FORSOFT

Bayerischer Forschungsverbund
Software Engineering

Architektur-Workshop München 2000

FIZ München
5. Dezember 2000

Manfred Broy
Michael Gnatz
Frank Marschall
Sascha Molterer
Gerhard Popp
Andreas Rausch
Wolfgang Schwerin
(Hrsg.)

Gesponsert durch:
BMW AG München
Bayerische Forschungsstiftung
FORSOFT - Bayerischer Forschungsverbund Software Engineering
Projekt MARLIN – Models and Architecture for Large Information Systems
Projekt ZEN - Zentrum für Technik, Methodik und Management der Software- und Systementwicklung




Softwarearchitekturen und Technologien sind eines der aktuellen, "heißen" Themen in der Informatik. In unserem Forschungsverbund FORSOFT und an unserem Lehrstuhl beschäftigen wir uns seit längerer Zeit mit diesem Thema. Wir, das Teilprojekt ZEN, veranstalten heute einen Workshop zum Thema Softwarearchitekturen, bei dem wir Industrie und Forschung zusammen bringen wollen.

Ziel dieses Workshops ist es sowohl positive wie auch negative Erfahrungen im Themenfeld Softwarearchitekturen auszutauschen. Dieser Workshop soll ein erster Schritt sein, um ein Diskussions- und Wissens-Forum im Bereich Softwarearchitekturen im Spannungsfeld zwischen Industrie und Forschung zu etablieren.

Der Workshop besteht aus einer Reihe von Vorträgen. Diese Vorträge sollen jeweils 20 Minuten dauern, mit anschließenden 10 Minuten Diskussion. Am Ende des Workshops wollen wir noch gemeinsam einige zentrale Fragen in einer moderierten Diskussion besprechen.

Die einzelnen Vorträge sollten keine Werbevorträge sein, sondern eine oder mehrere konkrete Architekturen aus der industriellen Praxis vorstellen und kritisch beleuchten. Dabei soll in einem Vortrag in den ersten 10-15 Minuten zunächst die konkrete Architektur vorgestellt werden, um diese in den verbleibenden 5-10 Minuten anhand von 3-4 Leitfragen zu analysieren.

Ich freue mich über Ihre Teilnahme.

Prof. Dr. Manfred Broy

Programm zum Architektur Workshop München 2000

5. Dezember 2000

Prof. Dr. Manfred Broy: Begrüßung	Technische Universität München	von 8:45 Uhr bis 9:00 Uhr

Andreas Rausch: Softwarearchitekturen für komplexe, verteilte Systeme	Technische Universität München	von 9:00 Uhr bis 9:30 Uhr

Bei kleinen Projekten ist die Softwarearchitektur häufig mehr oder weniger vorgegeben. Große, verteilte Softwaresysteme hingegen sind sehr komplexe Gebilde. Die Erstellung solcher Systeme birgt meist eine Vielzahl von Risiken. Einerseits ändern sich die Anforderungen an das System laufend und andererseits werden heute verstärkt neue und moderne Technologien eingesetzt. Eine stabile und erweiterbare Softwarearchitektur kann zumindest das technologische Risiko entscheidend minimieren. Darüber hinaus kann sie ein stabiles Fundament bieten, um die sich ändernden fachlichen Anforderungen, in die bestehende Lösung iterativ und inkrementell zu integrieren.

In diesem Vortrag werden wir einige, ausgewählte Softwarearchitekturen für verteilte Systeme vorstellen, die wir in verschiedenen Studentenprojekten erarbeitet und umgesetzt haben. Diese Softwarearchitekturen sind charakteristisch für verschiedene Anwendungsdomänen. Wir diskutieren die technische Lösung sowie die damit verbundenen Vor- und Nachteile der Architekturvarianten.

Dr. Markus Podolsky: Entwicklung und Einsatz von Musteranwendungs-architekturen bei BMW	BMW AG	von 9:30 Uhr bis 10:00 Uhr

Wesentliche Ziele der Softwareentwicklung bei BMW sind neben Verkürzung der Entwicklungszeit (Time-To-System) auch Erhöhung der Wiederverwendbarkeit, Verbesserung der Wartbarkeit usw.

Um diese Ziele zu erreichen, werden für immer wiederkehrende Fragestellungen mustergültige Architekturen - sowohl technische als auch Anwendungsarchitekturen - zusammen mit Pilotprojekten entwickelt und den Folgeprojekten zur Verfügung gestellt. Somit können diese auf bereits etablierten Lösungen aufbauen. Insgesamt erreicht man durch dieses Vorgehen eine große Ähnlichkeit im Design der verschiedenen Systeme, was sich gerade bei den Themen Wartung/Erweiterbarkeit als vorteilhaft erweist.

Im Rahmen der Präsentation wird aufgezeigt, wie Musterarchitekturen entwickelt werden und wie sie in den Projekten eingesetzt werden.

