# NACHLESE
# 4. ARBEITSKONFERENZ SOFTWAREQUALITÄT UND TEST (ASQT 2006)

**ÜBERBLICK UND DETAILS ZU EINIGEN AUSGEWÄLTEN VORTRÄGEN**

## ÜBERBLICK

Vom 14. – 15.09.2006 hat an der Universität Klagenfurt, Österreich, bereits die 4. ASQT stattgefunden. Insgesamt wurden 2 Keynotes und 14 Vorträge auf sehr hohem fachlichem Niveau präsentiert. Einige der Vorträge werden in dieser Nachlese detaillierter behandelt.

## DIE VORTRÄGE

### KEYNOTE 1- INTEGRATIONSTEST IST (AUCH) ETWAS ANDERES, KAROL FRÜHAUF, INFOGEM AG, 5400 BADEN, SCHWEIZ

Karol Frühauf nähert sich dem Thema über die Begriffe Integration (z.B. Herstellen eines Ganzen, System zusammenbauen, System einbauen) und Test (Ausführen einer Einheit/eines Gerätes zum Zwecke einer Prüfung oder eines Versuchs) und schließt die Einleitung seines Vortrags mit der Definition „Integrationstest“ aus den Normen ab:

- Test ausgeführt mit dem Ziel, Fehler in Schnittstellen und in der Interaktion zwischen Komponenten zu entdecken. [BS7925-1]
- Test in dem Software-Komponenten, Hardware-Komponenten, oder beide kombiniert getestet werden, um die Interaktion zwischen ihnen zu bewerten.[IEEE 610.12]
- Verifikation der Interaktion zwischen System-Komponenten. [SWEBOK 2004]

In weiterer Folge legt er das Prinzip des Funktionstests und des Schnittstellentests dar und beschreibt das jeweils nötige Testgeschirr. Anschließend werden die Merkmale des Schnittstellentests und die Unterschiede zum Funktionstest dargelegt. Wesentlich ist ihm auch, darauf hinzuweisen, dass es keinen Integrationstest, sondern nur Integrationtest**s** gibt (integrierte Lösungsteile, integriertes System) und legt dar, dass sich die Schnittstellentests je nach Schnittstellenart unterscheidet, z.B. Unterprogrammaufrufe, APIs, Datenübertragungsprotokolle, Files, Datenbank, Interaktion zwischen online/batch, Prozesssynchronisation, shared resources.

Er führt folgende Arten von Schnittstellenfehlern an, die jeweils mit Beispielen beschrieben werden:

- Verbindungsfehler
- Kommunikationsfehler (Missverständnisse in der Kommunikation)
- Interaktionsfehler (missverständliche Kommunikationsabfolge)
- Robustheitsfehler (Mängel in Ausnahme- und Grenzfällen)
- Parametrierungsfehler

Anschließend werden Schnittstellentests von Designdarstellungen mit dem jeweils schwächsten Testkritierium herausgearbeitet, z.B.

- Aufrufbaum: Ausführung jedes Aufrufs mindestens ein Mal
- Sequenz-/Kollaborations-Diagramme: jede Nachrichtenart zumindest ein Mal austauschen
- Zustandsübergangs-Diagramm: jeden Zustand mindestens ein Mal erreichen
- physische Architektur: jede Verbindung mindestens ein Mal ansprechen; bei jeder Verbindung ein Mal das Fehlen des Partners (beide Seiten) herstellen
- Prozess-/Jobablaufdiagramme: jeden Prozess/ Job mindestens ein Mal ausführen

Weiters widmet er sich dem Thema Fehlerquellen und stellt die Frage, warum häufig nur nach Entwicklungsfehlern gesucht wird, Fehler in der Integration aber nicht betrachtet werden. Die vielfältigen und komplexen Aufgaben der Integration bieten eine Fülle an Fehlerquellen. Dies wird bei folgender Definition des Integrationstests klar:

- Test ausgeführt mit dem Ziel, Fehler zu entdecken, die man bei der Integration begeht, d.h. Fehler bei der Auswahl der Integrationsobjekte, der Scripterstellung (Funktionstest von Scripten) für den Zusammenbau, bei An- und Einbau des Systems in die Systemlandschaft, sowie Variantenbau und Abbau.

Integrationstest ist daher

- Schnittstellentest und
- auch Test auf Integrationsfehler

Seine abschließende Frage lautet:
Wie systematisch testen wir auf diesem Gebiet?

