

Aspekte der Qualitätssicherung bei der Entwicklung eines Komponenten-basierten Systems

Dr. Josef Withalm

Siemens AG Österreich, Programm –und Systementwicklung
josef.withalm@siemens.com

Abstrakt. Dieser Vortrag thematisiert Aspekte des QM bei der Entwicklung eines Komponenten-basierten Systems. Anhand eines Beispiels aus dem Tourismusbereich würde ich gern die Einflüsse von Komponententechnologien auf QM-Systeme darlegen. Ich will aufzeigen, welche Anforderungen e-Business stellt und dass diese nur durch die Nutzung von Komponententechnologien erfüllt werden können. Andererseits scheinen diese Anforderungen mit den QM-Prozessen auch nicht kompatibel zu sein. In meinem Beitrag lege ich den Aufbau einer e-Business Architektur und die einsetzbaren Komponententechnologien vor, um sowohl dem QM als auch dem e-Business zu entsprechen. In weiterer Folge werde ich erklären, wie diese Aufgaben in dem dargelegten Projekt gehandhabt werden.

1 Einführung

Das Kontingentproblem stellt nach wie vor eine der größten Hürden bei der Einführung von Reservierungssystemen im Tourismus und insbesondere in Klein – und mittelständischen touristischen Unternehmen (KMTUs) dar.

Je kleiner das Unternehmen ist, umso größere Vorbehalte existieren gegenüber Kontingenten, einerseits weil der Reiseveranstalter hohe Prämien verrechnet, nämlich zwischen 20% und 60% des veranschlagten Zimmerpreises, andererseits weil dem Leistungsträger (Hotelbetreiber) nicht gewährleistet ist, seine Zimmer tatsächlich auch zu vergeben. Um sich der Vertriebskanäle beliebiger Reiseveranstalter bedienen zu können, sind unterschiedliche Kontingente erforderlich.

1.1 Nachteile des bestehenden Systems:

- Echtzeit-Informationen über vorliegende Kontingente sind nicht erhältlich, weil Daten manuell eingegeben werden
- Für den Franchisenehmer erwachsen aus dem System hohe Vermarktungskosten. Sein Tätigkeitsbereich umfasst folgende Aufgaben:
 - Vertragsabschluß mit Leistungsträgern über Kontingente
 - Katalogisierung von Produkten und Dienstleistungen

- Eintragung von Produkten und Dienstleistungen in das elektronische System
- Vermarktung von Produkten und Dienstleistungen über Reisebüros, Call Centern und im Internet
- - Hohe Prämien

Daher sind Alternativlösungen zu Kontingenten dringend erforderlich.

1.2 Unsere Lösung

Da wir gemeinsam mit der Firma Start-Amadeus das *euroSTART*[®] System entwickelt hatten (siehe „New Business Strategies Stress QA“, J.Withalm, 2000), entschlossen wir uns einen Prototyp zu bauen, der mit Hilfe der Agententechnologie das o.a. beschriebene Kontingentproblem lösen sollte. Diesen Prototyp konnten wir auf mehreren Konferenzen (Online 2000, Enter 2000) vorstellen, und er rief internationales Interesse hervor. Insbesondere internationale Hotelgruppen und die Interessensvertretungen der Hotels (z.B. ÖHV) waren sehr an dieser Technologie interessiert. Um nun das Produkt (SAPA: Solving Allotment Problems by Agents) auf der Basis dieses Prototyps zu entwickeln sollte es möglich sein, beliebige Buchungssysteme (sowohl Web-basierte als auch proprietäre C/S-Systeme) mit beliebigen Property Management-Systemen (PMS), die vor Ort in Hotels zur Verwaltung der Hotelressourcen eingesetzt werden, zu verbinden. Diese Entwicklung musste sich zwangsläufig des komponentenorientierten Entwicklungsparadigmas bedienen. Im folgenden wollen wir vor allem die methodische Vorgangsweise, die Basistechnologien und die Auswirkungen auf die QS beschreiben.