Dr. Andreas Dabelstein: Architektur für die Kopplung von IT-Systemen mit Simulationssystemen auf Basis HLA	ESG GmbH	von 10:00 Uhr bis 10:30 Uhr

Ziel von entscheidungsunterstützenden Systemen (Decision Support Tools) ist es, den Führungsprozess effektiv zu unterstützen. Effektive Unterstützung beinhaltet, die Qualität der Entscheidungen durch verbesserte Informationsaufbereitung und geringere Zeitanforderungen zu erhöhen. Eine Vorgabe ist dabei, unterschiedliche OR und Simulationsverfahren in den Führungsprozess integrieren zu können. Entscheidungszyklen müssen schnelle und hoch qualitative Entscheidungen erlauben. Daher ist es notwendig, dass Decision Support Tools langfristig gesehen ein integraler Bestandteil von Führungsinformationssystemen werden. Bis auf weiteres - solange dies noch nicht der Fall ist - müssen existierende entscheidungsunterstützende Systeme mit existierenden Führungsinformationssystemen verbunden werden.

Zur Realisierung dieser Anforderungen wird hier eine Integrationsplattform vorgeschlagen, die flexibel, modular, erweiterbar und interoperabel mit bestehenden Systemen sein soll. Hierzu sollen möglichst viele Standards benutzt werden. Dies betrifft einerseits eine standardisierte Beschreibung der auszutauschenden Daten (durch das Datenmodell ATCCIS) als auch andererseits die Verwendung von standardisierten Kopplungsverfahren, wie der High Level Architecture - HLA. Zum Nachweis wird ein Experimentalsystem entwickelt, dass das operationelle Führungsinformationssystem HEROS-2/1 Los 2 mit dem Simulationssystem KORA/OA über diese Integrationsplattform koppelt. Beschrieben wird die Systemarchitektur für diese Integrationsplattform. Betrachtet werden dabei die einzelnen SW-Module inklusive der zu definierenden Schnittstellen. Eingegangen wird auf das Konzept der High Level Architectur HLA sowie auf die notwendigen Abbildungen der Datenmodelle.

Kaffeepause	von 10:30 Uhr bis 10:50 Uhr

Prof. Dr. Johannes Siedersleben: Das Quasar-Architektur-Framework	sd&m research	von 10:50 Uhr bis 11:20 Uhr

Der Vortrag stellt Quasar vor, ein bei sd&m Research entwickeltes Framework für die Entwicklung individueller Anwendungen in Unternehmen. Quasar ist ein Akronym für Qualitäts-Software-Architektur und besteht aus einer Datanbank-Zugriffsschicht, einem Rahmen für den Anwendungskern (bestehend aus einer Reihe intelligenter Datentypen und aus Hilfsfunktionen/Utilities) und einer Präsentationsschicht für die Erstellung von GUIs.

Die Grundlage der Quasar-Architektur ist die Bildung von Komponenten, die über Schnittstellen entkoppelt sind. Ein wichtiges Prinzip beim Entwurf ist dabei die Trennung der Zuständigkeiten zwischen neutralen, techniknahen und anwendungsnahen Komponenten.

Der Vortrag führt in die konzeptuellen Grundlagen ein und stellt die Architektur in ihren Grundzügen vor. Da Quasar ein sehr anwendungsnahes Forschungsprojekt ist, werden Beispiele für den Einsatz bei der Erstellung von Anwendungen sowohl im Schulungs- als auch im Kundenumfeld gegeben.

Inge Hanschke: Entwicklung von Frameworks bei iteratec	Iteratec GmbH	von 11:20 Uhr bis 11:50 Uhr

E-Commerce und E-Business gewinnen zunehmend an Bedeutung. In diesem Kontext entwickelt die Firma iteratec für einen Kunden eine komplexe Web-Applikation. Auf Basis der J2EE-Plattform mit dem Bea WebLogic-Server erfolgt die Konzeption und Implementierung einer Multi-Tier-Anwendung mit Thin-Clients. Die technische Architektur dieser Anwendung soll die Basis für weitere Projekte im E-Commerce-Umfeld bilden. Durch die Abstraktion von spezifischen Anforderungen soll eine wiederverwendbare Grundlage (Framework) für Multi-Tier-Anwendungen geschaffen werden.

Die Entwicklung von Frameworks hat für iteratec eine große Bedeutung. Tragfähige technische Basen von Projekten bzw. das Wissen und die Erfahrungen aus der Entwicklung dieser technischen Basen bilden den Ausgangspunkt für die Entwicklung von Frameworks bei iteratec. iteratec hat organisatorische Strukturen und ein Vorgehen etabliert, um die technische Basis der im Rahmen von Projekten erstellten Software zu verbessern und um die Erstellung von Frameworks bzw. Komponenten von Frameworks für iteratec zu fördern. Anhand der technischen Ergebnisse der oben genannten E-Commerce-Anwendung wird das bei iteratec etablierte Vorgehen zur Entwicklung von Frameworks exemplarisch vorgestellt.