### STANDARDS, NORMEN, PRÜF- UND ZERTIFIZIERUNGSMÖGLICHKEITEN IM IT-BEREICH, RELEVANZ UND RECHTLICHE ASPEKTE, DIPL.-ING. JOHANNES BERGSMANN, SOFTWARE QUALITY LAB GMBH, LINZ

Herr Bergsmann eröffnet seinen Vortrag mit der Darstellung, was Normen sind: Regeln der Technik mit den Grundprinzipien „neutrale Gemeinschaftsarbeit, Konsens, Publizität und Widerspruchsfreiheit". Sie sind nicht gesetzlich verpflichtend, beschreiben aber den Stand der Technik und sind im Streitfall auf jeden Fall relevant.

Normen werden einerseits in internationalen Gremien (ISO, IEC, CEN, …) und andererseits auch national (ÖNORM, DIN, BSI, ...) erarbeitet. ISO, IEC und IEEE bieten für den IT- und Softwarebereich eine fast nicht mehr überblickbare Anzahl an Regelungen an. Insbesondere wird auf die neue Sicherheitsnorm ISO-27000, die die BS 7799 ablösen wird und die ISO-25000 (SQuaRE = Software product Quality Requirements and Evaluation) hingewiesen.

In weiterer Folge stellt er die Möglichkeiten der Zertifizierung und die Arbeitsweise von Prüfstellen, Wert und Nutzen von Zertifikaten, Alternativen zur Zertifizierung, rechtliche Relevanz, Haftungsfragen, den rechtlichen Status und die zukünftige Entwicklung dar.

### DAS VORGEHEN BEIM SOFTWARE-TEST – STRUKTURIERT, METHODISCH UND ANGEMESSEN, DR. LUTZ BREDEN, COMMERZBANK AG, FRANKFURT

Das Testhandbuch der Commerzbank enthält das zentrale Regelwerk für das Testen. Es umfasst 330 Seiten, ist prozessorientiert aufgebaut und in Prozesscluster (Überblick, Abgrenzung), Testprozess (Beschreibung, Auswahlkriterien), Vorgehen (Beschreibung, Aktivitätendiagramme) und Aktivität/Teilaktivität gegliedert. Der Prozesscluster umfasst

- Dispositive Testprozesse
  - Planung und Management (Steuern und Messen) aller Testaktivitäten
- Operative Testprozesse
  - Testdurchführung, Testen der Zwischen- und Endprodukte der Softwareentwicklung
- Administrative Testprozesse
  - Schnittstellenprozesse, Konfigurationsmanagement, Fehler-/Changemanagement
- Prozesse zu den Testumgebungen
  - Bereitstellung der Testsysteme, Testumgebungen und Testdaten
- Management-relevante Testprozesse
  - Unternehmensweite Vorgaben und Rahmenbedingungen wie Testpolitik / -kultur, Mitarbeiterqualifizierung für Testmethodik, -werkzeuge etc.

Die einzelnen Prozesse jedes Clusters werden kurz erläutert und das Testdokumentationssystem sowie die definierten Rollen dargestellt.

Herr Dr. Breden weist darauf hin, dass das Testhandbuch ein Leitfaden ist und nicht jedes Projekt alles umsetzen muss. Das Schlüsselwort heißt Tailoring. Maßgebend für die Anwendung des Handbuchs sind die Erfahrungen des Testmanagers und des Projektleiters. Das Testhandbuch ist ein Werkzeugkasten zur

- Auswahl der relevanten Aktivitäten, Vorgehen und Methoden
- Auswahl der Vorlagen und sonstiger Hilfsmittel
- Nutzung der Vorlagen als Gestaltungsbasis für das Teilprojekt Test
- Anpassung von Abläufen und Dokumenten für das jeweilige Projekt.

Insbesondere gilt es auch die Fragen zu beantworten:

- Was ist wirtschaftlich vertretbar?
- Was ist technisch/organisatorisch umsetzbar?

