch diese Sprünge in weit

zurückliegende Entwicklungszyklen und das späte Feedback fällt es schwer, noch während einer Phase im gleichen Projekt aus Fehlern zu lernen. Insgesamt fällt dadurch die Reaktion auf geänderte Anforderungen zu langsam aus.

Die Testdurchführung ist zu sehr von den persönlichen Erfahrungen, dem Verständnis und der Interpretation des jeweiligen Testers abhängig. Tests sind dadurch schwer wiederholbar und die Ergebnisse verschiedener Testdurchläufe nur schwer miteinander vergleichbar.

Der Aufwand für Installationstests ist durch die sehr große Zahl möglicher Kombinationen der unterstützten Betriebssysteme, Betriebssystemeditionen und Betriebssystemversionen, von optionalen Betriebssystemkomponenten, sonstiger installierter Software und Versionen bzw. Varianten der zu testenden Anwendung sehr groß, selbst bei gezielter Beschränkung auf eine praktisch relevante Auswahl von Betriebssystemen und ein sinnvolles Minimum von Kombinationen. Erschwerend kommt hinzu, dass bisher alle Installationstests manuell ausgeführt werden mussten, was im Laufe eines Projektes wegen der Versäumnisse beim Testen auf der Integrationsebene und der dadurch erforderlichen Rework-Zyklen auch noch mehrfach im Laufe eines Projektes notwendig wurde.

2.2.2 Bearbeitete Probleme

Im Rahmen eines Messprogramms wurden zuerst die Probleme identifiziert und priorisiert. Die zwei Hauptprobleme waren die geringe Effizienz der Systemtests und die mangelnde Effektivität der Installationstests.

(P1) Die Effizienz der Systemtests ist zu gering

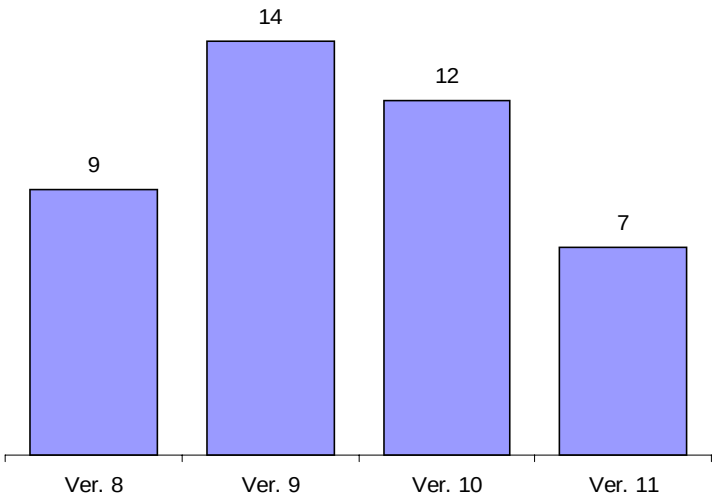

Abbildung 1: Anzahl der Durchläufe des Systemtests

In den Entwicklungszeiträumen der letzten vier Versionen der Produktfamilie wurden zwischen sieben und vierzehn, im Mittel über zehn Release-Kandidaten erstellt (siehe Abbildung 1), die alle jeweils einem Systemtest unterzogen wurden, der aufwändig manuell vorbereitet und durchgeführt werden musste.

Im Rahmen des Projekts wurde ein Messprogramm aufgesetzt, um die Verteilung der Fehler auf die Fehlerklassen

- Unit-Fehler
- Integrationsfehler
- Installationsfehler
- Systemfehler

zu bestimmen. Abbildung 2 stellt diese Verteilung dar.

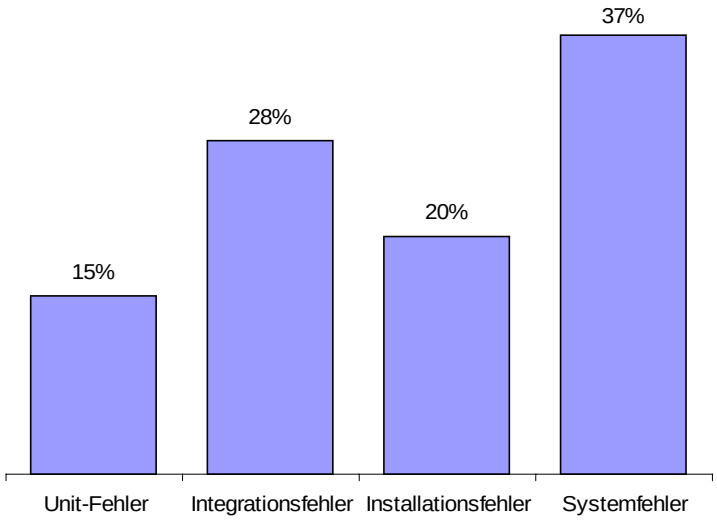

Abbildung 2: Arten der im Systemtest gefundenen Fehler

Nur etwa ein Drittel der in allen Systemtestdurchläufen gefundenen Fehler sind eigentliche Systemfehler. Die restlichen zwei Drittel hätten eigentlich in einem früheren Test gefunden werden müssen, sie passieren diese Tests jedoch unentdeckt. Sie werden erst viel später im Systemtest gefunden und erzwingen damit eine Rework-Phase, die einen erneuten Durchlauf des Systemtests zur Folge hat. Dies treibt die Gesamtanzahl der manuellen Systemtestdurchläufe unnötig in die Höhe und vergeudet im Systemtest wertvolle Ressourcen (siehe Abbildung 3).

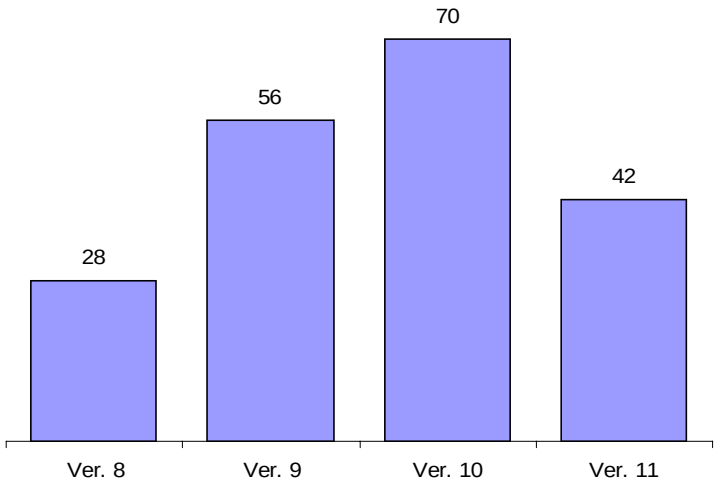

Abbildung 3: Gesamtaufwand in Personenstunden für alle Durchläufe der Systemtests einer Version

Im Rahmen des eingeführten Messprogrammes wurden bei der Entwicklung der vier beobachteten Produktversionen Gesamtaufwände für den Systemtest zwischen 28 und 70 Personenstunden festgestellt. Ein vollständiger Systemtest, also die Systemtests aller Release-Kandidaten einer Produktversion zusammengenommen, hat folglich zwischen etwa einer und zwei Personenwochen gedauert. Zwei Drittel der dabei gefundenen Fehler hätte sich in einer früheren Testphase finden lassen können, wodurch sich manch einer der Release-Kandidaten samt aufwändigem Systemtest hätte vermeiden lassen können.

