NACHLESE
5. ARBEITSKONFERENZ SOFTWAREQUALITÄT UND TEST (ASQT 2007)

ÜBERBLICK UND DETAILS ZU EINIGEN AUSGEWÄLTEN VORTRÄGEN

ÜBERBLICK

Vom 20. – 21.09.2007 hat an der Universität Klagenfurt, Österreich, nunmehr die 5. ASQT stattgefunden. Insgesamt wurden 2 Keynotes und 13 Vorträge auf sehr hohem fachlichem Niveau präsentiert. Diese Nachlese fasst die Kernaussagen aller Vorträge zusammen.

DIE VORTRÄGE

KEYNOTE 1- TEST NICHTFUNKTIONALER ANFORDERUNGEN BEI SAP, DR. MARTINA WROBLEWSKI, DR. ROLF ZIEGLER

Die Erfüllung von sicherheitsrelevanten oder gesetzlichen Anforderungen erfordert, diese auch im Test zu berücksichtigen.

SAP leitet die nicht funktionalen Anforderungen vom Qualitätsmodell ISO 9126 ab, interpretiert sie spezifisch, erweitert sie um zusätzliche Anforderungen (z.B. Open Source, Third Party) und erklärt sie zu Produktstandardanforderungen.

Die Einhaltung der SAP-Produktstandards wird durch Architekturrichtlinien, Installationsleitfäden, Third Party Vereinbarungen und verschiedene Testtypen (White-Box, Black-Box, Implizite Tests) sichergestellt. Die tatsächliche Erfüllung der Anforderungen wird an Quality Gates geprüft.

Abhängig vom Produktstandard kommen folgende Testmethoden zum Einsatz, um die Konformität nichtfunktionaler Anforderungen sicherzustellen:

- Fehlervermeidung durch toolunterstützte Programmierung, die anhand von Regelprüfungen die Generierung nicht-standardkonformen Codes verhindert
- White Box Test:
  - Automatische Code-Prüfung, unterstützt durch selbst entwickelte workflowbasierte Checktools zur Problemlösung
  - Code Reviews und Code-Inspektionen
- Black Box Test
  - Manuelle und automatische Tests nichtfunktionaler Anforderungen
  - Implizite Tests - ein Teil der Anforderungen wird in funktionalen Tests abgedeckt, z.B. Anforderungen an die Internationalisierung

Zentrale Testservices, die weiter ausgebaut werden sollen, unterstützen die Testaktivitäten.

Die Integration der nichtfunktionalen Anforderungen in den Entwicklungsprozess erfolgt derzeit durch teilautomatisierte Spezifikationschecks. Angestrebt wird eine toolunterstützte Konformitätsprüfung. Die vollständige Nachverfolgbarkeit erfolgt auf der Basis der Anforderungsspezifikation.

SAP will in Zukunft lokale Tools durch ein zentrales Produktstandard-Compliance-Planungstool ablösen, nicht-ABAP Werkzeuge und Unit Test integrieren sowie einheitliches Reporting der Produktstandard-Compliance durch das Zusammenführen von Statusinformationen aus unterschiedlichen Datenquellen aufbauen.

VERTRAGSGESTALTUNG, SPEZIFIKATION UND QUALITÄTSSICHERUNG IN KOMPLEXEN SAP-PROJEKTEN, DIPL.-ING. JOHANNES BERGSMANN, SOFTWARE QUALITY LAB GMBH, LINZ

Bei der Einführung eines Standardproduktes (z.B. SAP) sind folgende Fallstricke und Problembereiche zu berücksichtigen:

- Lieferantenauswahl
- Projektbeginn
- Vertragsgestaltung
- Anforderungsspezifikation
- Testen
- Beauftragung
- Datenmigration
- Schnittstellen
- Detailliertheit und Verständlichkeit

Als Lösungsansatz werden verschiedene Vertragsarten präsentiert:

- Spezifikationsvertrag (pauschal oder nach Aufwand)
- Umsetzungsvertrag (meist pauschal)
- Letter of Intent (optional) für einen eventuell auf den Spezifikationsvertrag folgenden Umsetzungsvertrag
- Wartungsvertrag (optional)

