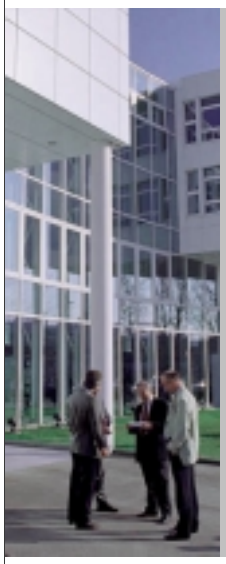
zühlke
DIE DENKFABRIK.



Rapid Web Application Development @ ALSTOM Power

Herausforderungen, Entscheidungen und Resultate

Jürgen Happel, jha@zuehlke.com

Einführung

- **Die Präsentation**
  - Ein Erfahrungsbericht über eBusiness Projekte mit Java
- **Ziele**
  - Unser Erfolgsrezept KISS verkaufen
  - Mut für einfache Lösungen machen
  - Open Source loben

zühlke

• Ziele:

• Einfachheit und Mut sind zwei von vier Grundwerten von “extreme Programming”

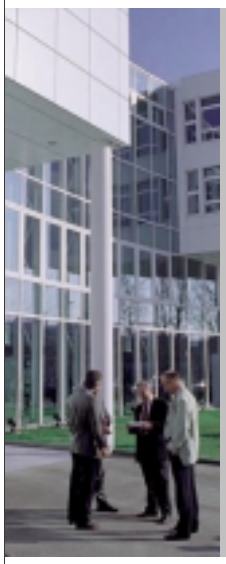
• Open Source hat in unseren Projekten eine grosse Rolle gespielt.

- Herausforderungen:
  - ALSTOM: Mit wem wir es zu tun haben
  - Umfeld: In welchem Rahmen wir uns bewegen
  - Projekte: Welche Aufgaben wir lösen mussten und noch müssen

- Entscheidungen:
  - Enterprise Java Beans: Ja oder nein
  - Framework: Wie können wir am effektivsten Entwickeln?
  - Persistenz: Wie wollen wir unsere Daten speichern?

- Resultate:
  - Framework: Was es ist und was es kann
  - Fazit: Lehren, die wir gezogen haben
  - Fragen und Antworten
  - Verständnis-Fragen auch zwischendurch

zühlke
DIE DENKFABRIK.



Herausforderungen

ALSTOM Power

Umfeld

Projekte

ALSTOM Power

- **ALSTOM**
  - Produkte und Dienstleistungen im Energie- und Transport-Sektor
  - Über 140’000 Angestellte in ca. 70 Ländern
  - Umsatz 2001 über 24 Milliarden Euro

- **ALSTOM Power**
  - Über 48’000 Angestellte
  - Umsatz 2001 über 12 Milliarden Euro
  - Im Mai 2000 ALSTOM übernimmt ABB Power zu 100 %

zühlke

• ALSTOM Power spielt eine sehr wichtige Rolle im ALSTOM Konzern, wie die Zahlen belegen.

• Durch die Verteilung auf 70 Länder, durch Übernahmen und Fusionen besteht ein grosses Potential für eBusiness Lösungen, die das Unternehmen transparenter, flexibler und damit effektiver machen können.

Umfeld - Organisatorisch

- **Organisatorisch**
- Multikulturell (Länder): 
- Multikulturell (Niederlassungen): ALSTOM, Ex-ABB
- Applikations Hosting Provider undefiniert
- eBusiness Dachprojekt im Anfangsstadium

zühlke

•Organisatorisches Umfeld

• Die Multikultur bringt viele unterschiedliche Erwartungen an die Projekte mit sich:

- Produkte Know-How Aufbau
- Effektivere Prozesse
- Fokus Intranet oder Internet

• Wo und bei wem unsere Applikationen gehostet werden ist undefiniert.

• Der Kunde hat Varianten, wo er den Hosting-Service in Anspruch nimmt. Dabei sind für ihn der Service und die Kosten relevant (Rugby, Baden, …)

• Ein eBusiness Dachprojekt existiert in einem Anfangsstadium

- Standards werden erst definiert
- Noch keine Plattform vorhanden
- kurzfristig keine Ergebnisse zu erwarten

Umfeld - Technisch

- **Technisch**
- Heterogene Umgebung (Solaris, AIX, HP-UX, Windows)
- Keine einheitliche Web-Applikations Infrastruktur
- Deployment Plattform undefiniert
- Legacy Tools (C++ auf Windows, Fortran auf AIX)