2 Methodische Vorgangsweise bei der Implementierung von e-Business Lösungen

Eine der grundlegenden Anforderungen an das QM ist es, sich stärker auf die schnelllebigen Ansprüche von e-Business anzupassen. Da es sich bei unserem Projekt um eine typische e-Business Anwendungen handelt, mussten auch wir die Vorgangsweise in der Entwicklung darauf ausrichten.

Unter e-Business verstehen wir die wichtigsten über das Internet angebotenen Geschäftsabläufe einer Organisation. Die Implementierung eines solchen Systems bringt eine Reihe von Herausforderungen mit sich, auf deren Lösungsmöglichkeiten im folgenden eingegangen werden soll.

Einige der herausragendsten Vorteile eines architekturorientierten Ansatzes für e-Business Anwendungen sind wie folgt:

- Schnelleres Time-to-Market
- Besserer Schutz der Investitionen gegen Technologieveränderungen

- Möglichkeit zur Skalierung mit wachsendem Bedarf
- Erreichen von Robustheit und Kontinuität der Lösung bei gleichzeitig minimierter Störung der bestehenden Architektur
- Minimieren der Personal und Wartungskosten
- Bereitstellen einer Architektur, die sich geschäftlichen Veränderungen anpasst
- Unternehmen ist in der Lage, Technologien der Vergangenheit, Gegenwart und der Zukunft effizient einzusetzen
- Sicherstellung einer Grundlage für wohlinformierte Entscheidungen zwischen Eigenentwicklung und Zukauf
- Unabhängigkeit von umfangreicher Kodierung und mehrfachen Middleware-Services

2.1 Schaffung einer e-Business Architektur

Um eine e-Business Architektur zu erzeugen, sind folgende Schritte erforderlich:

- Identifizierung von Schlüsselerfordernissen
- Auswahl eines Architekturmodells
- Synthese der High-Level Architektur
- Strukturierung der High-Level Architektur
- Überprüfung der Technologieerfordernisse
- Auswahl der Technologie

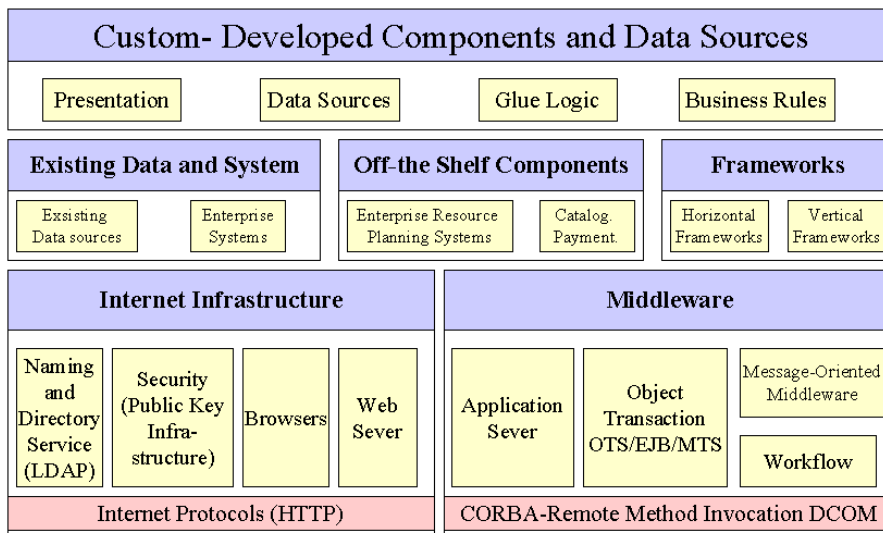


Tabelle 1. Für die Festlegung einer detaillierten Architektur zu evaluierende Schlüsseltechnologien

2.2 Validierung der e-Business Architektur

Eine e-Business Architektur unterliegt einer Validierung. Folgendes ist dazu erforderlich:

- Architektur-Review
- Prototyp
- Pilotanwendung

2.2.1 Architektur-Review

Diese sollte von einem kleinen Team von Beteiligten aus allen Bereichen durchgeführt werden. Ziel ist es, eventuelle Problembereiche ausfindig zu machen. Das Ergebnis dient als Basis für den Prototyp und die Pilotanwendung.