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

Beim Spezifikationsvertrag sind die genaue Festlegung des Vertragsgegenstandes (Erstellung eines Pflichtenheftes) und die Regelung des Change Management (insbesondere bei pauschaler Beauftragung) zu berücksichtigen. Ebenfalls wichtig sind Regelungen hinsichtlich der Gewährleistung für die Anforderungen an das Pflichtenheft (z.B. eindeutige Interpretation durch einen fachkundigen Dritten, Realisierung auch durch Mitbewerber, inhaltliche Detaillierungstiefe, Qualität). Inhaltliche Qualitätsvorgaben können z.B. bei Masken vorgeben, dass Layout, Felder, Feldeigenschaften, etc. beschrieben sein müssen.

Im Umsetzungsvertrag ist es wichtig, die Rechte an den Ergebnissen zu regeln, Projektorganisation und -vorgehen sowie das Abnahmeprozedere genau zu vereinbaren.

Abschließend wird insbesondere darauf hingewiesen, den Themen Schnittstellen, Migration, Test und Abnahme ausreichend Aufmerksamkeit zu widmen und den dafür notwendigen Aufwand nicht zu vernachlässigen oder zu unterschätzen.

Wie kann man im Outsourcing die Software-Qualität und Performance der Lieferanten steuern? Manfred Seufert (Mediaan ABS Deutschland GmbH, Düsseldorf)

Der Vortragende legt dar, dass Outsourcing von Applikationsentwicklung und -hosting einen gravierenden Einschnitt in die Prozess- und Arbeitsabläufe eines Unternehmens bedeutet.

Gleichzeitig behält die Organisation die Verantwortung für die Business Prozesse. Damit wandelt sich die Aufgabe der verbleibenden IT vom technischen Management zum Anforderungsmanagement. Ein Teil des Entwicklungsprozesses wird zur „Black Box“, die fehlende Kenntnis von Kosten (Projektaufwand) und Umfang von Aktivitäten (Test) führt zu Kontrollverlust. Qualitätsaspekte, wie Änderungshäufigkeit der Spezifikationen, Anzahl der Fehler im Test, Testqualität, Qualität der Applikationen in Produktion, können nicht mehr beantwortet werden.
Negative Strategien, z.B. Androhung einer neuen Ausschreibung oder sukzessiver Aufbau einer Schatten-IT (schleichendes Insourcing) sollten vermieden werden. Der Lösungsansatz besteht darin, Metriken (z.B. über Function Points nach IFPUG 4.2, NESMA 2.2, COSMIC FFP) für eine objektive Kontrolle über den gesamten Lifecycle - von der Spezifikation bis zum Betrieb - aufzubauen:

- Planung & Entwicklung - Function Points (FP)
- Abnahmetests - Test Function Points
- Wartung - Enhancement Function Points
- Betrieb - Function Points & Run time Function Points

Damit gelingt es, die Kontrolle durch Metriken wieder zurück zu gewinnen, z.B.

- Preis pro FP bei Änderungen
- Produktivität pro FP
- Fehler pro FP usw.

und Qualitätsaussagen z.B. zu folgenden Themen zu treffen:

- Anforderungs- und Design Qualität
  - Anzahl der geänderten FP gegenüber den Projekt FP
- Qualität der SW-Lieferungen
  - Aufwand pro FP und Anzahl der gefundenen Fehler pro FP
- Testqualität
  - Testaufwand pro FP (Test-FP versus Projekt-FP)
  - Verhältnis von gefundenen Fehler pro FP in Test und Produktion
- Qualität der Applikationen in der Produktion
  - Aufwand und Fehlerhäufigkeit pro FP

Lifecycle Quality Management - Mehr als Testen, Frank Lohmar (Borland GmbH, Langen/Deutschland)

Ausgehend vom Faktum, dass Software-Entwicklungsprozesse häufig eine „Black-Box“ sind und Themen wie Requirements Elicitation, Erarbeitung von Use Cases, Systemarchitektur Prototyping, Unit/Component Integration oft nur stiefmütterlich behandelt werden, weist der Vortragende darauf hin, dass gerade die Kosten der Fehlerbehebung in frühen Phasen sehr gering sind (z.B. Requirements Phase Faktor 1, in der Produktion aber 40 - 120 mal höher als in der Requirements Phase).