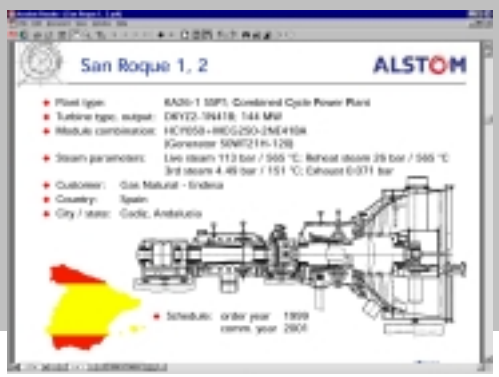
zühlke

•Technisches Umfeld

- Grundsätzlich heterogenes Umfeld (Solaris, AIX, HP-Unix, Windows)
- Produktive Applikationen (2) laufen unter Jrun oder mit ASPs
- Später wird IBM Websphere zur Standard Plattform erklärt
- Deployment Plattform zunächst undefiniert
- Legacy Applikationen müssen integriert werden

Das eTender Projekt 1/2

- **eTender als Grundstein**
- Erstes eBusiness Projekt mit ALSTOM Power
- Richtungsweisend für Folgeprojekte
- Sales Tool (Dampfturbinen)
- Selektion von Produkten mit Angebotserstellung

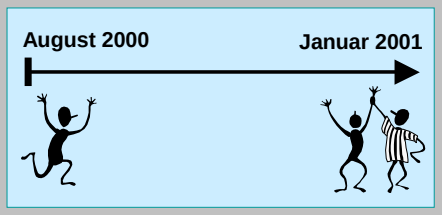


zühlke

• eTender
  • Erstes Projekt, das die Basis für alle weiteren Projekte gelegt hat.
  • Angebote werden online als PDF Dokument erzeugt
  • Komplexer Selektions-Prozess (Dokumentiert in 80 HTML Seiten)
  • Implementiert als Wizard-User-Interface
  • Web-Editor für Angebots-Textbausteine
  • Grosser Anteil Business-Process Engineering in allen Projekten

Das eTender Projekt 2/2

- **Zeitrahmen für eTender**
  - eTender soll innerhalb 6 Monaten eine erste Version liefern.



- **Ressourcen für eTender**
  - 2 ALSTOM Mitarbeiter
    - kein Java Know-How
    - z.T. kein OO Know-How
    - sehr gutes Business Know-How
    - z.T. in England
  - 1 Zühlke Software Ingenieur
  - 1 Zühlke Software Architekt
  - 1 Zühlke Projekt Manager

zühlke

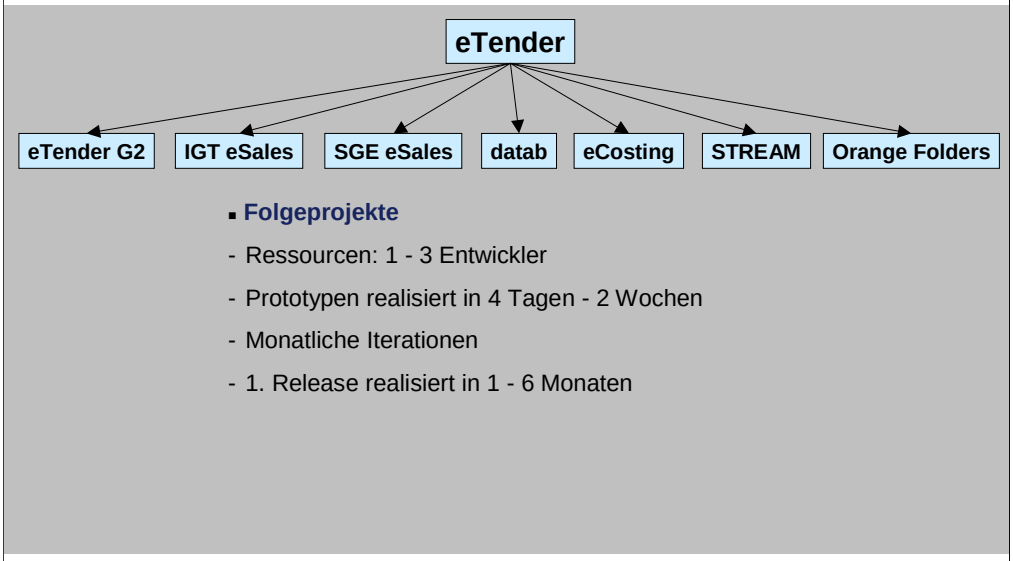
•Zeitrahmen:

- In 6 Monaten muss eine brauchbare Version fertig sein

• Ressourcen:

- 2 Entwickler von ALSTOM (z.T. temporär aus England)
- 1 Entwickler von Zühlke
- 1 Architekt von Zühlke (ca. 60 %)
- 1 Projektmanager von Zühlke (ca. 20 %)
- Die Kollegen von ALSTOM haben kein OO- und/oder kein Java-Know-How. Auch nicht im Bereich Web-Technologien.
- Kunde wünscht Coaching und Know-How Transfer