(P2) Die Effektivität der Integrationstests ist zu gering

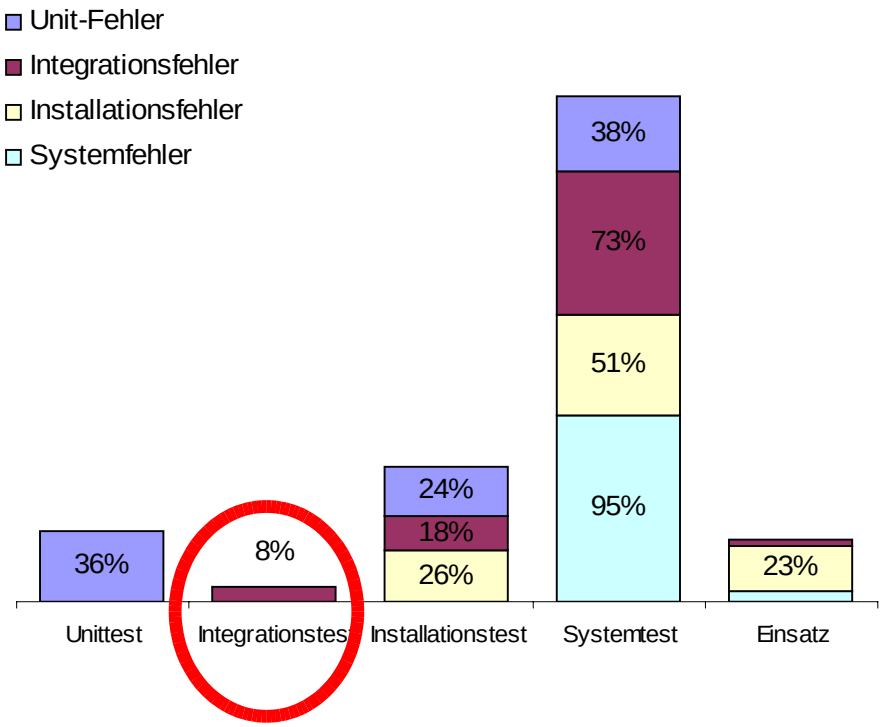

Abbildung 4: Integrationstest deckt nur 8 Prozent der Integrationsfehler auf

Die Ergebnisse des eingeführten Messprogrammes zeigen, dass lediglich 8 Prozent der Integrationsfehler im Rahmen des Integrationstests gefunden wurden, 18 Prozent erst auf der nächsten Teststufe des Installationstests und nahezu drei Viertel erst weitaus später beim Systemtest. Den Messergebnissen zufolge werden somit die weitaus meisten Integrationsfehler noch vor der Freigabe des Produktes für den Einsatz gefunden, allerdings zu spät im Entwicklungsprozess. Der Integrationstest beschränkte sich bei den beobachteten vier Produktversionen auf eine statische Überprüfung durch den Compiler im Rahmen eines

kontinuierlichen Build-Prozesses. Die Effektivität dieses „Tests“ ist mit 8 Prozent deutlich zu gering.

(P3) Die Effektivität der Installationstests ist zu gering.

Die Installation der diversen Varianten der Produktfamilie müssten eigentlich auf allen unterstützten Betriebssystemversionen getestet werden. Daraus ergibt sich eine große Anzahl von Testfällen, die eigentlich alle geprüft werden müssten (siehe Abbildung 5).

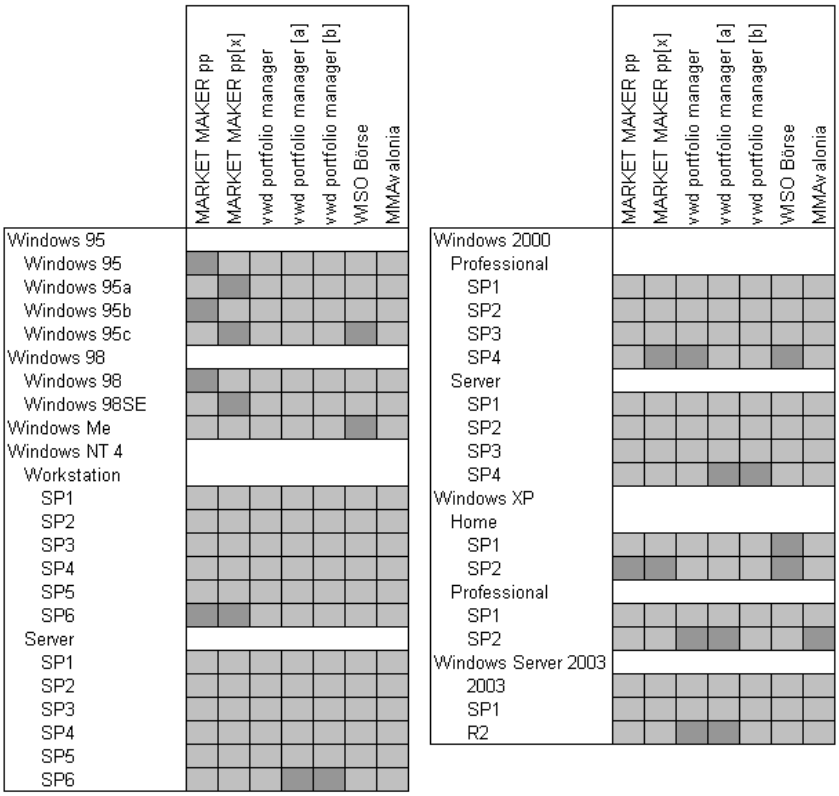

Abbildung 5: nötige Testfälle (hellgrau), typische Auswahl von Testfällen (dunkelgrau)

Da die Installationstests derzeit nur manuell durchgeführt werden können, kann unter Berücksichtigung der verfügbaren Testressourcen lediglich eine kleine, möglichst typische Auswahl von Betriebssystemen berücksichtigt werden. Die Testabdeckung fällt daher zwangläufig gering aus (siehe Abbildung 5).