Dr. Wolfgang Daxwanger: Komponentenbasierte SW-Architektur für CAM Systeme	SEKAS GmbH	von 11:50 Uhr bis 12:20 Uhr

Das Hauptziel von CAM (Computer Aided Manufacturing) Systemen ist die Steuerung industrieller Fertigungsprozesse vom Rohmaterial bis zum Endprodukt. Dies erfordert die Koordination verschiedenster Softwaresysteme zur Erfüllung der einzelnen Aufgaben. Eine Architektur zur Verbindung dieser verteilten Softwaresysteme muss die robuste Integration heterogener Plattformen ermöglichen. Die einzelnen Softwaresysteme müssen sehr flexibel sein, da ihre Funktionalität stark vom Einsatzgebiet und Produktionsumfeld abhängt. Ein hoher Grad an Konfigurierbarkeit und Modularisierung ermöglicht die Wiederverwendung von einzelnen Softwaresystemen oder deren Teilkomponenten in anderen CAM Systemen.

Der Vortrag gibt einen Überblick über eine komponentenbasierte verteilte Systemarchitektur für den CAM Bereich. Als Vertreter der Softwaresysteme werden exemplarisch ein Alarmmanagement System und ein System zum Sammeln und Verarbeiten von Maschinen- und Betriebsdaten vorgestellt.

Mittagspause		von 12:20 Uhr bis 13:30 Uhr

Dr. Klaus Bergner: Flexible Unternehmensanwendungen: Integration eines EJB-Application-Servers mit einer Workflow-Engine	4Soft GmbH	von 13:30 Uhr bis 14:00 Uhr

Moderne komponentenbasierte Anwendungsarchitekturen versprechen hohe Flexibilität, da sich bei ihnen sowohl Geschäftsobjekte als auch Geschäftsprozesse rasch und leicht umkonfigurieren lassen. Wir berichten von unseren Erfahrungen bei der Realisierung eines Systems auf Basis eines EJB-1.1-konformen Application Servers und einer eingebetteten Java-Workflow-Engine. Dabei gehen wir insbesondere auf die Architektur des Gesamtsystems sowie die Erfahrungen mit den verwendeten Komponenten, Ansätzen und Standards ein.

Bernhard Davignon: Eine 3-tier Architektur für einen Standardlösungsanbieter	NSE Financial Solutions	von 14:00 Uhr bis 14:30 Uhr

Vorgestellt wird eine Architektur für einen Standardlösungsanbieter im Finanzdienstleistungsbereich. Wichtige Features dieser Architektur sind:

- Verschiedener Verteilungsszenarien (vom standalone Notebook bis zum Großrechner)
- Unterschiedlicher Vertriebskanäle (Filiale, Internet, Selfservice, ...)
- Betriebssystem, Datenbank und Middleware-Unabhängigkeit
- Investitionsschutz: heute entwickelte Software muss in zukünftigen technischen Umgebungen lauffähig sein.
- Einfaches Customizing der Standardlösung

Diese Anforderungen führen dazu, dass die Architektur einige Besonderheiten aufweist, die sie von gängigen Architekturmodellen, wie sie vor allem für Individuallösungen zum Einsatz kommen, unterscheidet.

Stefan Finkenzeller: Entwicklung der Systemarchitekturen in der Bayerischen Landesbank	Bayerische Landesbank	von 14:30 Uhr bis 15:00 Uhr

Eine große Herausforderung für die Bayerische Landesbank ist die Integration einer Vielzahl neuer Anwendungen und die Öffnung bestehender Kernsysteme. Hierbei spielen Sicherheit, Flexibilität, Schnelligkeit (Time-to-Market) und Erweiterbarkeit eine große Rolle.

Die aktuelle Softwarearchitektur der Bayerischen Landesbank entstand aus den Erfahrungen der letzten Jahre und den neuen eBusiness Anforderungen. Das Ergebnis ist eine flexible, moderne und komponentenbasierte Architektur auf Basis von Enterprise Java Beans.

Der Vortrag beschreibt die Entwicklung vom Terminal Client zur Thin Client Architektur und beleuchtet kritisch die eingesetzten Techniken.

Kaffeepause	von 15:00 Uhr bis 15:20 Uhr

Dr. Heinz Koßmann: Software-Architektur für Vermittlungsrechner in Telefonnetzen mit Internet-Konvergenz	Siemens ICN	von 15:20 Uhr bis 15:50 Uhr

Es wird die Softwarearchitektur von Vermittlungsrechnern in Telefonnetzen sowie von Servern zur Konvergenz von Internet und Telefonnetzen beschrieben.

Die Architektur für Vermittlungsrechner beherrscht erstens die Komplexität der Software von über 10 Millionen Quellcodezeilen, zweitens die sehr hohe Verfügbarkeit (Gesamtausfallzeit in einem Jahr < 3 Minuten), und drittens die hohe Leistungsfähigkeit bezüglich mehrerer Millionen Verbindungswünsche pro Stunde. Hierzu dienen geeignete Software-Struktureinheiten, deren Service-Schnittstellen und das zugrundeliegende Kommunikationsmodell mit "Location Transparency" und "Service Addressing". Diese ermöglichen u.a. das störungsfreie Einbringen neuer Versionen (Upgrade) im laufenden Betrieb. Zudem werden die Aspekte einer schrittweisen Produktion berücksichtigt.