Die Commerzbank AG hat für die Tailoringentscheidungen folgende Kriterien festgelegt:

- Projektart (Neuentwicklung, Release, Wartung, Migration)
- Technik (Host, Client/Server, Standard-SW)
- Projektgröße (Anzahl Testfälle, Dauer)
- Rahmenbedingungen (Budget, Zeit)
- Beteiligte am Test (Qualifikation, Erfahrung, Verfügbarkeit, Unterstützung durch Service-Einheiten)
- Testziele (Qualitätsmerkmal)

Anhand von drei Projekten unterschiedlicher Größe und Charakteristik stellt Herr Dr. Breden das Tailoring (operative Testprozesse, Testdokumentation, Testintensität, Testrollen, Testumgebung, Testwerkzeuge) anschaulich dar.

### USABILITY - MYTHEN, FAKTEN, ANWENDUNGSSZENARIEN, MARTIN HITZ, GERHARD LEITNER, FORSCHUNGSGRUPPE »INTERAKTIVE SYSTEME«, INSTITUT FÜR INFORMATIK-SYSTEME, ALPEN-ADRIA-UNIVERSITÄT KLAGENFURT

Der Vortragende behandelt sechs Mythen rund um die Usability:

- Usability kann man automatisch, wenn man IT-Experte ist.
- Usability behindert die Kreativität.
- Usability ist nur deshalb erforderlich, weil es viele dumme Benutzer gibt.

- Usability besteht nur aus Testen.
- Usability bedeutet einfach das Berücksichtigen von Normen oder Standards.
- Usability verursacht zusätzliche Kosten, bringt aber kaum etwas.

Alle Mythen werden mit Fakten aus der praktischen Erfahrung des Instituts auf anschauliche und amüsante Art und Weise widerlegt.

### WIRTSCHAFTLICHES TESTEN - VON DER GEPLANTEN TESTABDECKUNG ZUR GEPLANTEN TESTLÜCKE, JOHANNES KREINER (GENERALI ÖSTERREICH, WIEN)

Die Basis wirtschaftlichen Tests wird in der Generali gesehen in

- Risiko basierendem Testen
  - Durchgängige Anwendung in allen Testphasen und Life-Cycle Phasen
- Life-Cycle orientiertem Testen
  - Anpassung der Testmethoden an den Life-Cycle der Anwendung

Ein weiteres Unterscheidungsmerkmal ist die Art der Projekte (Neuprojekt oder CR).

In der Entstehungsphase (Neuprojekte) erfolgt die Risikobewertung für die Testfallfindung mit Hilfe der Anforderungsanalyse und dem Anforderungsbaum anhand einer Matrix Fehlerwahrscheinlichkeit zu Fehlerkosten.

Da die Kosten für Verwaltung/Wartung pro Testfall bei 10€ - 15€ / Jahr liegen, werden strukturiert dokumentierte Testfälle in der Entstehungsphase nur für die Risikobewertungen „high“ und „medium“ erstellt.

Die Priorisierung von Testfällen erfolgt anhand der Kriterien „Gibt es ähnliche Testfälle (z.B. Grenzwert)“ und „Abdeckungsgrad des Testfalls (z.B. Vorbedingungen)“. Sie liefert eine Entscheidungshilfe bei begrenzter Testzeit und für die Durchführungshäufigkeit. Darüber hinaus gibt sie Anhaltspunkte für Reduktionspotentiale, also die zukünftigen Testlücken.

Der Testfalldokumentationsgrad wird im Life-Cycle schrittweise verfeinert - von der Grobbeschreibung zum Testfallablauf, dann erst von den groben Testdaten zu den Testdaten.

In der Entstehungsphase ist die Testautomation wirtschaftlich problematisch, weil sich die Anforderungen noch ändern können und dadurch hoher Wartungsaufwand für die detaillierte Testfalldokumentation entsteht. Besser ist es in dieser Phase, in allen Risikobereichen manuell heuristisch und in den Risikobereichen „high“ und „medium“ mit strukturierten Tests manuell zu testen. Einzig in langen, umfangreichen Entwicklungsprojekten sollte mit Testautomation in den „high“ bewerteten Bereichen, allerdings nur in nachweislich stabilen, frühzeitig begonnen werden. Diese automatisierten Testfälle sollten einen sehr hohen Funktionsabdeckungsgrad erreichen.

Für die Testfallfindung in der Releasephase lautet das Motto „Nur ein wichtiger und richtiger Testfall hat eine Lebensberechtigung“. Es müssen der Life-Cycle von Anforderungen und Testfällen beachtet und die Risikobewertungen überarbeitet werden.