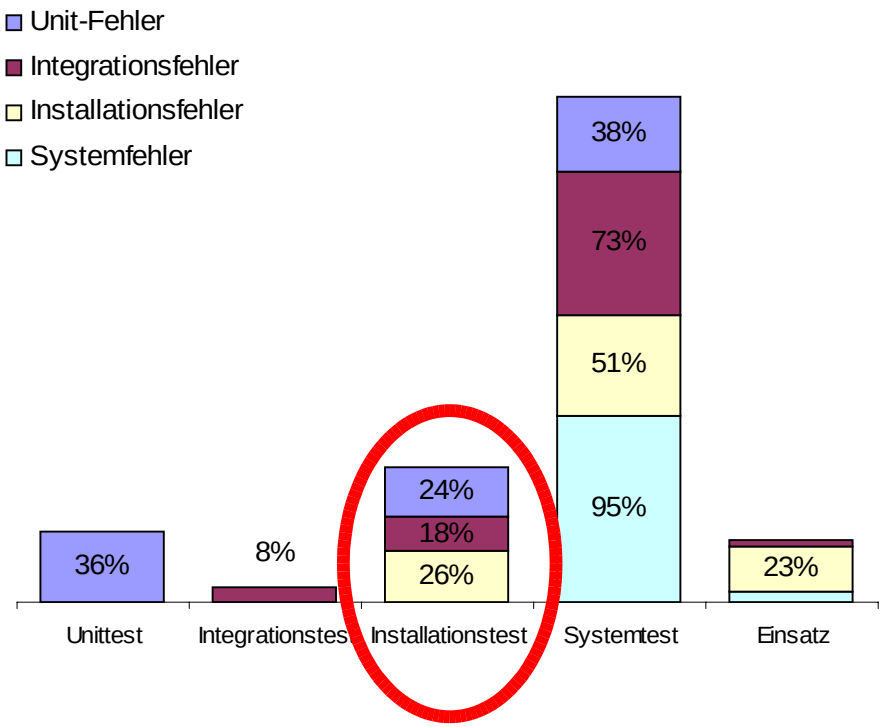

Abbildung 6: Installationstest deckt 26 Prozent der Installationsfehler auf

Logische Folge der geringen Testabdeckung ist eine geringe Effektivität beim Finden von Fehlern: Im Mittel werden gerade einmal 26 Prozent der insgesamt gefundenen Installationsfehler auch tatsächlich bei den Installationstests gefunden (siehe Abbildung 6).

2.2.3 Ziele

(Z1) Effizientere Systemtests durch effektivere Integrations- und Installationstests.

Fehler sollten im Laufe des Entwicklungsprozesses möglichst früh gefunden werden, um unnötige Durchläufe der aufwendigen manuellen Systemtests zu vermeiden. Typische Fehler der Integrations- und Installationsebene sollten daher nicht erst beim Systemtest gefunden werden. Vielmehr sollte dieser Systemtest hauptsächlich dazu dienen, Fehler auf der Systemebene aufzudecken. Abbildung 7 zeigt die momentane und die angestrebte Verteilung.

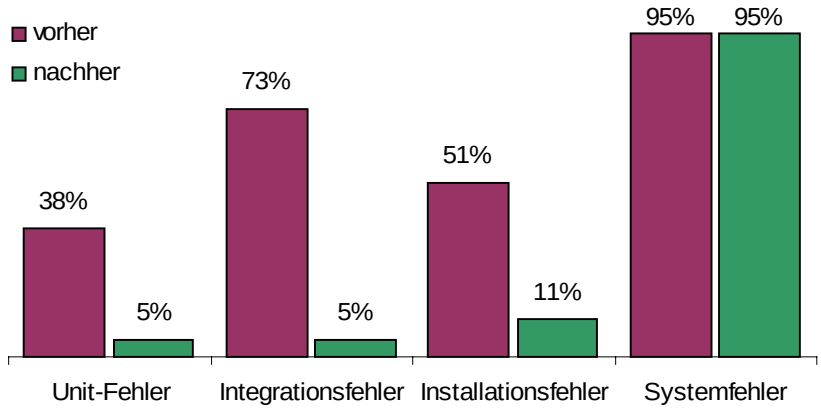

Abbildung 7: Aktuelle und angestrebte Verteilung der gefundenen Fehler

Im gesamten Systemtest sollten nicht mehr als 5 Prozent aller Unit-Fehler, nicht mehr als 5 Prozent aller Integrationsfehler und nicht mehr als 11 Prozent aller Installationsfehler gefunden werden.

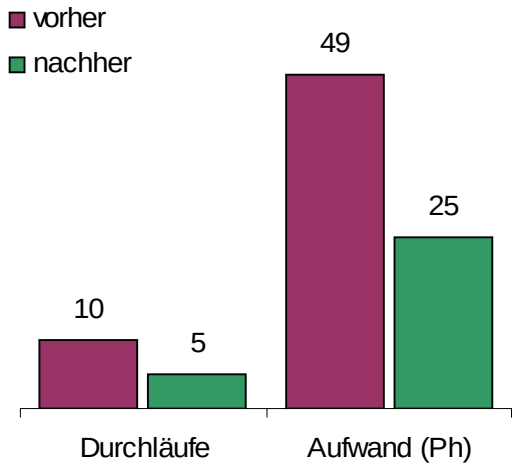

Abbildung 8: Aktuelle und angestrebte Anzahl der Systemtest-Durchläufe und deren Aufwand (in Personenstunden)

Abbildung 8 zeigt die angestrebte Aufwandsentwicklung. Die durchschnittliche Anzahl von Release-Kandidaten, für die bei der Entwicklung einer Produktversion der Systemtest jeweils einmal durchgeführt werden muss, sollte von 10 auf 5 halbiert werden, ebenso der durchschnittliche Gesamtaufwand für alle Systemtestdurchläufe von durchschnittlich 6 Personentagen (49 Personenstunden) auf 3 Personentage (25 Personenstunden).

(Z2) Effektivere Integrationstests durch höhere Testabdeckung.