2.2.2 Prototyp

Unter Prototypisierung verstehen wir die Unterziehung eines Teils einer Architektur oder Technologie einer Analyse hinsichtlich dessen Funktionalität. Das vorrangige Ziel liegt in der Erstellung von Lösungen, die auch auf technologische Anforderungen und Constraints sowie auf Budgeteinschränkungen Bedacht nehmen.

2.2.3 Pilotanwendung

Im Gegensatz zum Prototyp, der die Funktionsfähigkeit einer speziellen Technologie validiert, unterzieht eine Pilotanwendung die gesamte Geschäftslösung einer Validierung. Darüber hinaus liegt das Ziel einer Pilotanwendung im Erfahrungsgewinn.

2.3 Implementierung einer E-business Architektur

Bei der Implementierung einer e-Business Architektur ist insbesondere auf folgendes zu achten:

- Design -und Entwicklungsumgebung
- Applikationsdesign
- durchgängiger Entwicklungsprozess
- Designmuster
- Erstellung und Wiederverwendung von Geschäftskomponenten
- Einhaltung von Standards
- Skalierbarkeit und Performance

2.3.1 Erweiterung einer bestehenden Architektur in die e-Business Anwendung

Oft ist es erforderlich, eine bestehende Anwendung in die e-Business Architektur zu integrieren. Dabei müssen folgende Aspekte beachtet werden:

- Erweiterung bestehender Modelle unter Verwendung von Wrapper Codes
- Erweiterung bestehender Modelle unter Verwendung von Gateways

- Teilweise Ersetzung bestehender Geschäftsmodelle
- Skalierbarkeit und Performance

Die erfolgreiche Erweiterung einer bestehenden Anwendung in die e-Business Architektur wird großteils dadurch bedingt, wie gut deren Design ist.

- Technologische Constraints
- Anwendungen in Sprachen der dritten Generation oder in proprietären Systemen sind ohne kompliziert gebaute Brücken nicht notwendigerweise in die neu konzipierte Architektur integrierbar.

2.4 Inbetriebnahme der e-Business Architektur

Eine besondere Herausforderung bei e-Business Lösungen liegt in der tatsächlichen Inbetriebnahme der Architektur. Man ist dabei mit folgenden Aspekten konfrontiert:

- Change Management
- Software –und Datenmigration
- Schulung

2.4.1 Produktionsmanagement

Nach der Inbetriebnahme geht das System in die Produktion über, wobei hier folgendes beachtet werden muss:

- Neue Rollen und Verantwortlichkeiten
- Management-Tools
- Troubleshooting

3 Technologische Anforderungen und Constraints

Unser System macht sich die Möglichkeiten von Agenten zu eigen, wobei für uns insbesondere die Eigenschaft, autonom vor Ort zu agieren bzw. auf Grund einer standardisierten Sprache Verhandlungen zu führen, bedeutend ist. Technisch gesehen handelt es sich bei Agenten um Komponenten, wobei Java auf Grund seiner Eigenschaften hinsichtlich Portabilität und Sicherheit bevorzugt zur Implementierung von Agenten herangezogen wird.

Corba wurde bei uns vor allem deswegen eingesetzt, weil unsere Agenten mit bestehenden SW-Systemen wie z.B. beliebigen Internet-Buchungssystemen und mit Property Management Systemen in den Hotels abwechselnd arbeiten müssen.