Die Konvergenz von Internet und Telefonnetzen schafft das Potential für neue Mehrwertdienste wie "Click to Dial", "Click to Conference" etc, die die Nutzung von Telefonservices aus dem Internet bereitstellen. Die Architektur für Server zur Kopplung von Internet und Telefonnetz unterstützt die schnelle Bereitstellung von Internet-Services sowie die Möglichkeit der Offenlegung von Schnittstellen für die Anbindung externer Applikationsserver. Hierzu dient eine Gliederung der Server-Software in funktionale Komponenten, die in verschiedenen Services wiederverwendet werden können. Die Kommunikation zwischen den Komponenten erfolgt über klar definierte Schnittstellen, die zum Teil standardisiert sind.

Kriterien einer Architektur sind Verständlichkeit, Erweiterbarkeit, Skalierbarkeit und die Integration von Standards. Von zentraler Bedeutung sind dabei die Beschreibung der Schnittstellen und Systemabläufe.

Prof. Dr. Manfred Broy Moderierte Abschlussdiskussion	Technische Universität München	von 15:50 Uhr bis 17:00 Uhr

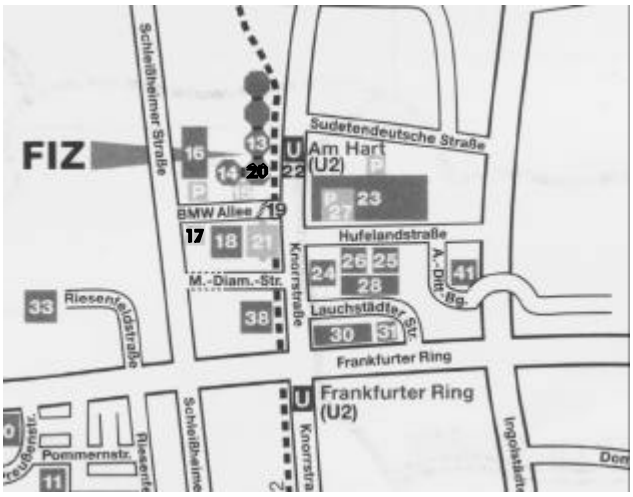
Organisation
Architektur-Workshop
München 2000

5. Dezember 2000

Tagungsort:

BMW AG
FIZ (Forschungs- und Ingenieur-Zentrum)
“FORUM“
Knorrstraße 147
80937 München

Anfahrtsskizze:



17:
Parkplatz Süd
(Anfahrt nur über Schleißheimer Straße)

20:
FIZ Haupteingang, Empfang

Auskünfte:

Gerhard Popp

Institut für Informatik
Technische Universität München
Arcisstraße 21
80290 München

Telefon 089/289-22094
Telefax 089/289-25310
E-Mail: popp@in.tum.de

Leitfragen zum Münchner Architektur Workshop 2000

Prof. Dr. Manfred Broy

Thema I: Beschreibungstechnik

1. Welche Beschreibungstechniken werden beim Entwurf eingesetzt und was wird beschrieben?
2. In wie weit wird die UML als Architekturbeschreibungssprache verwendet und welche Defizite tauchen dabei auf?

Thema II: Methodik und Vorgehen

1. Gibt es ein methodisches Vorgehen beim Entwurf und wie sieht es aus?
2. Wie weit ist der Architekturentwurf in den restlichen Entwicklungsprozess eingebettet?

Thema III: Wiederverwendung

1. Gibt es unternehmensweite Blueprints und Standardarchitekturen und in wie weit fließen diese beim Entwurf mit ein?
2. Auf welcher Ebene wird Wiederverwendung von Architekturen betrieben (Code-, Design-, Spezifikations-Ebene)?

Thema IV: Qualitätsmerkmale und Entwurfsprinzipien

1. Welche Qualitätsmerkmale werden beim Entwurf angestrebt und wie ist die Priorisierung (Wartbarkeit, Anpassbarkeit, Wiederverwendung, etc?)
2. Welche Entwurfsprinzipien werden beim Entwurf eingesetzt um bestimmte Qualitätsmerkmale zu erreichen (Modularität, lose Kopplung, klare Abhängigkeiten, etc.)?

Softwarearchitekturen für komplexe, verteilte Systeme

FORSOFT ZEN – Zentrum für Technik, Methodik und Management der Software- und Systementwicklung

Andreas Rausch
rausch@forsoft.de

Professor Dr. Manfred Broy
Lehrstuhl für Software & Systems Engineering
Bayerischer Forschungsverbund Software Engineering
Technische Universität München

Überblick

FORSOFT **ZEN**

- AutoMate: Mehrschichtige Architekturen für Computer Aided Engineering Werkzeuge
- AquaCard: Integration von Anwendungen vom Host bis zur SmartCard
- CROFT: Komponentenbasierte Informations-systeme mit Enterprise JavaBeans
- Softwarearchitekturen – Theorie und Praxis