Ziel ist es, auf lange Sicht keine Vergrößerung des Testfallportfolios zuzulassen. Dazu müssen Testlücken bewusst geöffnet werden. Ansatzpunkte sind z.B. fehlerfreie Bereiche und die Priorität eines Testfalls.

### KEYNOTE 2 - SYSTEM- UND SOFTWAREENTWICKLUNG IN EINEM DYNAMISCHEN UND GLOBALEN UMFELD - HERAUSFORDERUNGEN UND LÖSUNGSANSÄTZE IM PROJEKT- UND QUALITÄTSMANAGEMENT, MAREK LESZAK (LUCENT TECHNOLOGIES, NÜRNBERG)

Lucent betreibt Großprojekt-Softwareentwicklung für embedded systems. Die Herausforderungen des Marktes liegen in rasch wechselnder Technologie und Standards, rasant wachsendem komplexen Softwareanteil, eine über drei Jahre nicht hinausreichende Vorhersehbarkeit der technologischen Entwicklung, zu vielen Wettbewerbern in einem nur langsam wachsenden Markt und den Business-Prioritäten time-to-market neuester Technologie, Kosten und Qualität.

Schlüsselfaktoren zum Erfolg sind

- Produkt- und Projektmanagement-Fokus
  - Globale Entwicklung, starke System-Engineering- und Software-Architekturteams, Multi-release Management (späte “neue” und gestrichenen “alten” Anforderungen)
- Qualitäts-und Prozess-Fokus
  - TL9000 Zertifizierung und CMMI Maturity Level 3 Programm, “bottom-up” definierte Prozesse
- Innovation und Technologie-Fokus
  - Softwaretechnologie, best-practices-Etablierung durch ein zentrales CMMI Team
- People Management

Beispielsweise schafft es ein Systemengineering-Team, bei ca. 10 000 – 30 000 Anforderungen pro Produkt, dass nur 4% aller Produktivfehler auf Fehler in der Anforderungsphase zurückzuführen sind. Bei einem konstanten Anteil von Anforderungsänderungen pro Release entsteht dabei

nur 1 Anforderungsfehler je 6-22 neuen Anforderungen.

Das Ziel des Produkt- und Releasemanagements ist es, neue Produkte mit hoher Qualität und sinkenden Kosten zu erzeugen, die die Anforderungen des Marktes erfüllen oder übertreffen. Dies wird mit einer Produktlinien/Plattformarchitektur-Strategie, Minimierung der Kundenbeschwerden und Reduktion zwingend notwendiger Wartungs-Releases sowie starkem Qualitätsfokus (TL9000-Zertifizierung, CMMI Level 3 Programm) angestrebt.

Die typische Releasezeiten liegen bei 6 Monaten für Feature Releases und für Wartungsreleases bei 2-3 Monaten nach dem Featurerelease. Der gesamte Produktlebenszyklus umfasst 5-15 Jahre Marktpräsenz und 5-10 Jahre an Entwicklungszeit. Die Projektdauer liegt bei ca. 9 Monaten.

Die Gesamtverfügbarkeit wird durch zwei Gates, das Quality Gate "end of system test" und das Gate "volume launch readiness" sichergestellt. Folgende Exit-Kritierien werden für das Ende des System Tests gefordert:

- Alle zugesagten Features sind vorhanden
- Alle Fehler mit severity 1&2 sind beseitigt
  - Den Kunden bekannte Fehler sind bewertet, die Behebung zugesagt,
  - 75% Defect Removal Efficiency
- Das Ziel für Produktivfehler im 1. Jahr liegt bei maximal 2 Fehlern der Klassen 1 & 2

Agilität wird u.a. in folgenden Phasen angewandt:

- Beteiligung der Kunden (Market und Sales, Produktmanagement, Systems Engineering)
- Feature-Iteration: 2-8 Wochen
- Entwicklung komplexer Features: 2-4 Wochen
- wöchentliche Softwareentwicklung/-Integration/-Systemverifizierung
-

Das Qualitätswesen stützt sich auf

- TL9000/ISO9000-Zertifizierung von 150 Standorten und 75 Produkten
- Einheitliche strategische Prozesse und Tools, Ziele werden in der Balanced Score Card definiert und je Quartal erhoben und bewertet
- Customer Satisfaction Programm

Der CMMI Maturity Level 3, wurde vom CEO als Businessziel gesetzt, die Mobility R&D Organisation hat den Level 3 erreicht. Das Senior Management hat folgendes Erfolgskriterium definiert: "Wenn das CMMI-Deployment zur Reduktion von 1-2 Maintenancereleases im Jahr führt, ist dies ein positiver ROI der CMMI-Implementierung". Weiters wird CMMI von den Kunden immer stärker gefordert. Darüber hinaus unterstützt es die Verbesserung der System- und Produktqualität und dient dazu, Best Practices unternehmensweit zu etablieren.