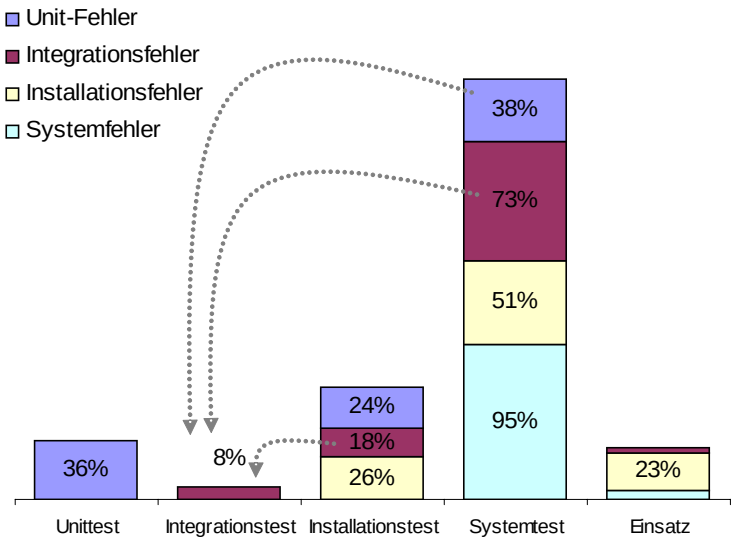

Abbildung 9: Aktuelle Verteilung der gefundenen Fehler

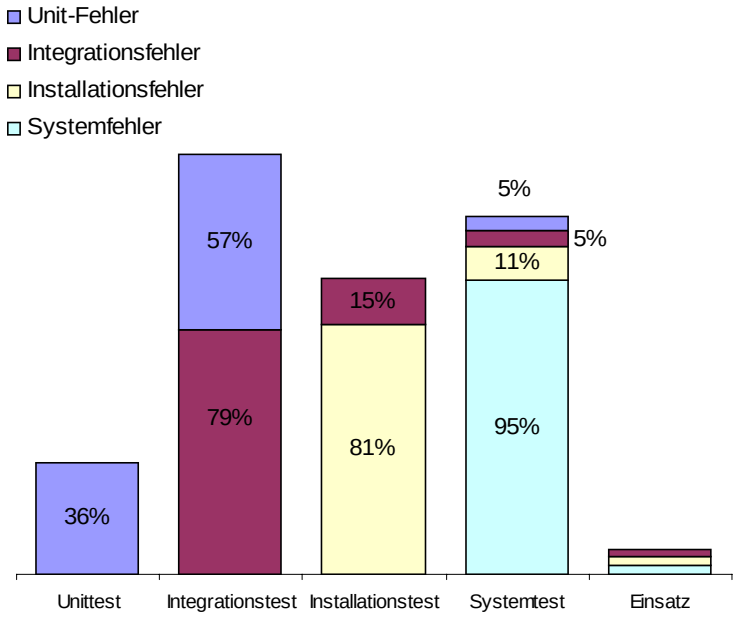

Abbildung 10: Angestrebte Verteilung der gefundenen Fehler

Die Effektivität der Integrationstests soll erhöht werden (siehe Abbildung 9 bzw. Abbildung 10). Dazu müssen Units und Subsysteme auf der Integrationsebene isoliert testbar sein. Die folgenden Ursachen geringer Testbarkeit sind anzugehen:

- Units und Subsysteme sind zu stark aneinander gekoppelt und dadurch teils schwer und teils überhaupt nicht isoliert testbar. Integrationstests derart gekoppelter Anwendungsteile benötigen übermäßig viele Teile des kompletten Systems und können daher in einen vollständigen Systemtest ausarten.
- Testen wird als lästige, der Entwicklung nachgelagerte eigenständige Phase verstanden. Dadurch werden Aspekte der Testbarkeit allgemein und speziell der Kopplung zu wenig bereits in den frühen Entwicklungsphasen der Anforderungsdefinition (testbare Anforderungen) und des Designs (testbare Architektur) berücksichtigt.
- Testbarkeit soll als integraler Bestandteil des gesamten Entwicklungsprozesses verstanden werden. Test- und Entwicklungstätigkeiten sollen iterativ, parallel und einander ergänzend ausgeführt werden.

(Z3) Effiziente Integrationstests durch Automatisierung

Die Verlagerung von Tests vom Systemtest weg in die Integrationsphase hilft zwar zunächst dabei, Aufwand durch weit zurückgreifende Rework-Zyklen zu vermeiden. Um aber wirklich Testressourcen zu schonen, sollte die Effizienz des Integrationstests durch Automatisierung gesteigert werden. Die Systemtests lassen sich technisch so gut wie nicht automatisch vorbereiten und durchführen. Die weniger komplexe Integrationsebene bietet aber einige Möglichkeiten zum automatisierten statischen und dynamischen Testen, die genutzt werden sollten.

(Z4) Effektivere Installationstests durch höhere Effizienz.

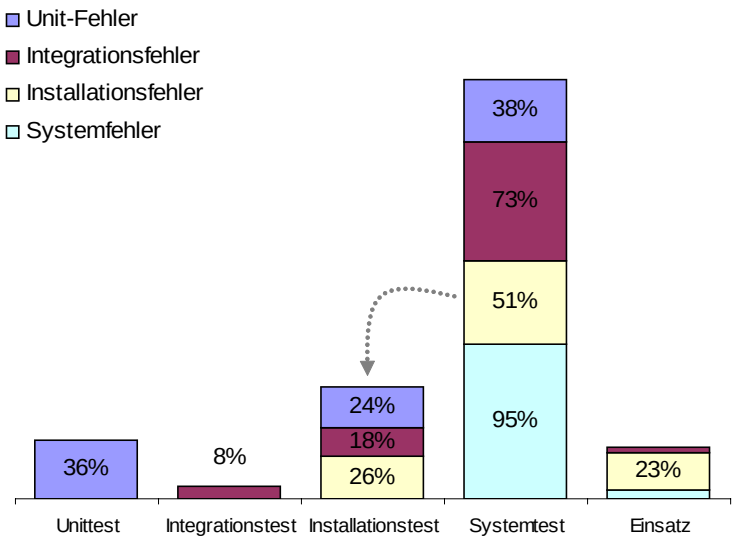

Abbildung 11: Aktuelle Verteilung der gefundenen Fehler

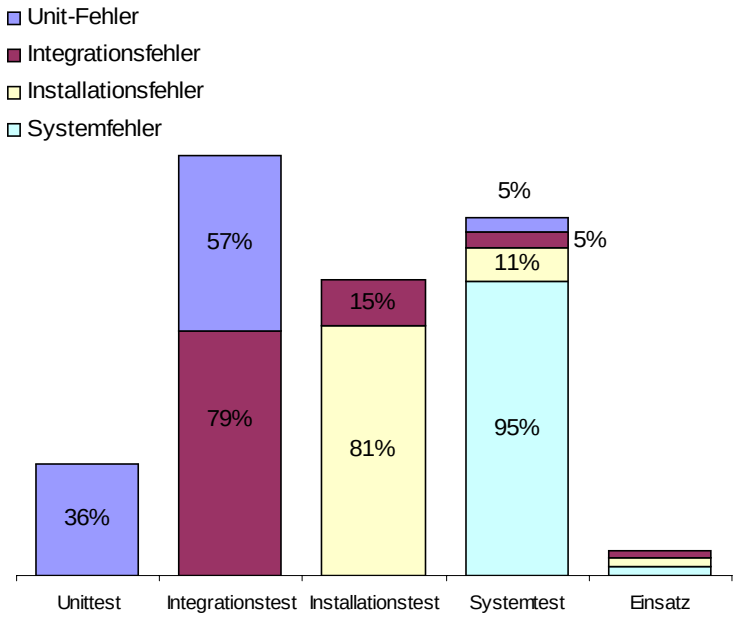

Abbildung 12: Angestrebte Verteilung der gefundenen Fehler

Der Aufwand zur Ausführung der Installationstests soll verringert werden. Dadurch werden Ressourcen frei, die es erlauben, mehr Testfälle zu

berücksichtigen. Dadurch ist eine höhere Testabdeckung möglich, es können also mit den gleichen Ressourcen früher mehr Fehler entdeckt werden (siehe Abbildung 11 bzw. Abbildung 12).

(Z5) Effizientere Installationstests durch Automatisierung.