Anforderungen an CAE Tools

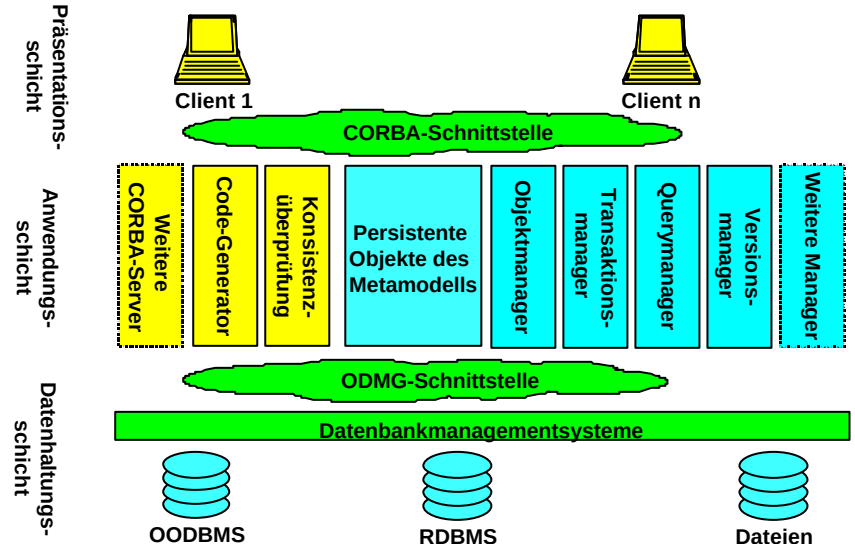
FORSOFT
ZEN

- Zentrales integriertes Produktdatenmodell
- Parallele und nebenläufige Bearbeitung der Daten
- Kooperation und Koordination verteilter Bearbeiter
- Einfache Integration neuer Funktionalität
- Standardisierte und einfache Entwicklung der Clients
- Integrierte Basisdienste mit Standardschnittstellen
- Verwendung von Standardkomponenten