### SOFTWAREMETRIKEN ALS GRUNDLAGE ZUR UNABHÄNGIGEN BEURTEILUNG KRITISCHER UNBEKANNTER SOFTWARE, CHRISTIAN REUMANN, CHRISTIAN SEJKORA (ARC SEIBERSDORF RESEARCH, WIEN)

Die ARC Seibersdorf research GmbH betreibt ein nach ISO/IEC 17025 akkreditiertes Software Prüflabor und bietet die Dienstleistung der unabhängigen Beurteilung von Software an. Auslöser für derartige Prüfungen können sein:

- Wunsch nach einem unabhängigen „Attest"
- Zukauf von (embedded) Software
- Zustandsüberprüfung aufgrund von Mitarbeiterfluktuation

Im Vortrage wird ein Bewertungsansatz mit statischer Analyse vorgestellt. Ausgehend vom Goal/Question/Metrik-Vorgehen wird ein Beispiel anhand der Metrik „Zyklomatische Komplexität" durchgespielt und dargestellt, dass mit dieser einfach zu berechnenden Metrik sehr gute Erkenntnisse gewonnen werden können.

Die zyklomatische Komplexität

- bewertet die Komplexität einer Methode/ Funktion,
- beruht auf der Komplexität des Kontrollflussgraphen,
- misst die Anzahl der linear unabhängigen Pfade und
- kann auch zur Bewertung des Risikos von Programmen herangezogen werden.

| Zyklomatische Komplexität | Risiko |
|---|---|
| 1-10 | einfach, risikoarm |
| 11-20 | komplexer, mäßiges Risiko |
| 21-50 | komplex, hohes Risiko |
| >50 | instabil, sehr hohes Risikos |

Mit der statistischen Auswertung über Häufigkeit (Säulendiagramm, Klassen 1-10, 11-20, …) und Verteilung der Module (Streudiagramm „Anzahl der Module zu zyklomatischer Komplexität") gewinnt man einen Überblick. Ausreißer und Cluster werden deutlich sichtbar. Durch Einteilung der Grafik in „Bereiche" werden die kritischen Module lokalisiert, die dann gezielt betrachtet werden. Analysiert man anschließend, in welchen Modulen die meisten Fehler gefunden wurden, wird in der Regel eine hohe Korrelation „viele Fehler in den komplexen Modulen" festgestellt werden. Im präsentierten Beispiel lag die Fehlerzahl für die komplexen Module bei 45% der Gesamtanzahl.

Diese einfache Methode eignet sehr gut, Qualitätssicherungsmaßnahmen (z.B. Codereviews), ökonomisch einzusetzen.

### COMPUWARE QUALITY MATURITY MODEL – SIND SIE REIF FÜR BESTMÖGLICHE QUALITÄT?, KURT AIGNER (COMPUWARE AUSTRIA GMBH, LINZ)

Laut CEO Magazine und Gartner fließen derzeit nur 21,8% der IT-Budgets in Innovation, 78,2% werden zur Sicherung existierender Werte eingesetzt. Für die Anwendungsqualität bestehen folgende Herausforderungen:

- Produktivität – zeitgerecht liefern
- Kundenloyalität – Anforderungen erfüllen
- Geschäftswachstum – ROI & Ziele erreichen
- Geschäftsrisiken – in allen Projektphasen

Auf der Basis einer von Compuware beauftragten Forrester-Studie wurden in Interviews mit CEOs folgende Kriterien für bessere Qualität ermittelt:

- Kontrolle der wichtigsten Metriken
- Konsistente Methodologien
- Prozessoptimierung

Darauf aufbauend hat Compuware sein vierstufiges Reifegradmodell Compuware Quality Maturity Model (CQMM) entwickelt, das aus den Bausteinen Methodologie – Prozesse – Metriken besteht, die durch die weiteren Bausteine Tools und Ressourcen unterstützt werden.