Folgeprojekte: Übersicht



- **Folgeprojekte**
- Ressourcen: 1 - 3 Entwickler
- Prototypen realisiert in 4 Tagen - 2 Wochen
- Monatliche Iterationen
- 1. Release realisiert in 1 - 6 Monaten

zühlke

- Wir haben eine Reihe von Folgeprojekten
- Pro Projekt sind jeweils 1 - 3 Entwickler beschäftigt
- Jeweils gleiches Vorgehen:
    - Ein Prototyp auf Basis von eTender
    - Monatliche Iterationen (maximale Dauer)
    - 1. Release innerhalb 1 -6 Monaten
    - Viel Zeit für Business Engineering notwendig

Folgeprojekte: IGT eSales, SGE eSales

- **IGT eSales**
  - Sales Tool (Gasturbinen)
  - eTender sehr ähnlich, aber einfacherer Selektions-Prozess

- **SGE eSales**
  - Sales Tool (Generatoren)
  - PDF-Generierung aus ausgewählten technischen Dokumenten
  - Selektion von Generatoren nach Art und Ausstattung
  - Berechnet Preis und Lieferzeit entsprechend gewählten Optionen

zühlke

- Ein kurzer Einblick in die Business Cases der Projekte …
- Jeweils zwei Projekte starten
  - Anfang 2001
  - Mitte 2001
  - Ende 2001

Folgeprojekte: datab, eCosting

- **datab**
  - Entwickler/Thermodynamiker Tool (Dampfturbinen)
  - Auswertung von Kraftwerks-Abnahme Messungen

- **eCosting**
  - Projekt-/Produktmanager Tool (Dampfturbinen)
  - Kosten-Kalkulation und -Verfolgung für Dampfturbinen Produkte

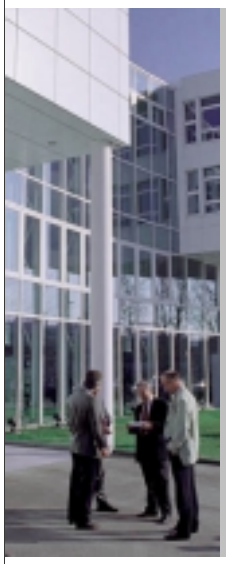
zühlke

Folgeprojekte: STREAM, Orange Folders, ...

- **STREAM**
  - Entwickler Tool (Dampfturbinen)
  - Entwicklung von Turbinenbeschaufelungen
  - Integration von Aerodynamik Tools
- **Orange Folders**
  - Produktentwickler & Sales Tool (Dampfturbinen)
  - Management System für technische Daten und Produkt Beschreibungen
  - Integration von Thermodynamik Tools

zühlke

zühlke
DIE DENKFABRIK.



Entscheidungen

Enterprise Java Beans

Framework

Persistenz

Entscheidungen - Übersicht

- **Java ist als Basistechnologie vom Kunden vorgegeben**
- **Die wichtigsten Entscheidungen:**
  - Technologie:
    to EJB or not to EJB
  - Framework:
    Open Source, kommerziell, Eigenbau
  - Persistenz:
    Datenbank oder Filesystem

zühlke

Ich erkläre alle Entscheidungen nach dem gleichen Muster.

- Varianten:
  - Die Entscheidungsmöglichkeiten

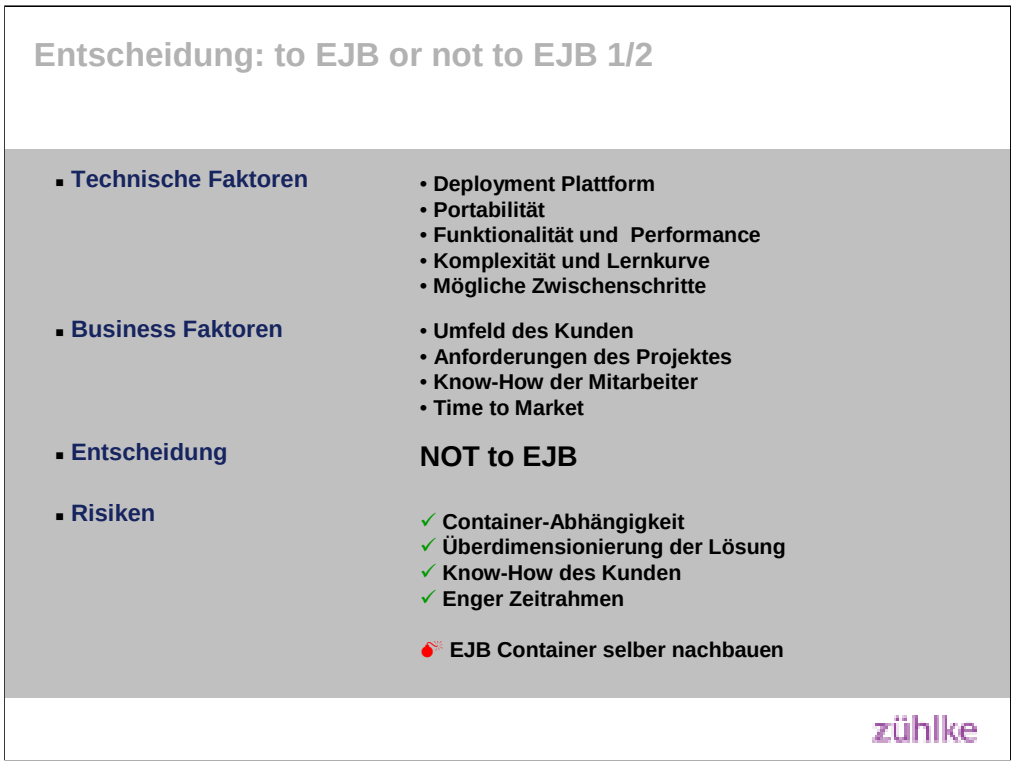
- Technische Faktoren:

- Business Faktoren:

- Entscheid:

- Risiken:
  - Adressierte Risiken
  - Neue/Offene Risiken

- Quellen:
  - Informationsquellen zur Entscheidungsfindung



• Deployment Plattform:
    • Applikations-Server
    • -> war lange Zeit bzw. ist noch immer undefiniert (JRUN, ASPs, später Websphere)

•Portabilität:
    • Applikations-Server ist undefiniert: Kann man die Lösung einfach portieren?
    • -> EJBs sind leider nicht einfach zwischen Containern zu portieren
    • -> eTender läuft auf JRUN, IPLANET, TOMCAT, Websphere

• Funktionalität und Performance:
    • Welche Features bieten EJBs? Welche brauchen wir?
    • Wie ist die Performance?
    • -> EJB Features nicht notwendig für eTender (z.B. keine Transaktionen)
    • -> Maximal 50 concurrent Users erwartet
    • -> Clustering, Load-Balancing, Connection-Pooling können auch Web-Container
    • -> Bekannte Performance Probleme mit EJBs

• Komplexität und Lernkurve:
    • Nötige Grundlagen
    • Einarbeitungsaufwand in Technologie + Produkte
    • -> ALSTOM Mitarbeiter müssen erst Java Grundlagen lernen (z.T. auch OO-Konzepte). Nicht gleich mit EJBs und EJB Container Know-How erschlagen.
    • -> ALSTOM betrachtet EJBs als zu „bleeding edge“
    • -> Fehlende Ressourcen bei ALSTOM, erfahrene Entwickler sind Mangelware

• Zwischenschritte:
    • Die Architektur soll EJB für später nicht ausschliessen
    • möglich z.B. mit JMS nutzen
    • Wir wollen klein und einfach anfangen und dann langsam ausbauen.

• Umfeld des Kunden:
    • Welche Standards gibt es?
    • Hat der Kunde Know-How und Manpower

Entscheidung: to EJB or not to EJB 2/2

- **Quellen**
  - **Do you really need Enterprise JavaBeans?**
    Anil Hemrajani, JavaWorld 2001
  - **Deciding whether EJB is appropriate**
    Ed Roman, Mastering Enterprise JavaBeans 2nd edition,
    September 2001 (Auszug als Artikel auf TheServerSide.com)
  - **Diskussion auf TheServerSide.com**
    zu Ed Roman's o.g. Artikel
  - **J2EE Project Dangers**
    Humphrey Sheil, JavaWorld 3/2001
  - **Editorial Java Magazin 10/2001**
    Sebastian Meyen
  - **Lutris Enhydra 3.5 Java/XML Application Server Solution**
    Sizing Guide, www.lutris.com/media/LutrisSG.pdf