⇨ Lösung: Skalierbare, flexible 3-Schichtenarchitektur

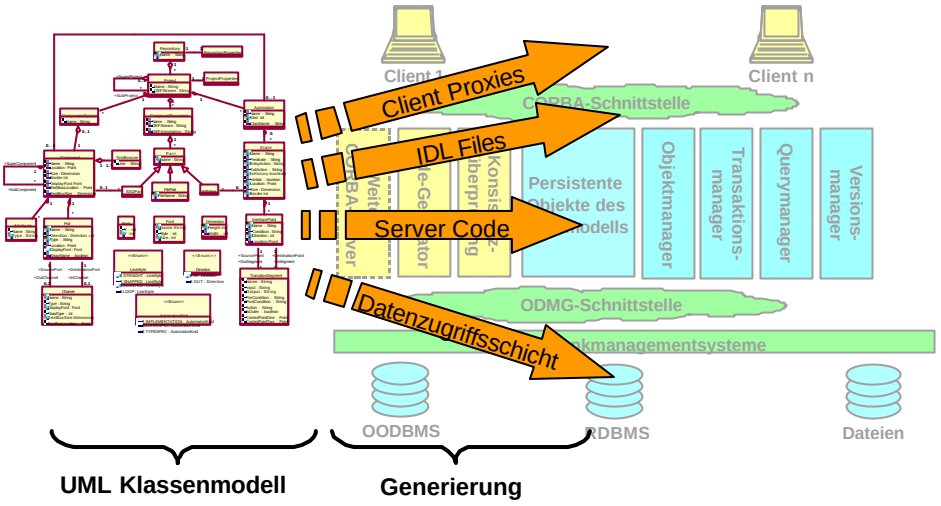
Architektur der Zielanwendung

FORSOFT
ZEN



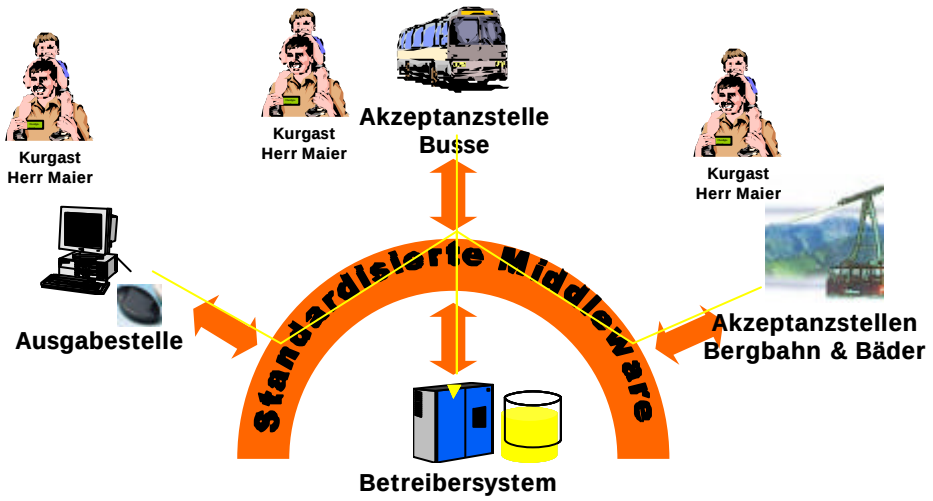
Generierung des Servers

FORSOFT
ZEN



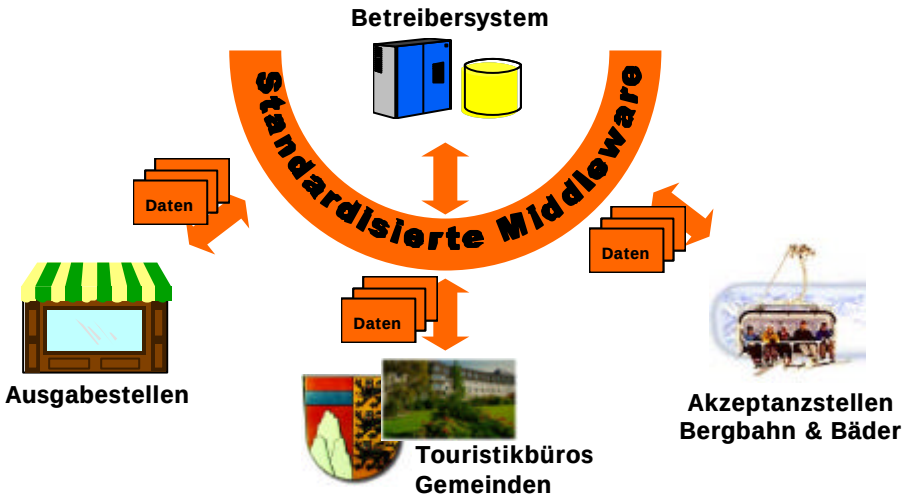
AquaCard aus Kurgast Sicht

FORSOFT
ZEN



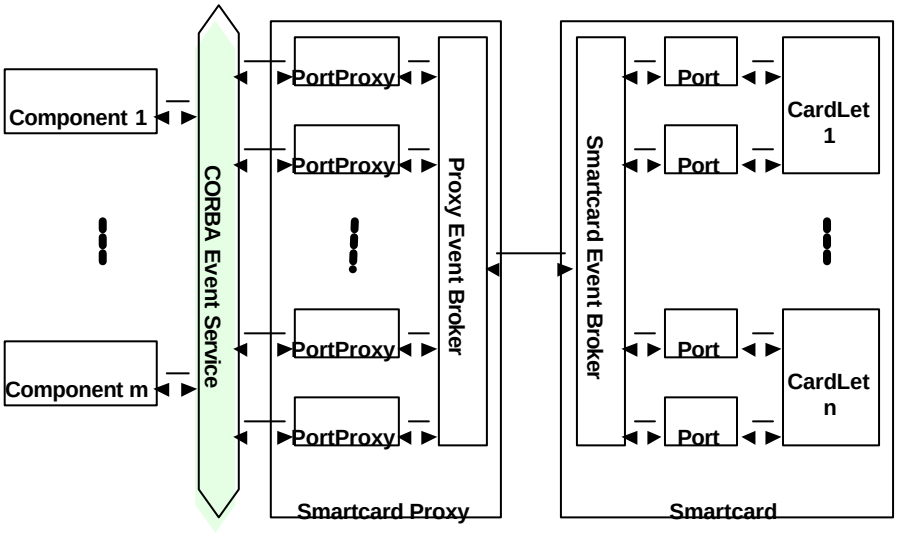
AquaCard aus Betreibersicht

FORSOFT
ZEN



Architektur der AquaCard

FORSOFT
ZEN

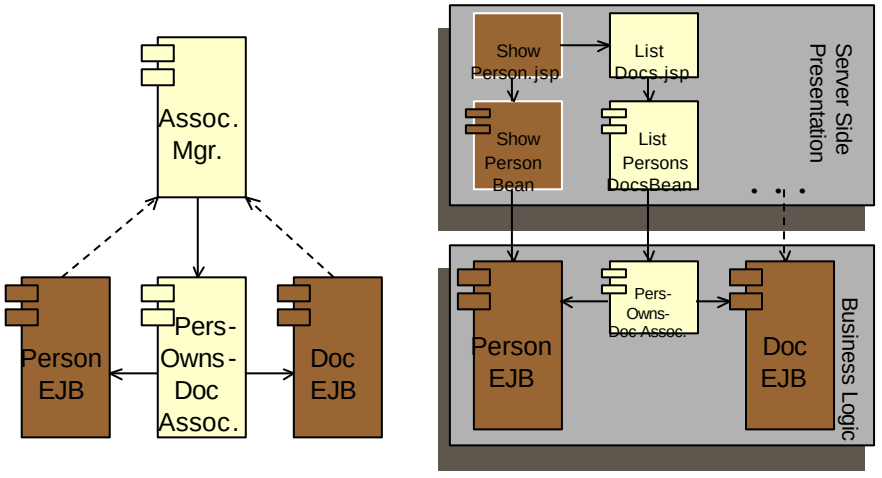


CROFT Informationssystem

FORSOFT
ZEN

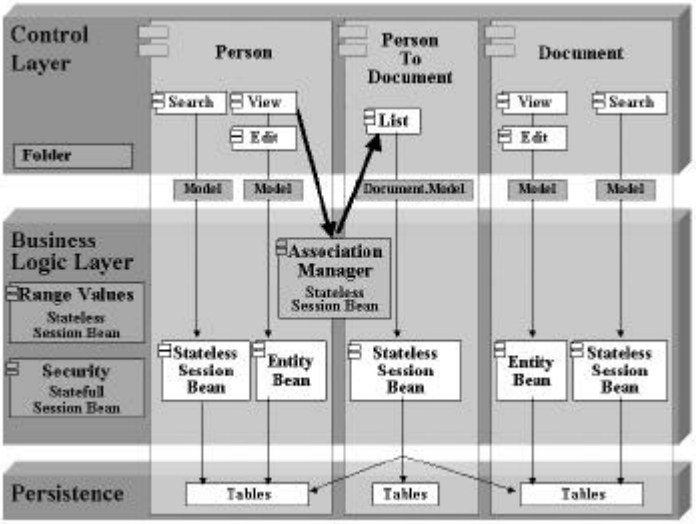
Flexible Assoziationen zwischen Enterprise JavaBeans

FORSOFT
ZEN



CROFT Architektur

FORSOFT
ZEN



Beschreibungstechniken

FORSOFT
ZEN

✓ Box & Line Diagramme für High-Level Architektursicht

✓ UML für Low-Level Architektursicht

? Standardisierte Beschreibungstechnik für High- und Low-Level Softwarearchitekturen

? Semantische Fundierung dieser Beschreibungstechnik

? Weitgehend Generierung von Code und Prototypen

Methodik und Vorgehen

FORSOFT **ZEN**

- ✓ Time Boxing und klare Einteilung in Phasen
- ✓ Ausgehend von einer weitgehend vollständigen Analyse
- ✓ Einbeziehung von existierenden Komponenten, Frameworks, Design Patterns und Standards
- ✓ Entwurf und weitreichender Test der Softwarearchitektur

- ? Fundiertes Verständnis des Entwurfsprozesses
- ? Welche Informationen müssen in den Entwurf einfließen
- ? Wie sieht das Ergebnis des Entwurfsprozesses aus
- ? Verfolgbarkeit der Anforderungen

Wiederverwendung

FORSOFT **ZEN**

- ✓ Blueprints und erfolgreiche Architekturen
- ✓ Design Patterns
- ✓ Standards

- ? Dokumentation von wiederverwendbaren Architekturen
- ? Klassifikation und Suche von wiederverwendbaren Architekturen
- ? Kombination wiederverwendbarer Architekturen

Qualitätsmerkmale und Entwurfsprinzipien