Wodurch zeichnen sich nun die einzelnen Stufen des CQMM aus?

- **Level 1: Qualitätskontrolle - Fehler finden**
  - Treibende Kraft ist die Kostensenkung
  - Einschätzung der Produktqualität anhand festgelegter Standards
  - **Identifizierung und Eliminierung** von Fehlern, die definierte Qualitätsstandards nicht erreichen und im Produktiveinsatz negativen Einfluss auf das Geschäft ausüben
  - **Maßnahmen**
    - Test-Team
    - Funktionales Testen
    - Involviert Produkte
    - Bewerten der Testergebnisse
    - Entwickler und Tester treffen Entscheidungen

- **Level 2: Qualitätssicherung – Fehler minimieren**
  - Treibende Kraft ist das Risikomanagement
  - Kontinuität der Produktqualität anhand festgelegter Standards sicherstellen
  - **Vermeidung und Minimierung** von Fehlern, die definierte Qualitätsstandards nicht erreichen und im Produktiveinsatz negativen Einfluss auf das Geschäft ausüben
  - **Maßnahmen**
    - QS Lebenszyklus - Testphasen
    - Qualitätssicherungs-Team
    - Involviert Produkte und Personen
    - Einheitliche Prozesse
    - Leiter Qualität und Entwicklung treffen Entscheidungen

- **Level 3: Qualitätsmanagement - Risiken & Kosten abwägen**
  - Treibende Kraft ist die Service Optimierung
  - Gewährleistung der Produktqualität anhand festgelegter Prozessstandards sicherstellen
  - **Vermeidung und Minimierung von Fehlerursachen**, die definierte Qualitätsstandards nicht erreichen und im Produktiveinsatz negativen Einfluss auf das Geschäft ausüben
  - **Maßnahmen**
    - Gesamter IT Lebenszyklus
    - Zentralisiertes QS-Team
    - Involviert Produkte, Personen und Prozesse
    - Kontinuierliche Prozessoptimierung
    - Lines of Business punktuell eingebunden
    - IT Leiter trifft Entscheidungen

- **Level 4: Quality Governance - Werte maximieren**
  - Treibende Kraft ist die Geschäftsausrichtung
  - Gewährleistung des Geschäftserfolges durch gezielte Qualitätssteigerung
  - **Ermittlung der Erfolgsfaktoren**, die definierte Qualitätsstandards übertreffen und im Produktiveinsatz positiven Einfluss auf das Geschäft ausüben
  - **Maßnahmen**
    - ROI messen
    - Optimierung des IT Lebenszyklus
    - Weiterbildung des Teams
    - Integriert Produkte, Personen und Prozesse
    - Permanente Einbindung von LOB
    - Geschäftsführung trifft Entscheidungen

Betrachtet man die Reifegrade näher, kann man eine Anlehnung an die SPICE/CMMI-Reifegrade erkennen. Anhand der verfügbaren Informationen ergibt ein Vergleich die grobe Zuordnung:

- CQMM Reifegrad 1 - SPICE/CMMI Level 1
- CQMM Reifegrad 2 - SPICE/CMMI Level 2/3
- CQMM Reifegrad 3 - SPICE/CMMI-Level 3/4
- CQMM Reifegrad 4 - SPICE/CMMI-Level 4/5

Der Ansatz erscheint sehr praktikabel und zweckmäßig. Der leidgeprüfte IT-Verantwortliche hat allerdings nun ein weiteres Modell zur Auswahl. Es wird interessant zu verfolgen sein, ob und in welchen Bereichen sich dieses Modell etabliert.


A-1160 Wien, Lorenz Mandl-Gasse 47/29, Tel. & Fax: +43 1 493 28 15, Mobil: +43 (0) 676 3550 686
email: dietmar.kropfitsch@kropfitsch-consulting.com, http://www.kropfitsch-consulting.com

### Herausforderung „Automatisierter Software-Test“, Otto Opietnik (Oberösterreichische Gebietskrankenkasse, Linz)

Der Vortrag legt dar, wie die Oberösterreichische Gebietskrankenkasse (OÖ GKK) ihren Testprozess von einem anfangs instabilen Zustand (Testfälle nicht definiert, Testdurchführung nicht nachvollziehbar, Qualität der Testfälle abhängig vom jeweiligen Tester) hin zu einem Automatisationsgrad von 90% entwickelt hat. Als Voraussetzung für die Etablierung automatisierter Tests werden folgende Punkte genannt:

- Reifegrad des Entwicklungsprozesses: insbesondere Fehler- und Änderungsprozess sowie Konfigurationsmanagement
- Reifegrad des Testprozesses: Testplanung, Testspezifikation, ...
- Testbarkeit der Applikation: einheitlicher Aufbau, Entwicklungsstandards
- Mitarbeiterqualifikation
- Testumgebungen und Wiederholbarkeit: Testdatenbanken, Wiederherstellung der Vorbedingungen

Kostentreiber der Implementierung automatisierter Tests sind die Bereitstellung der Testinfrastruktur, die Integration der Testautomation im Projekt, die Implementierung des Frameworks, der Aufbau der Testorganisation, Durchführung der Tests und Wartung der Testfälle. Anfangs ist daher mit höherem Aufwand zur rechnen, der Break-Even kann frühestens zum Test des dritten Build erwartet werden.
Erwartungen und die Realität sehen etwa so aus:

- Manager erhoffen sich eine Kostenreduktion, mehr Fehler in der Software zu finden und die Qualität zu erhöhen
- Realität ist aber, dass Testfälle für die Automatisierung sogar kostenintensiver (3 - 10 mal höher) sind und die Rentabilität von der Flexibilität, Wartbarkeit und Lebensdauer des Testobjekts abhängt.

Die wesentlichen Erfahrungen mit Testautomatisierung fasst Herr Opietnik so zusammen:

- Eigene Testumgebung inklusive Datenbanken
- Die Applikation muss für die Automatisierung geeignet sein, eventuell sind Anpassungen notwendig
- Trennung logischer von konkreten Testfällen
- Die Testspezifikation muss vorhanden sein
- Bereitstellen der notwendigen Daten
- Je nach Projektfortschritt müssen die Testfälle angepasst werden

Wie schon Herr Kreiner ausgeführt hat, werden auch in der OÖ GKK die Testfälle in den Testschritten verfeinert:

- Die Analyse ist abgeschlossen, eine grobe Testspezifikation wird erstellt
- Das Design ist abgeschlossen, die feine Testspezifikation wird erstellt
- Die Software (Dialog) ist vorhanden, logische Testfälle werden erstellt, teilweise können schon konkrete Testfälle erstellt werden

Ziel ist es, den Zeitabstand von der Fertigstellung der Dialoge bis zur automatischen Testausführung zu minimieren.

Dias Testcenter hat bisher folgende Erfolge erzielt:

- 8000 Testfälle werden automatisiert durchgeführt
- 500 Dialoge werden berücksichtigt
- 45.000 unterschiedliche Testdatenkombinationen werden bearbeitet

Warum automatisiert die Sozialversicherung?

- Hoher Qualitätsanspruch der SV-Träger mit unterschiedliche Anforderungen muss berücksichtigt werden
- Automatisierung minimiert die Kosten und Risiken der ausgelieferter Software
- Erhebliche Reduktion von Wartungs- und Folgekosten
- Transparenz wird in die Entwicklung gebracht
- Manueller Testaufwand beschränkt sich auf ca. 10 % des Testaufwands
- Voraussetzung für systemübergreifendes Testen (automatisiertes TestIntegrations-Center TIC)

In Anbetracht der Ausgangslage ist das ein bemerkenswerter Erfolg einer Prozessverbesserungsinitiative.

### Bewertungsansätze und Modelle für unternehmensweite Software-Messinitiativen, Martin Kunz (Otto-von-Guericke Universität Magdeburg)

Ausgehend von zwei Ansätzen eines Corporate Measurement Programms (Rapid Prototyping, Goal/Question/Metric) wird ein entscheidender Erfolgsfaktor für Messinitiativen, die automatisierte Datensammlung besonders hervorgehoben. Die einzelnen Schritte Messung, Datensammlung, Speicherung und Extraktion sowie Analyse werden hinsichtlich der Problemfelder und den Best Practices im Detail betrachtet. Ein kurzer Überblick über die auf dem Markt verfügbaren Tools für die Automatisierung und die Präsentation von drei Metrics Database (MDB) Architekturen

- Measurement Data Warehouse
- Mediated Measurement Repository
- Service Bus-oriented Measurement Repository

stellen den Bezug zur praktischen Umsetzung her.
Abschließend werden zwei Forschungsschwerpunkte "Maturity Modeling of the Measurement Collection Process" und "Towards a Service Oriented Measurement Infrastructure" kurz skizziert.