zühlke

- Die Quellen zeigen nur das für und wieder der Ansätze.
- Die Quellen bestätigen, dass es sinnvoll sein kann, auf EJBs zu verzichten.
- J2EE Dangers
  - Not knowing Java -> ALSTOM Mitarbeiter
  - Not knowing EJB -> ALSTOM Mitarbeiter
  - Not deploying where you develop -> Plattform undefiniert
  - Over-engineering (EJB ja/nein)
- Java Magazin / TheServerSide / JUGS Homepage:
  - Eine Studie von Gartner meldet viele überdimensionierte J2EE Lösungen. 60% der Web-Anwendungen würden keine EJBs benötigen.
  - Gartner: Unternehmen hätten letztes Jahr 1 Milliarde Dollar sparen können, weil ihre Lösungen auch gut ohne EJBs ausgekommen wären.
  - Know-How und Ressourcen Problem im EJB Bereich an. Erfahrene Java-Entwickler sind Mangelware.
- Lutris:
  - Entwickler des Enhydra Open-Source Applikations-Servers melden 7-14x bessere Performance mit einer reinen Servlet Lösungen der bekannten Pet Shop Applikation (mit vergleichbarer Architektur). Die Tests sind von Testlabors von Intel bestätigt worden (http://www.lutris.com/media/LutrisSG.pdf).
- **Die J2EE/EJB best practices und patterns funktionieren auch ohne EJBs!**
- EJBs bringen einen grossen Overhead mit sich, falls man viele Features nicht braucht.

J2EE Project Dangers

http://www.javaworld.com/javaworld/jw-03-2001/jw-0330-ten.html

Do you really need Enterprise Java B eans

http://www.javaworld.com/javaworld/jw-10-2000/jw-1006-soapbox_p.html

Deciding whether EJB is appropriate

http://www.theserverside.com/home/thread.jsp?thread_id=8961

Entscheidung:
Open Source, kommerzielles oder eigenes Framework 1/2

▪ **Technische Faktoren**	• **Komplexität des Frameworks** • **Flexibilität des Frameworks** • **Reuse von Open Source** • **Produktreife (Mitte 2000)**
▪ **Business Faktoren**	• **COTS Problematik (auch mit Open Source)** • **Applikationen, Umgebung und Mitarbeiter sollen mit dem Framework wachsen** • **Konzentration auf Projekt-Anforderungen**
▪ **Entscheidung**	**Eigenes Framework als Grundgerüst mit Integration von Open Source Lösungen**
▪ **Risiken**	✓ **Einarbeitungszeit** ✓ **Unabhängigkeit** ✓ **Komplexität und Over Engineering** **Das Rad neu erfinden**

zühlke

- Komplexität:
  - Features
  - Overhead
  - Einarbeitung
- Flexibilität:
  - Konfigurationen, Integrationen
- Reuse von Open Source:
  - Plug-in von Open Source Lösungen
- Produktreife
  - Mitte 2000
- COTS (commercial off the shelf) Problematik
  - gibt es proprietäre Ansätze?
  - Abhängigkeiten mit anderen Produkten (z.B. spezielle Datenbank)
  - Abhängigkeiten zu unreifen Technologien (XML-Umfeld Mitte 2000, z.B Schemas)
  - Weiterentwicklung (Ja/Nein)
  - Weiterentwicklung (in falsche Richtung)
  - Lebensdauer
  - Support
  - Lizenzen

- -> Das Framework entsteht zunächst als integrierter Teil des eTender Projektes.
- -> Unser Framework ist einfach und übersichtlich
- -> Wir konnten uns daher zunächst voll auf die Anforderungen des Projektes konzentrieren.
- -> Das Framework wird später aus eTender herausgelöst.
- -> Wir haben Reuse von Open Source unter unseren eigenen Interfaces
- -> Wir bleiben unabhängig von einem Produkt
- -> Applikationen, Umgebung und Mitarbeiter können mit dem Framework wachsen

Entscheidung: Open Source, kommerzielles oder eigenes Framework 2/2

- **Quellen**
  - **E-Business Anwendungen mit J2EE**
    Stephan de Haas, Objekt Spektrum 5/2000
  - **Writing a Reusable Implementation of the MVC Design Pattern**
    Prashant Sarode, JavaReport Juli 2001
  - **Frameworks save the day**
    Humphrey Sheil, JavaWorld 2001
  - **Alles unter einem Tipi**
    Projekte der Apache-Plattform im Überblick
    Frank Bensberg, Lofi Dewanto, Java Magazin 4.2001

zühlke

• Die Quellen stellen eigene Frameworks vor oder Diskutieren den Einsatz von Open Source Frameworks.

• Was muss ein Framework können/sein.

• Eine Organisation, die in ein eigenes Framework investiert kann:

- •klein anfangen
- •Know-How und Infrastruktur mit dem Framework wachsen lassen
- • die wichtigsten und kurzfristigen Probleme zuerst lösen
- • sich iterativ an ändernde Technologien und Anforderungen anpassen (Beispiel: wir haben JAAS in 2-3 Tagen integriert)

• Es gibt einen Open Source Boom:

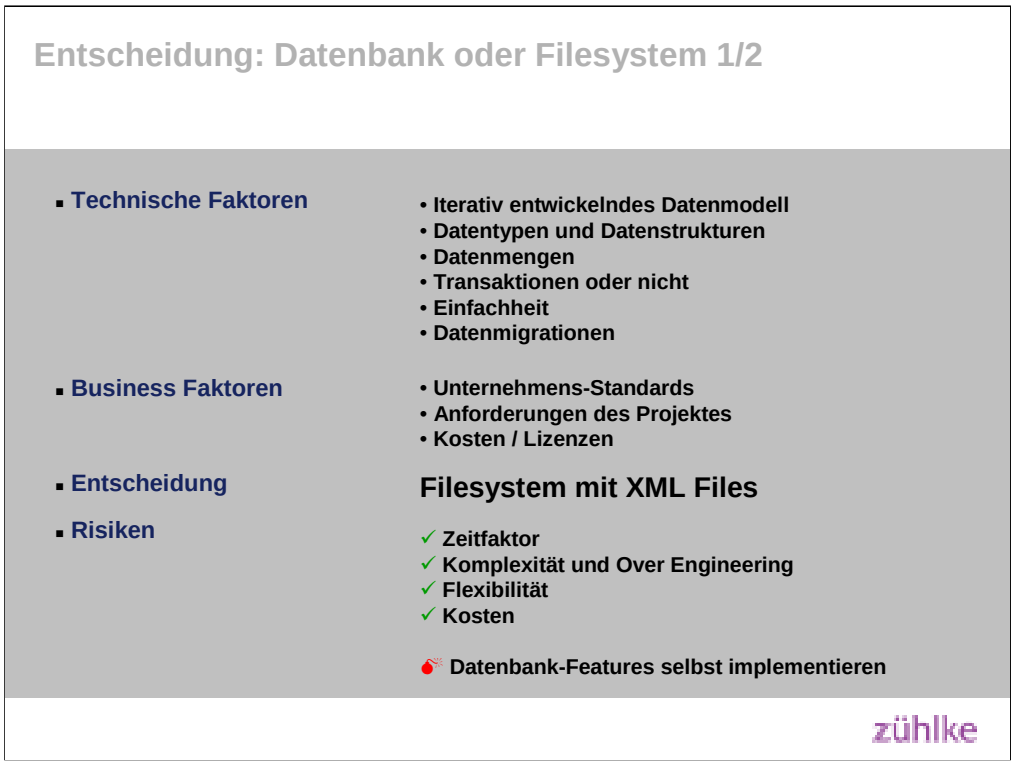
- • Positiv: Viele Probleme sind schon gelöst
- • Negativ: Auf welches Pferd setzen?

• Alleine bei der Apache Initiative gibt es mehrere Frameworks, die z.T. identische Bereiche abdecken (Cocoon, Struts, Turbine, Jetspeed = alles MVC Lösungen mit unterschiedlichen Schwerpunkten)

• Eine Konsolidierung der Open Source Projekte ist notwendig.

• Z.B. Expresso entwickelnt sich auch Richtung Apache/Struts

Frameworks save the day:
http://www.javaworld.com/javaworld/jw-09-2000/jw-0929-ejbframe.html



• **Filesystem mit XML Files:**

• Sehr effizient für iterative Entwicklung des Datenmodelles (kein ERM Design)

• In unserem Framework bleibt XML transparent für die Applikation.

• Als Strategie Pattern implementiert (nach dem Java Data Object bzw. Data Access Object Ansatz)

• Ermöglicht späteren Umstieg auf Datenbank

• Mehrheitlich komlexe, stark strukturierte, dokumenten-orientierte Daten

• Geringe Datenmengen (ca. 50 MB Stammdaten)

• Keine Transaktionen benötigt

• Einfachheit

    • Installationen

    • Daten-Manipulationen

    • Beispiel-/Test-Daten bereitstellen

    • Transparenz/Unabhängigkeit für Presentation Tier Entwickler

    • Datenmigration XML -> XML (merge, XSL)

    • Datenmigration XML -> Datenbank (Oracle, Oracle IFS, dbXML)

• Versionierung/Historisierung (z.B. CVS)

• Persistenz bereits in Prototypen kein Problem

• Hybrid-Lösungen möglich (XML + RDBMS, oder XML Datenbank)

• Keine Lizenz-Problematik

Entscheidung: Datenbank oder Filesystem 1/2

- **Quellen**

- **Objects in the Database**
  An overview of Sun's Java Data Object specification
  David Jordan, Java Report Juni 2000

- **Java Data Objects**
  Java Specification Request (JSR 000012)
  Version 1.0, Proposed Final Draft
  Craig Russel, Sun Microsystems Inc.