FORSOFT
ZEN

- ✓ Klare Entwurfsprinzipien: Modularität, Lose Kopplung sowie klare Schnittstellen und Abhängigkeiten
- ✓ Führen zu Qualitätsmerkmalen: Erweiterbarkeit, Flexibilität und Skalierbarkeit

- ? Vom Entwurfsprinzip zum Qualitätsmerkmal
- ? Bewertung von Architekturen anhand von Qualitätsmerkmalen und Entwurfsprinzipien
- ? Sicherstellung von Qualitätsmerkmalen beim Entwurf

BMW Group

Entwicklung und Einsatz von Musteranwendungsarchitekturen bei BMW

Markus Podolsky

BMW Group, FI-21

5.12.2000

Anwendungsarchitekturen bei BMW

Motivation von Musteranwendungsarchitekturen

Hohe Kosten für Betrieb

Schlechte Wartbarkeit, Erweiterbarkeit, Restrukturierbarkeit

Problem vorhandener Systeme

Kaum Wiederverwendung
Design, Code, ...

Schlechte Integrierbarkeit, Redundante Daten, Vermeidbare Batch-Läufe

Zu lange Entwicklungszeiten Time to System

Schlechte Portierbarkeit, Abhängigkeit von der Infrastruktur

Projektspezifische Anwendungsarchitekturen:
- indiviuell
- jedesmal neu

BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Was umfasst eine Anwendungsarchitektur?

Anwendungsarchitekturen

sind Leitfäden zur

- **Strukturierung fachlicher Anteile und deren Zusammenspiel**
- **Optimierung des Zusammenspiels der fachlichen Anteile und der technischen Infrastruktur**
- **Integration verschiedener Anwendungen**

Einsatz sowohl für Eigententwicklung als auch für Integration von Standardkomponenten

BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Arten von Architekturen

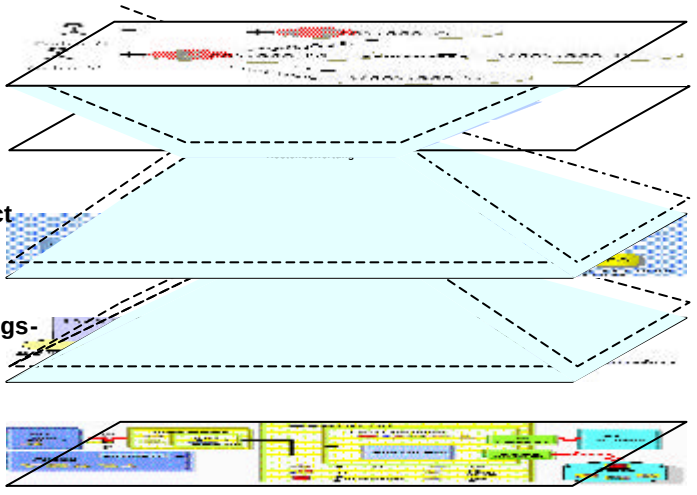
Fachliche Architektur

Subsystem-Architektur

Business Object Architektur

Implementierungs-Architektur

Infrastruktur Architektur



BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Schritte zur Anwendungsarchitektur