EJB soll sowohl auf den Servern, auf welchen die Internet-Buchungssysteme laufen, als auch auf dem Server auf welchen das PMS eingesetzt wird, zur Verwendung kommen. Die Argumente, warum wir auf EJB zurückgreifen sind folgende:

- Wird auf vielen Anwendungsservern zur Verfügung gestellt
- Ist kompatibel mit Corba

- Transaktionsverhalten und Sicherheitsverhalten sind einfacher zu implementieren als in Corba
- Beide Verhalten können bei der Inbetriebnahme in Form von Eigenschaftsdialogen gesetzt werden.

3.1 CORBA

Dieser Standard wurde von der Open Management Group (OMG) entwickelt. Corba war erforderlich, weil Bedarf bestand, verschiedene in unterschiedlichen Programmiersprachen (C, C++, Java, Cobol,...) angelegte und auf unterschiedlichen Betriebssystemen laufende Objekte interagieren zu lassen.

Corba 2.0 ermöglicht die Zusammenarbeit von auf unterschiedlichen ORBs registrierten Objekten über das Internet. Dabei wird IIOP (Internet inter-ORB protocol) eingesetzt. Das bedeutet, dass ein auf einem ORB registriertes Java-Applet (ein Java-Objekt) von jedem anderen beliebigen Corba-Objekt über das Internet Anfragen stellen kann. Ein wichtiger Aspekt von Corba besteht in der Eingliederung von übernommenen Systemen, wobei diese in modernen e-Business CIS-Anwendungen wieder eingesetzt werden können. Dies erfolgt dadurch, dass per Definition die Oberfläche eines jeden Corba-Objektes sich der Interface Definition Language (IDL) bedient. Alle Attribute, Methoden, Ereignisse und Ausnahmen des Objektes werden in dieser IDL ausführlich dargelegt.

Vorteile von Corba sind wie folgt:

- Schneller Zugriff auf bekannte (IDL-definierte) Objekte
- standardisierte IDL und ORB Infrastruktur
- Corba dient gewöhnlich als Anschlusspunkt für Serveranwendungen

Nachteile von Corba sind wie folgt:

- Keine selbständige Anpassung an modifizierte Datenbanken (besondere Ereignisse, Koordinierungsaufgaben)
- Sehr strenge Zugriffsregelung (Schnellsuche ist nicht möglich)
- Off-Line Arbeit nicht möglich (z.B. Monitoring und Warten auf Modifizierung)
- Keine Lernfähigkeit (Verhalten von Anwendern muss von der Server Software aufgezeichnet werden)

3.2 Agententechnologie

Software-Agent: „eine Rechenentität, welche von Anwendern entwickelte Aufgaben automatisch durchführt“

Vorteile eines Agentensystems sind wie folgt:

- Durch spezielle Dienste führt das Agentensystem im Rahmen des Datentransfers unter den Partnern eine Reihe von Koordinationsaufgaben durch
- Die Implementierung des Agentensystems ermöglicht standardisierte Kommunikation

- Das Agentensystem gewährleistet eine unproblematische Erweiterung
- Das Agentensystem gewährleistet einen hohen Automatisierungsgrad und erzeugt eine einheitliche Interface
- Hotelmanager können ihre Produkte über eigene Hotelagenten auf dem elektronischen Marktplatz anbieten. Auf diese Weise können Sonderangebote sehr leicht und schnell vorgestellt werden.
- Es ist sehr wichtig anzumerken, dass Änderungen weder am Reservierungssystem noch am PMS (Property Management System) erforderlich sind

Diese Vorteile sind für genau definierte Datenbanken und User Interfaces (wie für den Browser Applet für den Prototyp) nicht wesentlich relevant. Hingegen erfordert die zukünftige Anwendung mit anderen User Interfaces (Sprache, Handys, PDAs) und dynamischen heterogenen Datenbanken (z.B. Web-basierte dynamische Datenbanken) die Agententechnologie als Ergänzung zum Corba Rückgrat.