- **Java Data Objects Specification**
  Transparent Interface for Persistent Data Storage
  Heiko Bobzin, POET Software Corp., JavaOne 2000

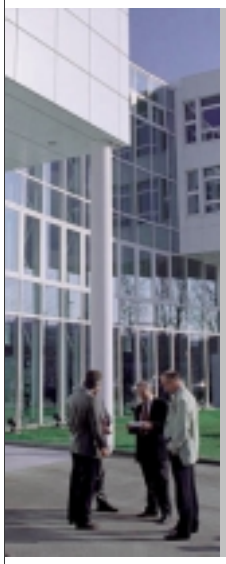
zühlke

- Java Data Objects:
  - Der Datenspeicher bleibt transparent für den Java Entwickler
  - Pluggable Java Data Object Implementierung z.B.
    - Filesystem
    - Relationale
    - OO-Datenbank
    - XML Datenbank
  - Integriert in Forte 3.0

Java Data Objects (JDO) Specification:
http://www.jcp.org/jsr/detail/12.jsp

Java Data Objects vs. Entity Beans:
http://www.theserverside.com/discussion/thread.jsp?thread_id=771

zühlke
DIE DENKFABRIK.



Resultate

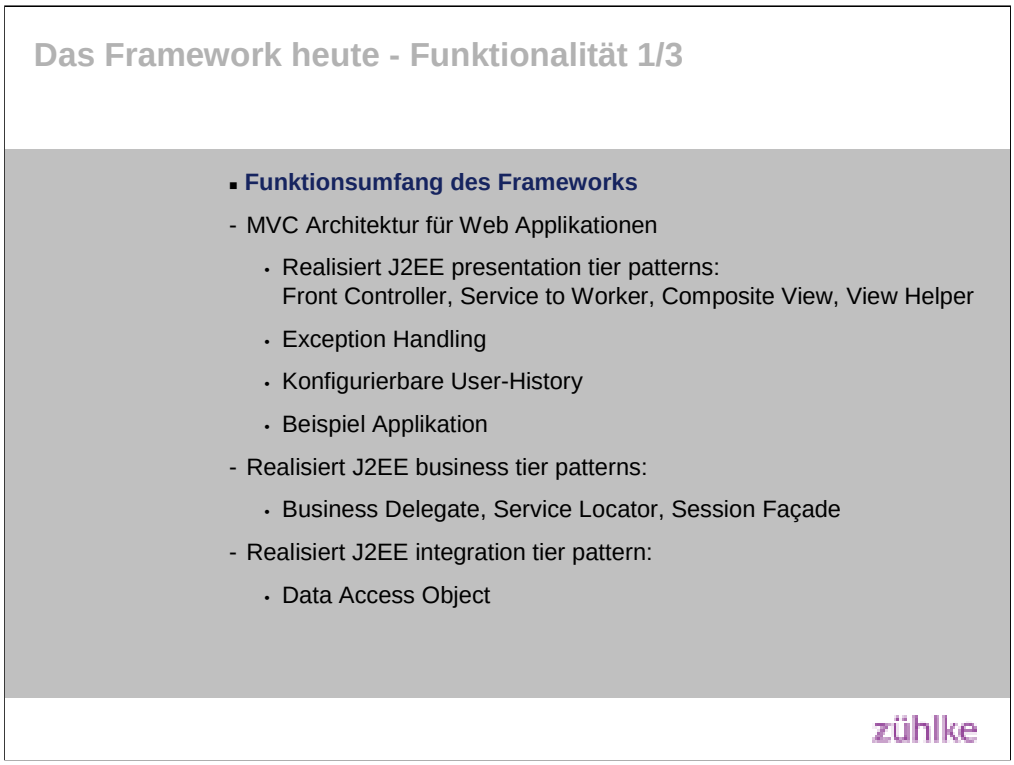
Das Framework heute

Fazit

Das Framework heute - Eigenschaften

- **Eigenschaften des Frameworks**
- Evolutionäre Prototypen innerhalb von Tagen realisiert
- Java Entwickler innerhalb von Stunden produktiv
- Beispiel-Applikation innerhalb von Minuten installiert
- Schlüssel für die Akquisition weiterer Projekte
- Inzwischen Oracle Internet Filesystem integriert (als Persistenz-Option)
- Ca. 260 Klassen und Interfaces
- Ca. 70 Unit- und Integrations-Test Klassen

zühlke



• J2EE Patterns funktionieren auch ohne EJBs. Ermöglichen später EJBs zu integrieren.

•MVC Architektur:

- • Alle anderen Services sind unabhängig vom mvc package
- • Theoretisch kann die MVC Implementierung immer noch durch ein Open Source Framework ersetzt werden.

• User-History

- • Optional kann die User-History (Eingaben des Users und Antworten des Systems) persistent gemacht werden.
- • Sales Leute sind z.T. am Verhalten der Anwender interessiert (z.B. “Mit welchen Parametern hat ein Anwender versucht eine Turbine zu finden?”).