Das Borland Application-Lifecycle-Management beinhaltet ein Prozess Lifecycle Quality Management (LQM), der sicherstellen soll, dass bereits bei den Anforderungen hohe Qualität durch frühzeitiges Beheben von Fehlern erreicht wird.
Der LQM gliedert sich in die Teilprozesse Plan, Verfiy & Validate und Improve sowie einen unterstützenden Teilprozess Manage, der durch Borland-Tools, z.B. Tempo, Caliber, SilkCentral, SilkTest, SilkPerformer, etc. unterstützt wird.

Wie lösen Sie das Chaos Ihrer IT-Anforderungen? Christian Weber (Compuware Austria GmbH, Linz)

Dieser Vortrag behandelt die Notwendigkeit, Anforderungen systematisch und strukturiert zu

erheben und zu dokumentieren, um das Risiko, mit einem Projekt zu scheitern, zu minimieren.

Auf unterschiedlichen Ebenen - Business View (Project Driver), Faculty View (Rough Use Cases), Development View (Use Cases) und QA View (Test Cases) - wird empfohlen, Reviewzyklen von Draftversionen einzuführen und die Ergebnisse jeweils nach der Überarbeitung explizit freizugeben.

Die technische Unterstützung des Prozesses wird anhand des Tools OptimalTrace dargestellt.

ISTQB Certified Tester, ein weiterer Tropfen im Meer der Zertifikate oder doch mehr? Helmut Pichler (International Software Testing Qualifications Board (ISTQB) / Austrian Testing Board (ATB))

Das ISTQB, dessen Mitglied das ATB ist, definiert weltweit gültige Ausbildungsstandards für das Berufsbild des Testers. Derzeit werden zwei Level angeboten - Foundation Level, die Basisausbildung und Advanced Level für erfahrene Tester. Ein Expert Level ist in Ausarbeitung.

Der Lehrplan des Foundation Level hat folgende Ziele:

- Verständnis der gängigen Testpraxis und Prinzipien erfolgreichen Software Testens
- Anwenden grundlegender Techniken zur Planung von Tests
- Aufbau einer Grundlage für die weitere Karriere als Software Tester

und behandelt folgende Themen:

- Grundlagen
- Testen im Softwarelebenszyklus
- Statischer Test
- Testfallentwurfsverfahren
- Testmanagement
- Testwerkzeuge

Um die Prüfung zu bestehen, müssen 60% (65% ab 2008) der Fragen korrekt beantwortet werden.

Der Advanced Level setzt sich aus drei Ausbildungsblöcken zusammen:

- Test Manager
  - Testprozess, Testmanagement, Risikobasiertes Testen, Problemmanagement, Testprozessverbesserung, Testteams
- Test Analyst (dzt. Functional Tester)
  - Systematische & nichtsystematische Testtechniken (z.B. Äquivalenzklassen, Grenzwertanalyse, …), Reviewtechniken, Auswahl von Test- und Prüftechniken
- Technical Test Analyst (dzt. Technical Tester)
  - Nichtfunktionale Testtechniken (z.B. Zuverlässigkeit, Performance,..), Auswahl von Testtechniken, Toolunterstützung

Das ATB hat bisher fast 600 Foundation Level und 170 Advanced Level Zertifizierungen durchgeführt.
Detaillierte Informationen findet man auf der Homepage www.austriantestingboard.at.