Die multiple Versionsmethode ist ein weiterer berücksichtigungswürdiger Punkt. Heute finden sich überall Inkompatibilitäten (z.B. Dokumente aus unterschiedlichen MS-Office Versionen), wodurch das QM sehr gefährdet wird. Agenten sind sehr dynamisch und können sich schnell Änderungen anpassen. Daher bieten sie einen wesentlichen Vorteil gegenüber den anderen Systemen.

Ein offener globaler Markt erfordert die Anwendung intelligenter Middleware, wobei Corba den ersten Schritt darstellt und eine Verbindung zu der Agententechnologie erzeugt.

3.2.1 Beschreibung der Aufgabe/des Ziels

Aufgrund der autonomen „Sichten und Buchen“ Funktionsweise der Agenten bieten sich Alternativen zu Kontingenten. Darin ist auch das Ziel dieses Projektes begründet.

Daher fordern wir von einem Software-Agenten folgende Attribute:

Delegation	(der Anwender delegiert die Verantwortung an den Agenten)
Kommunikationsfähigkeit	(erweiterte und komplexere zukünftige Aufgaben sind möglich)
Autonomie	(Arbeiten im Hintergrund, Erlaubnis zu verhandeln)
Monitoring	(Fähigkeit, sich in der eigenen Nachbarschaft zu orientieren)
Aktion	(Nachbarschaft wird durch Aktion geändert)
Intelligenz	(Fähigkeit zu interpretieren und Entscheidungen zu treffen)
Mobilität	(von einem Apparat zum anderen Transfers vornehmen)
Sicherheit	(JAVA-implementiert, Sandbox)
Persönlichkeit	(Integration der eigenen Identität)

Folgende Agentenfähigkeiten werden unterschieden:

Aufgabenbezogene Fähigkeiten: typische Aufgaben sind Informationsbeschaffung, Informationsfiltrierung und Coaching

Fachwissen: Es gibt zwei wesentliche Wissenskategorien:

- A-Priori-Wissen: kann entwickler-, anwender-, und systemspezifisch sein
- Lernen: kann dialog- oder fallbezogen sein, oder kann mit Hilfe von Neuronennetzwerken erfolgen.

Kommunikationsfähigkeiten: Auch hier wirken zwei wesentliche Kategorien:

- Anwenderspezifisch: um eine Lernsituation zu schaffen muss der Anwender mit dem Agenten kommunizieren. Dies kann entweder anhand einer Interface, sprachenbasiert (Voice Agents [1]) oder durch soziale Interaktion erfolgen.
- Mit anderen Agenten: Kommunikationssprache zwischen Agenten

3.3 Enterprise Java Beans

Die Architektur Unternehmen Java Beans ist eine Komponentenarchitektur für die Entwicklung und den Einsatz von Komponenten-basierten verteilten Geschäftsanwendungen. Anwendungen, die sich der EJB Architektur bedienen sind skalierbar, transaktionsfähig und können von mehreren Anwendern bedient werden. Solche Anwendungen können einmalig geschrieben und folglich auf jeder beliebigen Serverplattform in Betrieb gesetzt werden, die EJB-Spezifikationen unterstützt.

Etwas vereinfacht könnte man das für unseren Anwendungsfall auch so definieren: EJB ist ein serverseitiges Standardkomponentenmodell für Komponenten-Transaktionsmonitore.

3.3.1 Verteilte Objekte

Verteilte Objekte bilden die Grundlage moderner 3-tier Architekturen. Tier 1 bietet die Darstellungslogik, Tier 2 die Businesslogik, und Tier 3 bietet den Zugriff auf weitere Ressourcen wie z.B. Datenbanken. Verteilte Objekt-Architekturen basieren auf einer Netzwerk-Kommunikationsschicht. Im wesentlichen besteht diese Architektur aus 3 Teilen, dem Objekt-Server, dem Skeleton und dem Stub (Details siehe 3.1. Corba).