### TESTAUTOMATISIERUNG UND AGILE SOFTWARE-ENTWICKLUNG – EIN WIDERSPRUCH?, ARMIN BEER (SIEMENS AG ÖSTERREICH, WIEN)

Einleitend werden die Probleme der agilen Softwareentwicklung (oft fehlt die genaue Planung und Steuerung der Inkremente, hoher Kommunikations- und Wartungsaufwand) und der Testautomatisierung (kurze Entwicklungszyklen, instabile Anforderungen, kurze Testphasen, späte Einführung der Testautomatisierung, …) angesprochen.

Automatisierte Komponenten-/Integrations-Tests sind laut Kent Beck dann erfolgreich, wenn sie wiederholbar sind, die Testergebnisse automatisiert ausgewertet werden können, rasches Feedback geliefert wird und das Entwicklungsteam den Fortschritt sieht.

Als Voraussetzungen für den automatisierten Komponenten-/Integrations-Test nennt Herr Beer das Vorliegen einer Architektur der Anwendung, Schulung der Entwickler (Testmethodik, Tools) und die Verfügbarkeit einer Testumgebung mit Open-Source-Tools. Die Implementierung kann durch Erstellung von XML-Testfall-Beschreibungen und Testscripten, tägliche Aktualisierung und Ausführung der Testfälle, Ausführung und Protokollierung der Testfälle im Rahmen des nächtlichen Build, Bereitstellung der Testberichte über einen Webserver und Unterstützung der Entwicklung vom Testdaten-Administrator erfolgen.

Automatisierte Systemtests erfordern, dass der Status der Anforderungen immer aktuell gehalten wird, die GUI testbar ist und eine Qualitätsaussage innerhalb eines Tages möglich ist.

Um die Testautomatisierung zu realisieren, sollte der „Prove of concept" erfolgreich verlaufen sein, ein Auswahl manueller Testfälle vorliegen und ein Automatisierungs-Spezialist verfügbar sein. Es wird folgende Vorgangsweise vorgeschlagen:

- Erstellung einer Automatisierungs-Umgebung (Testrahmen, „Frontend" zur Testfall-Wartung)
- Erfassung/Wartung der Test-Scripte/-daten
- Durchführung von Regressionstests nach Freigabe einer Version oder eines Inkrements
- Erstellung eines Testberichts

In einem Fallbeispiel wird die Testautomatisierung in der Sozialversicherung für eine Java-WEB-Applikation vorgestellt. Interessant ist der schrittweise Ansatz: im ersten Inkrement werden „nur" Modultests, ab dem zweiten Inkrement Modul-, System- und Lasttests, im dritten Inkrement auch der Abnahmetest automatisiert durchgeführt. Insgesamt wurden ca. 240 Unittestfälle und 300 von 1100 Systemtestfällen automatisiert, die Regressionsläufe dauerten ca. 5-7 Stunden, der ROI wurde nach ca. 6 Testdurchläufen erreicht.

Herr Beer erläutert den Ansatz der Testfallgenerierung mit der Siemens-Methode IDATG (Integrating Design and Automated Test Case Generation)

- Spezifikation von API und GUI Verhalten auf 3 Abstraktionsebenen mit Hilfe graphischer Editoren (Requirement Spezifikation, Taskflow-Modellierung, Low-Level Spezifikation)
- Zuordnung von Requirements zu Test-Paketen
- Ein integrierter GUI Spy erlaubt die Zuordnung von Test-Steps zu GUI-Elementen (Low-Level Spezifikation)
- Generieren von Test-Scripts in verschiedenen Formaten (XML, SilkTest, WinRunner etc.)

Im zweiten Fallbeispiel, dem automatisierten Test des Spacecraft Operating Systems SCOS-2000 der ESA, werden die Anwendung der IDATG und das eingesetzte Testframework beschrieben, die seit 2004 erfolgreich eingesetzt werden.

Abschließend werden die Erfolgsfaktoren für die Testautomatisierung nochmals hervorgehoben:

- definierte Anforderungen,
- testbare Benutzeroberflächen,
- ein eingeführtes Testmanagement und
- Investitionen in Tools und Spezialisten.