Testmanagement beim Redevelopment von Anwendungssystemen – ein Erfahrungsbericht aus einem Großprojekt am Beispiel SCHUFA: „Diener zweier Herren“, Brunhilde Steffens (SCHUFA Holding AG, Wiesbaden), Jens Borchers (Steria Mummert Consulting AG, Hamburg)

Die Schufa verwaltet in Deutschland den größten Datenpool zur Beurteilung des aktuellen Zahlungsverhaltens natürlicher Personen. Die Daten kommen von Vertragspartnern, aus Schuldner-, Insolvenzverzeichnissen und Anschriftenänderungen. Die Auskünfte gehen an Banken, Versandhandel, Telcos, usw. aber nicht an „Jeden“.

Eine seit 35 Jahren bestehende Mainframe-Anwendung, fast 100% in /370-Assembler programmiert, hoch performant und stabil, mit hunderten externen Rechner-Anbindungen und tausenden Service-Nehmern, musste abgelöst werden. Man entschied sich, kein technisches Reengineering, sondern umfassendes Redevelopment durchzuführen und die Host-Anwendung schrittweise durch eine neue J2EE-Anwendung während einer mehrjährigen Übergangszeit abzulösen.

Die Teststrategie ist geprägt vom Spannungsfeld Regressionstest gegen die bestehende Funktionalität versus Neuentwicklungstest auf Basis der neuen Spezifikationen, wobei die Konfliktlösung durch die fachlichen Tester erfolgt, die als Entscheider unverzichtbar sind.

Folgendes Testvorgehen wurde gewählt

- Erstellen fachlicher Testfälle auf Basis der neuen Spezifikation
- Nutzung fachlicher Testfälle zur Ermittlung von Referenzergebnissen (Sollergebnisse) mit dem bestehenden Mainframe-System
- Ausführen aller Testfälle für jedes Release
- Weitgehend automatisierter Vergleich mit den die Sollergebnissen und abschließende Bewertung durch das fachliche Abnahmeteam

Die wesentlichen Schritte der Testdurchführung sind

- formalisierte fachliche Beschreibung in Excel-Files und automatisierte Erstellung der technischen Eingabeformate
- nur sehr wenige manuelle GUI-Tests
- Durchführung aller technischen Aufgaben in einer externen „Testfactory“
- Aufbereitung der technischen Ergebnisse zur endgültige Bewertung und Freigabe durch dass Fachtestteam

Als kritische Erfolgsfaktoren werden Konfigurationsmanagement, die Verfügbarkeit fachlicher Tester und Pessimismus (alle Fehler treten irgend wann doch wieder auf) und die Automatisierung gesehen. Letztere war und ist besonders entscheidend, da aus insgesamt 5.000 fachlichen Testfällen ca. 50.000 technische Testfälle erstellt und durchgeführt werden müssen, um für jedes Release ausreichend Sicherheit zu gewinnen.

Den Graben überbrücken – Aufgabenteilung zwischen QS und Entwicklung unter Einsatz der UML für eine bessere Software, Dietmar Kropfitsch (IT Consultant, Wien), Uwe Vigenschow (oose Innovative Informatik GmbH, Hamburg)

Der Vortrag behandelt die frühzeitige Durchführung von QS-Maßnahmen durch die Entwickler im iterativen Entwicklungsprozess mit Hilfe der UML (Anforderungsdokumentation, Verfeinerung der Aktivitätsdiagramme nach Architekturaspekten, technischer Objektfluss und Ableitung technischer Klassendiagramme, Partitionen für die Architekturschichten, Ableitung von Sequenzdiagrammen usw.) und der QS (Ableitung der Testfälle aus den Aktivitätsdiagrammen, Priorisierung und Betrachtung der Überdeckungskriterien nach den Zweigen der Aktivitätsdiagramme). Um Projekte erfolgreich zu gestalten, müssen die Aktivitäten durch verschiedene Kommunikationsinstrumente - Prozesse, Organisationsformen, technische Instrumente – unterstützt werden. Im Detail werden die Prozesse Änderungs- und Fehlermanagement mit ihrem jeweiligen Fokus gegenübergestellt sowie der Wert des Change Control Boards und der effiziente Einsatz von Fehlerklassen erläutert.