Durch Automatisierung der Installationstests soll der Durchführungsaufwand verringert werden. Dabei sollten sich automatische Installationstests sowohl bei der Entwicklung, aber vor allem auch als Nachweis einer ordnungsgemäßen Installation als Teil des Abnahmetests vor Ort beim Kunden sowie zum Zweck der Problemanalyse während des Einsatzes verwenden lassen. Zu prüfen sind dabei

- statische Aspekte wie Verzeichnisse, Dateien, Dateigrößen, Dateiversionen, Dateiprüfsummen
- dynamische Aspekte im Sinne ausführungsbasierter Built-in-Tests

Ziele für neue oder geänderte Software

Die angeführten Ziele Z1 bis Z5 haben offensichtlich direkte Auswirkungen auf die Gestaltung der Produkte des Projektpartners market maker Software AG. Darüber hinaus sind auch spezielle Werkzeuge, z.B. für die Automatisierung von Vorgängen, unerlässlich. Unter Zuhilfenahme bereits im Einsatz befindlicher und neu zu implementierender Werkzeuge gelten als Ziele für die Produkte der market maker Software AG:

- testbar auf Unit- und Integrationsebene
- statisch und dynamisch testbar auf Systemebene

Motivation für die Implementierung eigener Werkzeuge

Aufgrund der sehr speziellen Anforderungen der market maker Software AG bezüglich des Installationstests gibt es kein kommerziell erhältliches Werkzeug, welches die Anforderungen in ausreichendem Maße erfüllt [AAB+05]. Gleichzeitig fordern jedoch Kunden verstärkt den formalen Nachweis der korrekten, lauffähigen Installation der market maker-Software auf ihren Systemen.

Ein solcher Installationstest soll dem Kunden über den reinen Funktionsnachweis hinaus auch zur selbst durchzuführenden Analyse einer bestehenden Installation dienen. Der Test muss daher möglichst einfach und ohne Installation anderer Werkzeuge lauffähig und benutzbar sein.

2.2.4 Implementierung der Lösung

Um die angeführten Ziele zu erreichen, wurden verschiedene Implementierungsmaßnahmen gestartet. Dieser Abschnitt gibt einen Überblick über die den Zielen zugeordneten Maßnahmen, erforderliche Aufwände und erste Ergebnisse.

Neuimplementierungen/Wissensaufbau

- (Z1): Umsetzung durch Verfolgung von Ziel (Z2)
- (Z2): Test-Know-How ausbauen:
  - Testen allgemein
  - Testen speziell objektorientierter Systeme
  - Zusammenhänge zwischen Testbarkeit und Design (Kopplung)
  - Konzept der testgetriebenen Softwareentwicklung
  - testgetriebene Entwicklung als Designmethode
  - firmeninterner Wissenstransfer bezüglich Testen, Testbarkeit und testgetriebener Entwicklung
- (Z3): Kontinuierlichen Build-Prozess in einen kontinuierlichen Integrationsprozess überführen, also den bestehenden Build-Prozess erweitern um automatische Unit-Tests und automatische Built-In-Tests.
- (Z4): Umsetzung durch Verfolgung von Ziel (Z5)
- (Z5): Tool zum automatischen statischen und dynamischen Testen und Prüfen einer Installation entwickeln. Für die dynamischen Tests greift dieses Tool auf die für (Z3) notwendigen Built-in-Tests zu.

Erweiterung vorhandener Produkte und Infrastruktur

- Für die dynamischen Built-in-Tests müssen alle vorhandenen relevanten ausführbaren Module wie beschrieben erweitert werden.
- Für den kontinuierlichen Integrationsprozess muss der vorhandene kontinuierliche Build-Prozess wie beschrieben erweitert werden.

Bis Projektende bereits erfolgte Maßnahmen

- Es wurde ein Messprogramm aufgestellt und kontinuierlich durchgeführt. Die bisherige Dauer beläuft sich auf 12 Monate, beteiligt sind 9 Entwickler. Beobachtet wird die Entwicklung von vier Produktversionen der Produktfamilie von Beginn des Entwicklungszyklus bis hin zu Auslieferung und Einsatz.
- Anforderungen an Know-How-Transfer zum Thema Testen, Testbarkeit und testgetriebene Entwicklung wurden aufgestellt.
- Seit November 2004 gibt es wöchentliche Entwicklertreffen, bei

denen u.a. Anforderungen, Ideen und Probleme aus den Bereichen Testen, Testbarkeit und testgetriebene Entwicklung ausgetauscht werden.

- Weiterhin wurden Möglichkeiten zur Erweiterung des Build-Prozesses hin zu einem kontinuierlichen Integrationsprozess auf Basis des eingesetzten Tools untersucht. Dies umfasst beispielsweise die folgenden Fragen:
    - Wie kann der Testkontext auf dem Buildsystem hergestellt werden?
    - Wie werden GUI-Meldungen im Batch-Betrieb des Build-Prozesses verarbeitet?
    - Was sollen die Auswirkungen eines positiven bzw. eines negativen Testergebnisses sein?
    - Sollen Freigabemechanismen an den Testausgang gekoppelt werden?
    - Sollen die erstellten Module abhängig vom Testausgang abgelegt werden?
    - Wie können die Tests nachvollziehbar protokolliert werden?
- Es wurden Anforderungen an ein selbst entwickeltes Tool für statische und dynamische Installationstests im Rahmen der Entwicklung hinsichtlich Einsatz, Umgebung, Funktionen, Daten und Leistungen aufgestellt. Darüber hinaus wurde ein Stufenplan erstellt und ein Grobkonzept entworfen.
- Es wurde ein Framework für die Built-In-Tests (dynamische Installationstests) entwickelt.
- Es wurde der Prototyp eines Frameworks zur Automatisierung von Installationstests entwickelt.
- Da während der Projektlaufzeit von Kundenseite überraschend eine starke Nachfrage nach dem Installationstest-Tool zu verzeichnen war, wurden die internen Anforderungen erweitert. Sie wurden ergänzt um Kundenanforderungen zur Nutzung der Testergebnisse im Rahmen der Systemabnahme und zum Einsatz des Tools durch den Kunden selbst während des produktiven Betriebs des Produkts. Die neuen Anforderungen gehen über den zunächst geplanten Umfang hinaus.

Eingesetzte Fremdsoftware

Im Rahmen der Integrations- und Installationstests wurde Software von folgenden Fremdherstellern eingesetzt:

- Integrationstest: CruiseControl der ThoughtWorks, Inc. [CC]
- Automatisierung des Installationstests: VMware Workstation und VMWare Player der VMWare Inc. [VMW]

2.2.5 Aufwandsabschätzungen

Da bei Projektende noch nicht alle Implementierungsmaßnahmen abgeschlossen waren, können sich die angegebenen Aufwände sowie der Umfang der Implementierungsarbeiten noch verändern.

Umfang der Programmierarbeiten

- Framework für Built-In-Testing der ausführbaren Module: 100 LOC
- Erweiterung aller ausführbaren Module um Built-In-Testing:
  - ca. 80 EXE-Module
  - ca. 60 DLL-Module
  - Aufwand jeweils mind. 10 LOC auf Ausbaustufe 1
  - Aufwand nach oben offen auf den höheren Ausbaustufen (stufenweiser Ausbau erfolgt nach Projektende)
- Tool für Installationstests: geplanter Umfang ca. 20.000 LOC

Aufwände

- Framework für Built-In-Testing: 2 Personentage
- Framework zur Automatisierung von Installationstests: 3 Personentage
- Prototypische Erweiterung von fünf Modulen um Built-In-Testing: 5 Personentage
- Tool für Installationstests: 30 Personentage (geschätzt). Das Tool wurde bisher noch nicht implementiert, weil die ursprünglichen Anforderungen nur die Nutzung im Rahmen der Entwicklung vorsahen, dies jedoch aufgrund von Kundenanforderungen erweitert werden musste. Das Tool wird somit Bestandteil der nächsten Produktversion sein und ist damit an den Produktlebenszyklus gebunden. Die Konzeptionsphase für die nächste Produktversion lief bis Ende 2005, die Realisierung des Produktes und damit des Test-Tools wird erst nach dem Projektende beginnen.