3.3.2 Komponentenmodelle

EJB definiert ein serverseitiges Komponentenmodell. Unter Verwendung einer Reihe von Klassen und Interfaces aus den java.ejb-Packages können Entwickler eigene Komponenten erzeugen, zusammensetzen und in Betrieb nehmen, die der EJB-Spezifikation gehorchen. In EJB kann eine Komponente ein Kundengeschäftsobjekt sein, das in jedem EJB-Server verwendet werden kann, um eine Geschäftsanwendung zu entwickeln, in der ein Kundengeschäftsobjekt benötigt wird.

3.3.3 Komponenten-Transaktionsmonitore

Sind Hybride an ORB und TPM. Die grundlegenden Merkmale eines CTM sind verteilte Objekte, eine Infrastruktur, zu der die Transaktionsverwaltung und andere Dienste (z.B. Sicherheit, Persistenz, Namen,...) gehören, sowie ein serverseitiges Komponentenmodell. CTMs stammen aus verschiedenen Entwicklungsansätzen; dazu

zählen der Bereich der relationalen Datenbanken, jener der Anwendungsserver, jener der Web-Server, die Corba-ORB-Industrie und die TP-Monitor Industrie.

3.3.4 Überblick über die Architektur

Um eine Enterprise-Bean zu implementieren, muss man zwei Interfaces und eine oder zwei Klassen definieren.

Remote interface. Definiert die Geschäftsmethoden der Beans, z.B. reservieren, buchen, zahlen,...

Home interface. Definiert die Lebenszyklus-Methoden der Beans (Methoden zum Erzeugen neuer Beans, zum Entfernen und Auffinden von Beans)

Bean Class. Implementiert die Geschäftsmethode; man unterscheidet:

- Entity Beans: modellieren Geschäftsmethoden, die als Substantive ausgedrückt werden können
- Session-Beans: sind nützlich, um die Interaktion zwischen anderen Beans (z.B. Geschäftsprozess) zu beschreiben, oder um bestimmte Aufgaben zu implementieren. Im Gegensatz zu Entity Beans repräsentieren Session Beans keine gemeinsam genutzten Daten in der Datenbank, können auf diese aber zugreifen. Man unterscheidet zwischen zustandslosen und zustandsbehafteten Session Beans.
- Zustandslose Session Beans: verwaltet keinen Konversationszustand in bezug auf das gerade bediente EJB-Objekt. Sie ist weder durchgängig noch einem bestimmten Server zugeordnet.
- Zustandsbehaftete Session Beans: sie sind für die Lebensdauer der Bean Instanz einem einzigen Client zugeordnet; sie agieren im Namen des Clients als dessen Agent. Sie verwalten einen Konversationszustand: d.h. die Instanzvariable der Bean-Klasse kann Client-bezogene Daten zwischen Methodenaufrufen zwischenspeichern.

Priority Key. Stellt einen Zeiger in der Datenbank zur Verfügung. New Entity Beans benötigen einen Prioritätsschlüssel

Zusätzlich sind im EJB noch die Ressourcenverwaltung und die bereits erwähnten Primärdienste spezifiziert.

Weitere Details finden Sie in "Enterprise Java Beans" (O'Really)

4 Implikationen auf die Qualitätssicherung

Die QS ist in diesem Projekt von mehreren Dimensionen abhängig:

- wegen des Business-Modells muss die SW auf verschiedenen Computern in Betrieb gesetzt werden und dort mit bereits existierender Software interagieren.