Keynote 2 - Mechanismen zur Entdeckung von Betrugsabsichten in Softwaresystemen, Roland Mittermeir (Software Engineering and Soft Computing, Institut für Informatik-Systeme, Alpen-Adria Universität Klagenfurt)

Da durch ubiquitäre IT und ubiquitäre Software der Spruch „Gelegenheit macht Diebe!“ an Bedeutung gewinnt, Sarbanes-Oxley Act (SOX) und Basel II sowie sonstige gesetzliche Regelungen Impulsgeber sind, macht es Sinn, sich mit diesem Thema auseinanderzusetzen.

Qualitätssicherung prüft, ob Software gestellten Anforderungen entspricht. Nicht geprüft wird, ob Funktionalität enthalten ist, die NICHT spezifiziert wurde. Außerdem ist es nicht einfach, bei einer Abweichung festzustellen, ob es sich nur um einen Fehler handelt oder ob schon Betrugsabsicht vorliegt. Man muss unterscheiden zwischen

- manipulierten Features
  - Code, der bewusst in die Produktionsversion von Software integriert wurde aber nicht der Spezifikation des Systems entspricht
  - Betrugs-Feature (fraudulent code)
    - Manipuliertes Feature, das in betrügerischer Absicht eingefügt wurde

Es wird die Annahme getroffen, dass Featuremanipulation auf die Datenebene wirken muss, um wirksam zu werden und daher (mindestens) eine Schnittstelle einer Softwarekomponente involviert, die nicht manipuliert sein muss. Folgende Fälle können unterschieden werden:

- Modifikation der Schnittstellenbreite
  - Erweiterung => Intrusion, Spionage
  - Reduktion => Maskierung, Ausschalten einer Barriere
  - Abänderung => beliebig
- Erweiterung eines Schnittstellen-Typs
  - unzulässiger Eingabewert => führt in einen vorbereiteten Programmteil
  - unzulässiger Ausgabewert => Fälschung, interpretierbares Signal
- rein algorithmische Modifikation
  - scheinbar korrekter Wert innerhalb der Schnittstellen-Spezifikation

Zur Prüfung von Betrugsfreiheit genügen die Ansätze der Qualitätssicherung Review und Test nicht, sie müssen um Assertionen und Beweise oder Zertifizierungen erweitert werden.
Die statische Analyse (Reviews) könnte durch Checklisten und Werkzeuge unterstützt werden. Der Test wird in der Regel nur Zufallstreffer liefern.

Eine Methode zur Erkennung schnittstellenrelevanter Manipulationen sind Transfer-Slices:

- Forward-Slice ist der minimale ausführbare Teil des vorliegenden Programms, der die im Slicing-Kriterium genannte Variable beeinflusst.
- Backward-Slice ist der minimale ausführbare Teil des vorliegenden Programms, der von der im Slicing-Kriterium genannten Variablen beeinflusst wird

Verallgemeinert kann dieser Ansatz auch so formuliert werden:

- Kriteriumsvariablen sind all jene Variablen, die im zugehörigen Use-Case als Eingaben genannt sind
  - Berechne Forward-Slice (FS)
- Kriteriumsvariablen sind all jene Variablen, die im zugehörigen Use-Case als Ausgaben genannt sind
  - Berechne Backward-Slice (BS)
- Transfer-Slice
  - TS = FS ∩ BS

Folgende Fälle werden unterschieden:

- Sei (FS ∪ BS) – TS = Ø ∧ Code – $\cup_{alleUC}$ TS = Ø, der Code erweitert den Use-Case nicht
- Sei Code – $\cup_{alleUC}$ TS ≠ Ø, der Code enthält Features, die keinen Use-Case abdecken, z.B. diagnostische Features (Perf.Monitoring, „Test-Channel“) oder manipulative Features
- Sei X = (FS ∪ BS) – TS ≠ Ø, der Code erweitert den Use-Case um
  - Features, die anderen Use-Cases zuzurechnen sind
  - Lücke(n) in der Spezifikation
  - manipulative Features.

Insgesamt steht die Fraud-detection in Software noch am Anfang.