2.3 WIKON Kommunikationstechnik GmbH

Die **WIKON Kommunikationstechnik GmbH** ist ein Unternehmen, das sich auf die Entwicklung von Bus- und Fernwirksystemen zur Überwachung technischer Anlagen spezialisiert hat. Das bedeutendste Softwareprodukt der Firma ist „WatchMyHome“, ein Internet-Fernwirk-Server. Er überwacht und visualisiert Fernwirksysteme weltweit und bietet einen Überblick über alle überwachten Geräte.

Die **verit Informationssysteme GmbH** ist ein IT-Dienstleister mit Schwerpunkt auf Softwareentwicklung und Qualitätssicherung. Als Produkt

entwickelt und vertreibt sie die Testplattform PETA (Plattform für Effektive Testautomatisierung) zum automatisierten funktionalen Testen von kommunizierenden Systemen auf Integrationsebene.

Zusammen haben die beiden Unternehmen die Testfallerstellung und Testdurchführung bei WIKON durch die Nutzung eines Testframeworks optimiert.

2.3.1 Ausgangssituation

WIKON

Viele Fehler an Webseiten des WatchMyHome-Servers sind erst im Live-System aufgefallen und wurden oft durch die Kunden bemerkt. Dies ist sowohl für den Kunden als auch für WIKON unbefriedigend. Die Situation entstand hauptsächlich durch mehrere Ansätze von Tests, die parallel verwendet wurden:

- Testen der Funktionalität der Weboberfläche (Regressionstests)
- Testen der Datenbankinhalte (Regressionstests)
- Testen der ein- und ausgehenden SMS-Inhalte
- Testen der Mail- und Fax-Connectivity
- Testen des Webservice-Interfaces

Ziel war es daher, eine einheitliche Testumgebung zu schaffen, die alle oben erwähnten Bereiche abdecken kann.

verit Informationssysteme GmbH

Die PETA-Plattform wurde bereits vor Beginn der Kooperation entwickelt und im Projektumfeld einer Mobilfunk-Provider-übergreifenden, mobilen Bezahlplattform mehrfach erfolgreich eingesetzt.

PETA bietet einen generischen Ansatz zum Testen von Softwaresystemen auf Basis der unter den Systemkomponenten ausgetauschten Nachrichten. Dieser Ansatz lässt sich nahtlos auf verwandte Anwendungsfälle (wie bspw. der Verifizierung von Datenbankinhalten oder Dateiinhalten) erweitern.

Eine Vorauswahl potentieller Testwerkzeuge anhand der Anforderungen von WIKON ergab, dass PETA ein geeigneter Kandidat ist. Es war somit naheliegend, zu erproben ob es mit Hilfe der PETA-Plattform möglich ist, die

Testsituation bei WIKON entscheidend zu verbessern.

Während der Projektlaufzeit wurde die eng an den Erfordernissen des initialen Projektumfeldes orientierte Funktionalität der Kernplattform (PETA-Core) durch eine grafische Entwicklungsumgebung (PETA-Designer), umfangreiche Dokumentation, Installationsprozeduren sowie ein Plattform-Modul zum generischen Zugriff auf Datenbankinhalte erweitert.

2.3.2 Erste Schritte

Voraussetzungen zur Installation und Benutzung von PETA-Core:

- Betriebssysteme: Windows, Unix
- Java Runtime Environment
- PETA-Core
- Apache Ant (optional)

Bei WIKON wurde folgende Konfiguration für das Testsystem verwendet:

- Betriebssystem: SuSE Linux, Version 10.0
- Java Runtime Environment, Version 1.4.2_06
- PETA-Core, Version 1.4.0
- Apache Ant, Version 1.6.2

2.3.3 Implementierung von PETA bei WIKON

Die Testspezifikationen für PETA-Core liegen im XML-Format vor. Sollen die Spezifikationen von Hand erstellt werden, so sind Grundkenntnisse in XML und XPath hilfreich. Sehr viel komfortabler sind sämtliche Spezifikationen jedoch mit dem grafischen Editor PETA-Designer zu erstellen, welcher die XML-Schicht bei Bedarf komplett vor dem Benutzer verbirgt. Diese Komponente wurde während der Einführungsphase zusätzlich zur Verfügung gestellt, um einen leichteren Zugang zum Thema Testfallspezifikation zu ermöglichen.

Alle bei der Installation aufgetretenen Fragen wurden durch den Support der verit Informationssysteme GmbH schnell behoben. Während der zwei bis drei Tage dauernden Einarbeitungsphase wurden vorhandene Dateien aus den Samples abgeändert.

Mit zunehmender Vertrautheit des zugrunde liegenden Konzepts rückte die Erstellung komplett eigener Testsuites in den Vordergrund.

Schon während der Testfallerstellung hat sich gezeigt, dass die Testautomatisierung mit PETA das Potential hat, viele Fehler aufzuspüren, die bisher erst im Live-System entdeckt wurden. Kein Werkzeug - so auch PETA nicht - kann eine systematische und gründliche Testplanung ersetzen. Allerdings unterstützt PETA, insbesondere in Verbindung mit dem grafischen Editor, sehr gut ein systematisches Vorgehen zur Erstellung hochwertiger Testsuiten.

2.3.4 Bewertung

Testfallerstellung als XML-Spezifikation

Vor dem Einsatz von PETA wurden geänderte Webseiten des WatchMyHome-Servers nur auf einem Testserver vom jeweiligen Entwickler im Browser getestet. Testprotokolle und –dokumentation waren lückenhaft und beschrieben daher nur teilweise, wie und in welchem Umfang getestet wurde. Als Konsequenz daraus wurden viele Fehler erst im Live-Betrieb (teilweise von Kunden) gefunden. Der Einsatz von PETA veränderte diese Situation grundlegend.

Im Einzelnen wurden mehrere Testsuites definiert, die jeweils einzelne Testabläufe zusammenfassen. Eine der Suites dient beispielsweise zur Überprüfung des Login-Vorgangs und zur Feststellung ob die erwarteten Zugriffsrechte und verfügbaren Funktionen auf der Weboberfläche den tatsächlich angebotenen entsprechen. Eine weitere Suite enthält Testfälle die prüfen, ob aufgerufene Seiten innerhalb eines Profils auch ordnungsgemäß geladen werden. Dadurch kann die Navigation des Benutzers auf der Seite beliebig simuliert werden.