Eine Produktabnahme impliziert in unserem Fall die Abnahme von 3 verschiedenen Systemen:

- Teil von SAPA auf der Seite des Internet-Buchungssystems
- Abnahme der Agentenplattform, die nicht selbst entwickelt wurde

- Teil von SAPA auf der Seite des PMS

Um eine erfolgreiche Abnahme zu erreichen, musste unser SEM

- für dieses Projekt angepasst
- SEM wurde 1983 geschaffen. Seit dem hat sich jedoch die Software Entwicklung gänzlich geändert. Weitere Details finden Sie in „The long way to CMM – level 4“ (T.Kasse, J.Withalm)
- die Kollegen der Partnerfirmen geschult werden
- ein einheitliches Q-Management festgelegt werden
- Erweiterung von SEM um Aspekte, die aus Chestra kommen und welche ich beim bereits ausführlich erläutert habe (Siehe „New Business Strategies stress QA“, J.Withalm, 2000). Im wesentlichen handelt es sich um eine gemeinsame Vision/Strategie aller beteiligten Partner und eine Unterteilung des Gesamtprojektes in kleinere überschaubare Einheiten, die relativ schnell – allerdings SEM folgend – entwickelt werden, um hier auf allfällige geschäftliche bez. technologische Änderungsanforderungen hinreichend schnell reagieren zu können.
- QM sollte nicht auf Kosten der schnellen Anpassung an Markterfordernisse gehen

Bei e-Business-Applikationen gibt es üblicherweise eine gänzlich andere Priorisierung der Q-Merkmale:

- **Zuverlässigkeit:** das System muss an 365 Tagen 24h zur Verfügung stehen
- **Performance:** kein Internet/Call-Center-Anwender ist bereit, längere Responsezeiten zu akzeptieren. Aufgrund der ISDN und ADSL Technologien sowie den einheitlichen Tarifen in vielen Staaten wird dieses Problem zukünftig gelöst werden können.
- **Funktionserfüllung:** da Buchungen auch Zahlungsverkehr bedingen, muss dieses Q-Merkmal 100% erfüllt sein. Eine besondere Betonung liegt hier auf der Transaktionssicherheit, dem sogenannten ACID-Prinzip (atomicity, consistency, isolation, durability). Diesem Prinzip muss Rechnung getragen werden.
- **Wartung:** auf Grund der extremen Verfügbarkeitsanforderungen müssen entsprechende Vorkehrungen für Austausch von Komponenten getroffen werden
- **Benutzungsfreundlichkeit:** dieses Q-Merkmal tangiert uns bei diesem Projekt nicht: auf der Internet-Seite werden bewährte Web-Seiten unserer Partner verwendet, die sich durch millionenfache Page-Views bewährt haben.
- Auf der Seite des PMS (Property Management System) benötigen wir keine GUI. User Agent ist ein „unsichtbarer Agent“ Er hat quasi die Funktionalität eines unsichtbaren Rezeptionisten.
- **Anpassbarkeit:** ist größtenteils durch Wahl der komponentenorientierten Entwicklung Genüge getan (siehe Punkt 3).

5 Was wir gelernt haben:

Ich möchte diesen Abschnitt folgendermaßen untergliedern:

- A: Projektmanagement insgesamt (Review/Testplanung)
- B: Projektmanagement der einzelnen Teilprojekte (Review/Testplanung)
- C: Erreichung der geforderten Qualität (Qualitätsmerkmale) unter Bedachtnahme der ausgewählten Technologie
- D: Abnahme

Ad **A**: Aufgrund unserer Erfahrung mit der Abwicklung des Eurostart-Projektes wurde nach Vereinbarung mit unseren Partnern festgelegt, dass der Projektleiter aber auch der QSV von uns kommen. Die Projektorganisation sah von vornherein eine Aufgliederung in Teilprojekte vor. Die Teilprojektorganisation möchte ich in B näher erläutern. Die Teilprojekte waren:

- Internet-Buchungssystem/Agenten
- Beschaffung und Abnahme des zugekauften Agentensystems (Aglet/IBM)
- PMS/Agenten

Die Projektleitung war unter anderem dafür zuständig:

- Mit den Managern der Partner eine gemeinsame Vision zu erarbeiten
- Sinnvolle Auslieferungseinheiten gemeinsam mit den Auftraggebern zu erarbeiten z.B. als erste Auslieferungseinheit eine reine Vakanzabfrage zu implementieren um diese befreundeten Kunden (z.B. Mitarbeitern in unserer Firma) zur Verfügung zu stellen

Ad **B**: In jedem Teilprojekt gab es wieder eine Projektorganisation, deren Aufgaben im wesentlichen durch das SEM abgesteckt wurden.