SOFTWARETEST UND QUALITÄTSSICHERUNG BEI AXA DEUTSCHLAND, STEFAN ELFGEN, RALF SCHAFFRATH (AXA DEUTSCHLAND, KÖLN)

Softwaretest und Qualitätssicherung der AXA Deutschland sind in zwei Bereichen angesiedelt. Die operative Tätigkeit (Testvorbereitung/–durchführung) ist Aufgabe im Delivery Management, die Verantwortung für Testprozess, Entwicklung der Testmethodik, Unterstützung der operativen Testdurchführung (Beratung), Qualitätssicherung der Testdurchführung (Metriken, Fortschrittsberichte) und Know-How-Transfer obliegt der Abteilung ISAM-TQM im Application Management.

Die besonderen organisatorischen Herausforderungen liegen darin, kompetente und motivierte Test- und QS-Mitarbeiter zu gewinnen sowie Fachwissen, Prozessorientierung und Test-Know-How zu verbinden. Dies wird durch gezielte interne Schulungen, individuelle Einarbeitungspläne und den Aufbau eines Glossars (am „Certified Tester“ orientiert) erreicht.

Der Aufbau einer Testkultur, in der Tester, Requirements Engineer und Entwickler gleichberechtigt sind, soll sich durch die frühzeitige Einbindung der Tester im Projekt und die Etablierung des Berufsbilds für Tester (Zertifizierung) entwickeln. Weiters muß das Verantwortungsgefühl des Einzelnen für die Qualität seiner Arbeitsergebnisse gesteigert werden.

Dem beinahe „State-of-the-art“ Problem unzureichender Dokumentation kann mit dem Hebel SOX, das zwingend Dokumentation erfordert, und dem Ansatz je nach Projekttyp „skalierbarer“ Dokumentation begegnet werden.

Für die Zukunft strebt man eine engere Zusammenarbeit zwischen Requirements Management und Test an, um die Übersetzungsleistung vom Fachkonzept zum Testkonzept zu reduzieren. Weiters sollen in Großprojekten Testmanager mit Projektleiterformat etabliert, das Berufsbild des Testers gestärkt, Softwaretest/QS-Maßnahmen in den Projektplänen stärker berücksichtigt, aussagefähige Metriken, risikobasiertes Testen anhand von Qualitätskriterien, Entry-/Exit-Kriterien für Entwickler-/Fachtests sowie automatisierte Regressionstests eingeführt werden.

TESTMANAGEMENT IM PROGRAMM – TESTEN VON PROGRAMMEN, DIE KEINE MEHR SIND, STEFAN ZEDER (INFOGEM AG, BADEN/SCHWEIZ)

Dieser Vortrag geht auf den Test in Programmen, d.h. Projekten, die voneinander abhängig sind, ein.

Die wesentlichen Erfolgsfaktoren sind

- Der Führungsansatz „Duldung“ funktioniert nicht, einer muss die Führung übernehmen bzw. zumindest abstimmen und koordinieren
- Es reicht nicht wenn jedes Projekt seinen Job erledigt, selbst wenn der Test die eigenen Schnittstellen beinhaltet!
- Die Aufgaben des Programm-Testmanagements beinhalten
  - Identifikation und Testen der Schnittstellen unter den Projekten und zu Umsystemen
  - Prüfen der gemeinsam genutzten Komponenten und Architekturteile
  - Prüfen der Einhaltung von Standards
  - Kundenerlebnis erreichen durch
    - End-to-End Testing
    - Durchlaufzeiten/Transaktionsraten
- Abgrenzung zum Rollout Management

Zwei Organisationsformen für das Testmanagement sind erfolgsversprechend:

- Testmanager Programm und direkt den Projekten zugeordnete Testmanager

- Testmanager Programm, dem Testmanager für Integration, System und Acceptance unterstellt sind

Wesentlich ist, dass der Testmanager Programm in beiden Fällen Gesamtverantwortung und Koordination beibehält.

Die Teststrategie orientiert sich an der Architektur und wird über die Pfade im System bestimmt.