Neben den Routineüberprüfungen wurden auch ganz konkret aufgetretene Probleme gezielt mittels einer Testsuite gesucht. Beispielsweise fehlte in der Vergangenheit auf einigen Modulseiten ein Auswahlfeld mit dem Namen 'action'. Die entsprechende Testsuite ruft Modulseiten auf und überprüft das Vorhandensein des Steuerelements. Auch wenn während der Projektlaufzeit dieser Fall nicht wieder eingetreten ist, so ist dieser Test jetzt Bestandteil der Standardtests, die nach einer Änderung an den Webseiten durchgeführt werden. Damit ist das Auftreten dieses (bisher im Live-Betrieb immer wieder auftretenden) Fehlers von jetzt an so gut wie ausgeschlossen.

Nach einer Einarbeitungsphase und der zunehmenden Vertrautheit mit den Elementen der Testbeschreibungssprache geht die Erstellung der Testfälle direkt in XML immer zügiger. Mithin konnte der Zeitaufwand auf etwa ein

Viertel reduziert werden. Hierbei ist nicht zuletzt das Handbuch mit seiner ausführlichen Sprachdokumentation eine große Hilfe. (Der Einsatz von PETA-Designer beschleunigt die Arbeit durch die grafische Abstraktion von der XML-Schicht zusätzlich, siehe nächster Abschnitt.) Macht man vom Modulkonzept Gebrauch und verwendet gleiche Teile wieder, so werden die Testfälle zusätzlich kürzer und sind selbstverständlich auch schneller erstellt.

Durch den Einsatz der PETA-Plattform sind zusätzlich mehrere Fehler in der Zeichencodierung der erzeugten Webseiten aufgefallen, die bisher im Live-Betrieb aufgrund der Fehlertoleranz moderner Webbrowser nicht zutage getreten sind. Erfreulicherweise konnten jedoch noch keine schwerwiegenden Fehler aufgespürt werden.

Testfallerstellung mittels PETA-Designer

Im Laufe des Projekts wurde mit dem PETA-Designer von der verit Informationssysteme GmbH eine weitere Komponente von PETA freigegeben, die die unkomfortable und fehlerträchtige manuelle Erstellung der XML-Testspezifikationen durch einen grafischen Editor ersetzt. Der PETA-Designer erleichtert die Erstellung und Erweiterung von Testfällen und sorgt für eine deutlich übersichtlichere Darstellung als beispielsweise die Anzeige der XML-Dateien im Texteditor. Die Bedienung ist einfach und intuitiv gestaltet.

Sowohl erfahrenen Testfallentwicklern als auch unerfahrenen Neulingen erleichtert der grafische PETA-Designer die Testfallerstellung erheblich. Insbesondere hilft der Designer den Überblick zu behalten wenn die Sammlung an Testfällen allmählich wächst. Ebenso werden Querbezüge zwischen Modulen visuell dargestellt, was die Wiederverwendung von (Teilen von) Testsuiten unterstützt.

Der tägliche Umgang mit PETA

Die Testautomatisierung bringt WIKON den Vorteil, dass bereits erstellte Tests beliebig oft und praktisch ohne zusätzlichen Aufwand wiederholt werden können. Auf diese Weise lässt sich nach einem Update mit den vorhandenen Testsuites leicht feststellen, ob die Basisfunktionalität noch gegeben ist oder im Rahmen des Updates beschädigt wurde.

Bei zukünftigen Entwicklungen am WatchMyHome-Server, besonders bei der Redesignphase der Weboberfläche in Kürze, wird der Gedanke der Testbarkeit, insbesondere auch der Testautomatisierung, von Anfang an eine wichtige Rolle spielen. Das konkrete Ziel hierbei ist es, Funktionalität und Testprozeduren von Anfang an parallel zu entwickeln, um so eine

möglichst umfassende Testfallabdeckung zu gewährleisten.

2.3.5 Fazit

WIKON

Wir danken der verit Informationssysteme GmbH für den Einblick und die Erfahrungen, die wir durch das Projekt im Bereich Testen und Testautomatisierung sammeln konnten. Besonders möchten wir uns für die schnelle und kompetente Hilfe bei Fragen und Problemen bedanken.

PETA macht auf uns einen leistungsfähigen und stabilen Eindruck. Insbesondere die Erstellung der Testfälle wurde durch den grafischen Designer stark vereinfacht. Das Thema Testautomatisierung wird für uns auch weiterhin wichtig bleiben und wir hoffen, die Qualität und Stabilität unserer Produkte dadurch in Zukunft noch weiter verbessern zu können.

verit

Wir möchten WIKON für das Feedback danken, insbesondere zum neuen grafischen Editor PETA-Designer. Der Installationsprozess konnte durch die Entwicklung von Installern für Windows und Linux stark vereinfacht und anwenderfreundlicher gestaltet werden. Zusätzliche Beispiele sorgen für einen noch leichteren Einstieg in die Testfallmodellierung mit PETA.

Die besonderen Anforderungen der WIKON führten zur Entwicklung von Plattformerweiterungen wie dem oben genannten Datenbankzugriffsmodul, welches nun unser Produktportfolio sinnvoll ergänzt.

Zusätzlich zum bereits vorhandenen Core-Referenz-Handbuch haben wir aufgrund des Feedbacks auch ein Tutorial im PDF-Format mit Schritt-für-Schritt-Anleitungen durch Screenshots erstellt, sowie animierte Flashfilme zum besseren Verständnis der grundlegenden Funktionsweise der Plattform.

3 Arbeitspaket 7: Integration und Schulung

3.1 Einleitung

In diesem Kapitel wird beschrieben, wie die angepassten Tools und die Neuentwicklungen in die bestehende Werkzeuglandschaft der Projektpartner integriert wurden. Weiterhin werden die nötigen Schulungsmaßnahmen erläutert.

3.2 market maker Software AG

Die Integration der entwickelten Werkzeuge kann erst nach Fertigstellung derselben erfolgen. Da die Werkzeuge im Laufe des Projekts integraler Bestandteil des Produkts wurden, sind sie nun an den Produktlebenszyklus gebunden. Daher ist die Integration noch nicht abgeschlossen, daher wird in diesem Abschnitt hauptsächlich die geplante Integration geschildert.

Integration

Grundsätzlich wird das Tool für die Installationstests in den Kontext der nächsten Produktversion integriert. Diese Integration wird sich jedoch auf ein Minimum beschränken, da das Tool vor allem bei einer teilweise oder ganz zerstörten Installation der anderen Produktbestandteile lauffähig sein muss und gerade in solchen Fällen aussagekräftige Testergebnisse liefern muss. Zur Ermittlung der Referenzwerte wird das Tool im Rahmen der Entwicklung an den bestehenden Build-Prozess angebunden werden.

Schulung / Wissenstransfer

Sobald das Werkzeug einsatzfähig ist, werden Mitarbeiter aus der Entwicklung in der Verwendung des Built-In-Testing-Frameworks geschult. Hierbei werden insbesondere Aspekte wie Parameter, Verhalten, Testablauf, Prüfpunkte, Testprotokoll, notwendige Änderungen der bestehenden Module und Richtlinien für die Verwendung Frameworks im Vordergrund stehen. Der dafür geplante Aufwand beträgt etwa 5 Personentage.