Auf Grund der schlechten Erfahrungen (Konkurrenz zwischen Server und Client-Projektmitarbeitern) beim Projekt Eurostart wurde bei dieser Projektorganisation vermieden, dass sich Konkurrenzverhältnisse zwischen Teilprojekten bilden können. Ein besonderer Schwerpunkt lag auf Testfällen im frühen Projektstadium, um die Funktionalität der eingesetzten Komponenten und Teilprojekte zu gewährleisten.

Ad **C**: Trotz der kritischen Erfahrungen beim Projekt euroSTART insbesondere mit den Qualitätsmerkmal Performance musste auch in diesen Projekt der Großteil des Aufwands für dieses Qualitätsmerkmal aufgewendet werden.

Die Bemühungen begannen bereits bei der Architektur und setzten sich fort, in den bei allen 3 Lösungstechnologien, immer vor allem alle Aspekte des Lastenausgleichs und der Skalierung untersucht wurden.

Während der gesamten Entwicklung fokussierten alle Reviews und Tests/Prototypen auf die Erreichung dieses Qualitätsmerkmals.

Generell lässt sich sagen, dass durch den Einsatz großteils bereits entwickelter und nur mehr anpassbarer Komponenten, die anderen Q-Merkmale relativ leicht erfüllt werden konnten. Die Entwicklungszeit/Aufwand konnte sehr genau erreicht werden und die Fehler waren wesentlich geringer und traten vor allen in den frühen Phasen auf. Die Fehlerbehebung gestaltete sich auch wesentlich einfacher als bei konventionell entwickelten Systemen.

Ad **D**: Die Testfälle für die Abnahme wurden von den Teilprojektleitern bei den Partnerfirmen beauftragt – diese haben von allem das Domain Know-how.

Dadurch, dass wir 3 Teilabnahmen machten, gestaltete sich die Totalabnahme noch immer nicht trivial.

Insbesondere wurde ein eigenes Team beauftragt auf Grund der Testfälle für die Teilabnahmen ein Set für die Gesamtabnahme festzulegen.

Weitere kritische Probleme bereitete uns die Sicherheit. Obgleich üblicherweise Hotels kaum Firewalls eingerichtet haben mussten wir prinzipiell mit dieser Möglichkeit rechnen.

In diesem speziellen Fall konnten wir insbesondere die Möglichkeiten von EJB nutzen, wie man Firewalls gezielt und sinnvoll umgehen kann.

Nachschlagwerke

1. T. Kasse, J. Withalm: The long way to CMM – level 4. In Proceedings of the First World Congress for Software Quality, San Francisco, USA, Juni 1995.
2. J. Withalm: The Impact of New Technologies on Quality Assurance. In Proceedings of the Sixth European conference on Software Quality, Vienna, Austria, April 1999.
3. J. Withalm: Requirement engineering – Most important for improving the software quality. In Proceedings of the Fifth European Conference on Software Quality, Dublin, Ireland, September 1996.
4. S. Sasabe Ryuzo Kaneko: Incorporating technology innovation organization into a new software quality paradigm. In Proceedings of the Sixth European conference on Software Quality, Vienna, Austria, April 1999.
5. Siemens Business Services GmbH & Co. OHG Consulting: Chestra V3 – For companies on the move; Überblick über Chestra, Band1, Aug 1996
6. J. Withalm: New Business Strategies Stress QA in Proceedings of the Second World Congress for Software Quality, Yokohama, Japan, September 2000.
7. O'Reilly Enterprise Java Beans (ISBN 3-89721-192-0).