Beim Testkonzept sollte man folgende Schwerpunkte setzen:

- Die Business-Übersicht, die auf Programmebene häufig fehlt, ist für die Konzeption der Tests unumgänglich
- Erarbeiten der Grenzen des Programms und der Schnittstellenbeschreibungen intern/extern
- Analyse der Funktionen & Services, Identifikation der Testfälle
- Bestimmung eines Defect Tracking Tools
- Testplanung mit Hilfe von Meilensteinen

„Gut ist manchmal gut genug – Software-Entwicklung der SEZ AG („the single wafer advantage“) im Spannungsfeld zwischen Qualität und Flexibilität, Heinz Pozewaunig (SEZ AG, Villach)

Die Softwareentwicklung der SEZ muss folgende Herausforderungen bewältigen:

- 4 Betriebssysteme, MS-DOS – Windows XP
- viele Produkte, für jedes Tool ca. 22 Hauptentwicklungslinien
- spezielle Kundenerweiterungen
- sehr große Software, mehr als 800K+ LOC
- komplexe Konfigurationen
- complexes Aufgabengebiet – Chemische Prozesse, HW, Firmware SW, real time, safety, …
- Dynamischer Markt

Die Softwareentwicklungsabteilung (ca. 60 Mitarbeiter) ist Teil der Systementwicklung (ca. 120 Mitarbeiter). Der SW-Entwicklungsprozess ist an den RUP angelehnt, besonderer Wert wird darauf gelegt, dass QA/Test frühzeitig in den Requirements Engineering Prozess eingebunden ist.

SEZ nahm am GPARD Im Jahr 2004 teil, ließ die Prozesse SW Development und SW Requirements Engineering nach ISO 15504 assessieren und verpasste den Capability Level 4 (predictable) nur knapp. Nach dem Assessment wurde die Situation analysiert und entschieden, den Level 4 nicht anzustreben. Dies wird damit begründet, dass die SW Defects im Feld vor allem für die neuen Produktlinien hoch sind, bedingt durch die frühzeitige, teilweise zu wenig ausgereifte Markteinführung. Man strebt daher in erster Linie eine Stabilisierung der Situation an.

Für die Zukunft sieht man Verbesserungspotential vor allem im Bereich System Development (Systemanforderungen, teilweise selbst der Kunden unbekannte Anforderungen) und Software Engineering (Verbesserung des Requirements Management, Know-How-Aufbau von QA/Test).

Warum sind gute Daten eigentlich schlechte Daten? – Über die Qualitätsproblematik von guten Daten in operativen Systemen und schlechten Daten in Data Warehouse-Systemen am Beispiel des DWH an der Universität Wien, Harald Riedel-Taschner (Zentraler Informatikdienst der Universität Wien)

Die Implementierung des Universitätsgesetzes 2002 erfordert den Aufbau eines Data-Warehouse (DWH), als Datenlager für

- die Erstellung der Wissensbilanz für das Wissenschaftsministeriums,
- nachhaltiges Berichtswesen für Managemententscheidungen und Zielvereinbarungen des Rektors
- Transparenz der Berichte
- ad-hoc-Abfragen der Fachabteilungen

Dies bedingte den Aufbau eines Data-Warehouse innerhalb von eineinhalb Jahren. Ziel war es, dass jeder „Dateninhaber“ (jede Fachabteilung) seine Daten auch wieder selbst abfragen kann, ohne dafür Techniker beiziehen zu müssen. Alle operativen Daten liegen im Verantwortungsbereich derjenigen Abteilung, die gemäß Organisationsplan dafür verantwortlich ist. Die Verantwortung beinhaltet Erfassung, Korrektur und Auswertung.

Im Zuge des Aufbaus des DWH zeigten sich strukturelle Probleme der Daten aus dem Primärsystem. Diese Daten waren zwar für das Primärsystem ausreichend genau, bei der Auswertung im DWH stellte man aber fest, dass die Datenqualität für das DWH unzureichend war. Folgende Beispiele wurden genannt:

- Manche Attribute wurden zu wenig genau vergeben, z.B. für neue Studierende die Kategorisierungen Erstzulassung, Neuzulassung, Wiederzulassung“
- Das Spannungsverhältnis Fachabteilung – Berichtersteller – Software anhand der Fragestellungen interessierte bzw. zugelassene (inskribierte) Studien versus alle theoretisch möglichen Kombinationen

- Arbeitsverhältnisse werden im Primärsystem als einzelne Datensätze verwaltet, im DWH will man sie allerdings wieder zusammenfassen.