Weiterhin werden Mitarbeiter aus der Entwicklung bezüglich der Erweiterungen des Build-Prozesses um automatisierte Integrationstests und im Umgang mit den Testprotokollen geschult werden. Hier stehen

insbesondere Aspekte wie Konfiguration und Ablauf der Integrationstests, Testprotokoll (Inhalt, Ablage, Umgang) und die Ablage des Testkontextes für spätere Analyse im Vordergrund. Der dafür geplante Aufwand beträgt auch etwa 5 Personentage.

Schließlich werden Mitarbeiter aus Entwicklung und Kundendienst im Umgang mit dem Testwerkzeug geschult werden. Hier stehen insbesondere Aspekte wie Tool-Einsatz, Testablauf, Testprotokoll (Inhalt, Ablage, Umgang) im Vordergrund. Der geplante Aufwand hierfür beträgt etwa 2 Personentage.

3.3 WIKON Kommunikationstechnik GmbH

Bei WIKON wurde mit PETA ein Werkzeug zur systematischen Spezifikation und automatisierten Ausführung von Tests der Weboberfläche eines zentralen Produkts eingeführt. Vor der Einführung wurden sämtliche Tests der Weboberfläche individuell vom jeweiligen Entwickler durchgeführt. Die Dokumentation dieser Tests war lückenhaft und ebenfalls von einzelnen Entwicklern abhängig.

Da PETA manuelle Tests ersetzte, war keine echte Integration in die bestehende Werkzeuglandschaft notwendig. Die Testdaten werden jedoch in die Produktdokumentation integriert, d.h. spezifizierte Testfälle und dokumentierte Testergebnisse werden in die Dokumentation des jeweiligen Produkts in der entsprechenden Version integriert.

Die Schulung der Entwickler zur Benutzung von PETA erfolgte in zwei Durchgängen. In einem Pilotprojekt wurden zwei Entwickler von verit in die Benutzung des Werkzeugs (Testfallerstellung, Testausführung, Ergebnisinterpretation) eingewiesen. Der initiale Aufwand hierfür belief sich auf wenige Stunden. Während des Pilotprojekts half der verit-Support bei auftretenden Fragen weiter.

Nach Abschluss des Pilotprojekts wird die weitere Schulung der WIKON-Entwickler aus Kostengründen durch Inhouse-Workshops stattfinden. Als First-Level-Support dienen die Pilot-Entwickler, als Second-Level-Support der Support von verit.

4 Arbeitspaket 8: Evaluation

4.1 Einleitung

In diesem Kapitel werden die eingeführten Veränderungen im Testprozess und in der Testautomatisierung mittels Tools bei den Projektpartnern beschrieben.

4.2 market maker Software AG

Durch interne Schulungsmaßnahmen und die wöchentlichen Entwicklergespräche besser ausgebildete und motivierte Entwickler erstellen besser testbare Anwendungen, die sich bereits früher im Entwicklungsprozess testen lassen. Außerdem werden ersten Expertenschätzungen zufolge mehr Fehler früher im Entwicklungsprozess gefunden. Das weiterlaufende Messprogramm wird diese Schätzungen überprüfen.

Die Automatisierung von Integrations- und Installationstests setzt darüber hinaus Ressourcen frei, mit denen die Testabdeckung erhöht werden kann. Auch dies wird durch das Messprogramm bestätigt werden.

Schließlich hat sich die Entwicklungszeit neuer Produktversion verkürzt, weil das Ausmaß von Rework-Zyklen abnimmt. Damit kann die market maker Software AG schneller auf neue und geänderte Anforderungen reagieren.

4.3 WIKON Kommunikationstechnik GmbH

Der Einsatz des Testframeworks PETA ermöglicht es, den bisher etwas vernachlässigten Bereich des Testens der Weboberfläche eines zentralen WIKON-Produktes wieder in den Vordergrund zu holen. Der systematische Ansatz sorgt insbesondere für die folgenden Verbesserungen:

Qualität

Die Weboberfläche des WatchMyHome-Servers wurde bisher auch schon getestet, allerdings nicht systematisch. Daraus folgte, dass jeder Test andere Dinge überprüfte, insbesondere war nach Abschluss einer Testreihe nicht sichergestellt, dass alle kritischen Funktionen getestet waren.

Der Wechsel hin zu systematisch spezifizierten und automatisiert durchführbaren Tests behebt diese Situation. Insbesondere ist durch die praktisch ohne Aufwand durchführbaren Regressionstests sichergestellt, dass genau spezifizierte Funktionalitäten zuverlässig getestet werden. Damit ist deren Funktionieren im Live-Betrieb garantiert.

Zeitersparnis

Dadurch, dass die Tests an sich komplett automatisiert ablaufen, ist die Zeitersparnis beim Testen immens. Gegenüber der manuellen Bedienung der Weboberfläche kann eine Zeitersparnis von deutlich über 90% realisiert werden. Gleichzeitig fällt weniger Dokumentationsaufwand an, da das Testframework die ablaufenden Tests selbständig dokumentiert. Der Aufwand zur Spezifikation der Testfälle steigt hingegen geringfügig, da die Spezifikation formaler als bisher erfolgt (und vor allem auch zwingend erforderlich ist). Insgesamt sinkt der Aufwand für Tests der Weboberfläche jedoch bedeutend, insbesondere in Bezug auf Regressionstests.

Kostenersparnis

Bedingt durch den sinkenden Aufwand sinken auch die Kosten für den Test der Weboberfläche proportional. Durch die steigende Qualität des Produkts „WatchMyHome“ sinken indirekt auch die Kosten, indem beispielsweise das Risiko, dass Fehler in der Weboberfläche verbleiben und deshalb Regressforderungen aufkommen, verkleinert wird. Auch wenn solche Kostenpositionen schwieriger zu ermitteln sind als Aufwand o.ä., so müssen sie doch in Betracht gezogen werden.

Fazit

Der Wechsel von einer ad-hoc-Teststrategie zu einem systematischen, werkzeuggestützten Ansatz führte bei WIKON zu einer höheren Produktqualität, deutlicher Aufwandsreduktion beim Testen und dadurch bedingt einer Kostenersparnis. Das Pilotprojekt zeigt, dass auch bei kleinen und mittleren Unternehmen mit beschränktem Budget eine Testautomatisierung möglich ist und erfolgreich durchgeführt werden kann.

Literatur

[AAB+05] Samir Amiry, Ove Armbrust, Julia Berger, Janine Klinck, Konstantin Luttenberger: Research Lab Rheinland-Pfalz Testen und Testautomatisierung: Anforderungen an Testwerkzeuge und Marktstudie. IESE-Report Nr. 131.05/D, 2005.

[CC] CruiseControl. ThoughtWorks, Inc., http://cruisecontrol.source-forge.net

[VMW] VMWare Workstation und VMWare Player. VMWare Inc., http://www.vmware.com

Dokumenten-Information

Titel:	Research Lab Rheinland-Pfalz Testen und Testautomatisierung: Implementierung, Einführung, Evaluation der Lösungen
Datum:	19. April 2006
Report:	IESE-047.06/D
Status:	Final
Klassifikation:	Öffentlich