• Beispiel Applikation:

- • Einfache Installation mit Ant-Script
- • Ermöglicht einen schnellen Einstieg
- • Zeigt die wichtigsten Dienste des Frameworks
- • Kann direkt als Basis für ein neues Projekt dienen

Das Framework heute - Funktionalität 2/3

- **Funktionsumfang des Frameworks (Fortsetzung)**
- Logging
- Sicherheit (Auf Daten- und Funktions-Ebene)
- Persistenz (XML-Filesystem und Oracle IFS)
- Versionierung
- Datentypen-Definition und Validierung
  - Darstellung mit Custom JSP Tags
  - Einheiten und Einheiten-System Konvertierungen
- Workflow- und UI-Wizard Framework
- Notification (mail)

zühlke

- Sicherheit:
  - JAAS kann als Strategie-Pattern optional eingesetzt werden (JDK 1.3)
  - Eigene Implementierung als weitere Option (ohne JDK1.3). Deckt in einer Implementierung Login-, Daten- und Funktions-Zugriffe ab (gleiche Abstraktion für 'Zugriff auf Daten' und 'Zugriff auf Funktionen')
- Persistenz:
  - Interface verbirgt Implementierung (XML-Filesystem, IFS, relationale DB)
  - Konzept von Java Data Objects übernommen
- Versionierung:
  - Jedes persistente Objekt kann versioniert werden. Das muss nicht von Anfang an sein.
  - Versionierung von Objekte bleibt transparent falls nicht von Interesse.
- Datentypen:
  - Eingabemasken können als XML-Files definiert werden
  - Keine Änderungen in JSPs bei
    - Änderungen an den Datentypen-Definitionen
    - Einfügen/Löschen von einzelnen Eingabe-Werten
    - Tag-Library

Das Framework heute - Funktionalität 3/3

- **Funktionsumfang des Frameworks (Fortsetzung)**
- Benutzer Verwaltung (auch als Applikations-Feature möglich)
- Anbindung von Legacy Tools über SOAP-RPC
- Dokumenten Generator (XML zu PDF)
- Deployment Scripts
- Tests

zühlke

- Benutzer-Verwaltung:
  - z.T. werden auch in Applikationen neue Benutzer und Gruppen angelegt (eCosting)
- Dokumenten Generator:
  - Realisiert mit FOP
  - Diesen verwenden wir auch für die RUP Dokumente unserer Projekte
- Deployment-Scripts:
  - Bietet Ant-Scripts für Deployment und Release-Management
  - Bietet Ant-Scripts für Tests mit neuen Framework-Releases
- Tests:
  - Junit und HttpUnit Tests garantieren für die Qualität
  - Applikationen können mit verschiedenen Releases getestet werden.

Das Framework heute - Open Source

- **Open Source im Framework**
  - Apache
    - Ant
    - Xerces
    - Xalan
    - FOP
    - SOAP
  - JDOM
  - JUnit
  - HttpUnit
  - (Tomcat)

zühlke

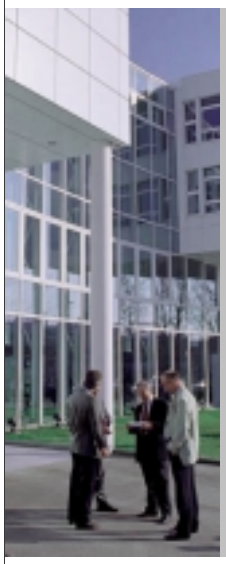
- Ant (Skripte für)
  - Build
  - Release
  - Tests
  - Deployment
- Xerces
  - Unser XML-Parser
- Xalan:
  - XSLT Prozessor
- FOP:
  - PDF Generierung
- SOAP:
  - Simple Object Access Protocol
  - Implementierung des Protokolles
  - Legacy Tool Integration
- JDOM:
  - Allgemeines XML-Handling
- Junit:
  - Unit Tests
- HttpUnit:
  - Last Tests
  - Navigations-Test
- Tomcat:
  - Entwicklungsplattform

Fazit: KISS

- **Man darf sich für einfache Lösungen nicht zu schade sein.**
- **Einfache Lösungen funktionieren häufiger und länger als man glaubt.**
- **Einfache Lösungen können wachsen. Komplexe Lösungen schrumpfen nie (mehr).**
- **J2EE und EJB Patterns funktionieren auch ohne EJBs.**
- **Erfolgsrezept:**
  Design Patterns und flexible Interfaces ermöglichen,
  - mit der einfachsten Variante zu starten.
  - bei Bedarf auf komplexere Implementierungen umzusteigen.
- **Open Source war entscheidend für den Erfolg.**

zühlke

zühlke
DIE DENKFABRIK.



Rapid Web Application Development @ ALSTOM Power

Fragen & Antworten

Jürgen Happel, jha@zuehlke.com