Diese Probleme werden durch Erkennen, Identifizieren und Analysieren der Datenprobleme aus DWH-Sicht, erhöhte Aufmerksamkeit der operativ Tätigen bei Pflege und Auswertung und durch die Etablierung eines Datenqualitätsmanagement-Projektes zur nachhaltigen Bereinigung der Daten entsprechend ihrer Gewichtung schrittweise minimiert.

QS und Test in der Automotive- und Versicherungsbranche – ein Vergleich Johannes Kreiner (Generali Österreich, Wien), Michael Neubacher (Porsche Holding GmbH, Salzburg)

Die Testmanager stellen ihre jeweiligen Organisationen und Prozesse gegenüber:

- Organisation
  - Generali - Testzentrum (ca. 35 MA)
  - Porsche - zentrales SW-Testteam (ca. 4 MA)
- Funktionen im Unternehmen
  - In beiden Unternehmen sind Testprojektleitung, Testfallfindung, -durchführung, -dokumentation, -verwaltung, Testumgebungsmanagement und Problemverfolgung ausgeprägt
  - Testabnahme und Testevaluierung bei Porsche ausgeprägt, bei Generali als interner Dienstleister nicht vorhanden
- Einbindung des Testprozesses in die Softwareentwicklung
  - In beiden Unternehmen frühe Einbindung
- Reporting
  - Generali - Fehlerprognosemodelle und Ampelstatus für jede Testversion, Fehlerkurve mit Interpretation
  - Porsche - Messung der Testeffizienz mit der Defect Detection Percentage, Fehlerzahlen aus dem Echtbetrieb
- Anforderungsmanagement
  - Generali – wird vom Test getrieben
  - Porsche – wird von der Entwicklung getrieben
- Testumgebung
  - Generali – Entwicklungs-, TZIT autonome Integrationstestumgebung, VIS Integrationstestumgebung System- und Betreibbarkeitstest, Maintenanceumgebung für Hot Fixes
  - Porsche – Entwicklungstest-, QS-Test- und Produktionsumgebung

Nutzungsqualität im Ubiquitous Computing – Plädoyer für eine föderalistische Betrachtung von Gebrauchstauglichkeit, Martin Hitz, Rudolf Melcher (Interactive Systems, Institut für Informatik-Systeme, Alpen-Adria Universität Klagenfurt)

In den allgegenwärtigen Rechnern sind die Benutzer mit inkonsistenten Strukturen konfrontiert, was zu kognitiver Überlastung und ungenutzten Potentialen führt. Daher ist es notwendig, eine neue Architektur zu entwickeln. Diese Aussage wird mit einfachen Beispielen untermauert:

- Synchronisation Mobiler Geräte
- Mehrdimensionale Klassifikation
  - Eigene Bilder, Eigene Videos, Eigene …
  - Projekt A, Projekt B,
- E-Mail vs. Dateiverwaltung (CMS) vs. Projektmanagement

Damit werden die Computer aber als universelles Werkzeug in Frage gestellt. Ein klares Schichtenmodell zur Artefaktverwaltung ist gefordert, das basieren soll auf

- Selbstorganisierenden Systemen
  - Kopplungen ohne Konfigurationsaufwand (siehe USB-Stick)
- Virtuellen Dateisystemen (VFS)
  - konsequent transparenter Zugriff auf netzgebundene und lokale Artefaktpools
- Concept/Topic Maps
  - Implementierung einer Schicht zwischen physischem Gerät (HW/OS) und „Denkgebäude“ (-strukturen) der Benutzer
  - Flexibilität ähnlich den Fenster Managern bei Unix basierten Systemen
- Action Spaces
  - Bieten den entsprechenden Kontext zu den Strukturen