- **Grobarchitektur**
  - **Zerlegung in Subsysteme**
  - **Einbinden fertiger Subsysteme**
- **Feinarchitektur:**
  - **Zerlegung der Subsysteme in Business Objects**
  - **Verteilung der Business Objects auf logische Ebenen**
  - **Bildung von Komponenten**
  - **Umsetzung mit konkreten Technologien**
  - **Einbinden in die technische Infrastruktur**
  - **Integration weiterer Anwendungen**

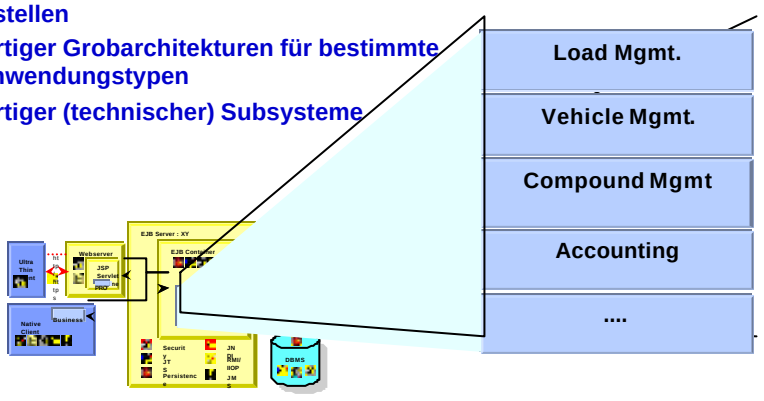
BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Grobarchitektur: Zerlegung in Subsysteme

- **Aufteilen der Use-Cases in fachliche bzw. technische Subsysteme (reichen über alle Tiers hinweg)**

➢ **Guidelines für die Zerlegung und das Zusammenspiel von Subsystemen**

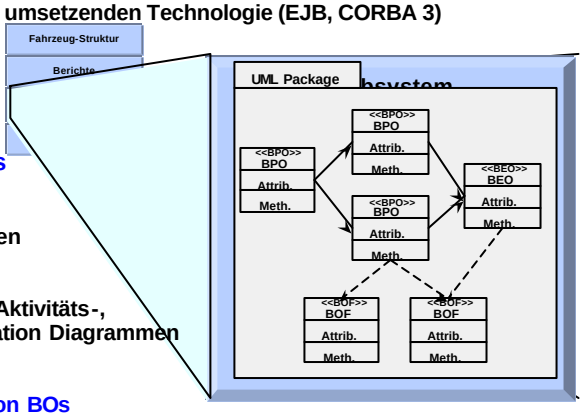
➢ **Bereitstellen**
- ➢ **fertiger Grobarchitekturen für bestimmte Anwendungstypen**
- ➢ **fertiger (technischer) Subsysteme**



BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Feinarchitektur: Zerlegung der Subsysteme in Business Objects

- **Was sind Business Objects (BOs) ?**
  - Fachlich getriebene logische Einheiten (z.B. Load, Vehicle, Checks, ...)
  - Einheit aus Daten, Logik, Regeln und Events, verschiedene Sichten
  - Granulare und schon im Design hoch wiederverwendbare Einheiten
  - Unabhängig von der umsetzenden Technologie (EJB, CORBA 3)



- **Beschreibung von BOs**
  - mittels UML
  - Struktur und Verhalten
  - Klassendiagramm
  - Zustandsdiagramm, Aktivitäts-, Sequence / Collaboration Diagrammen
- **Guidelines zur Bildung von BOs**

BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Arten von Business Objects

Business Objects:

Business Presentation Object

Business Process Object

Business Entity Object

Zusätzlich:

Client Object

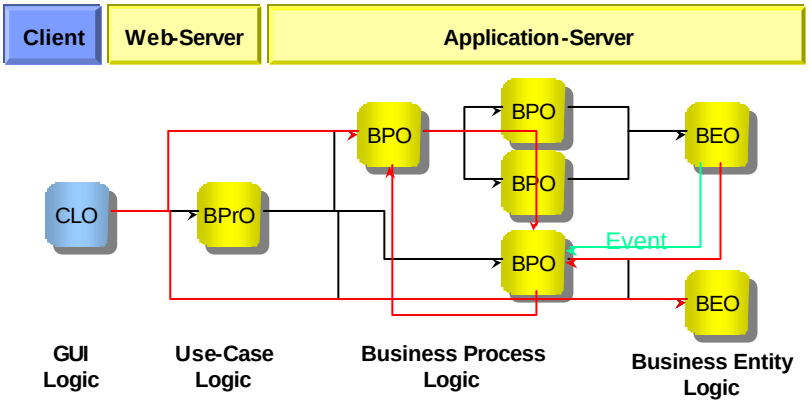
Business Object Facility

BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Feinarchitektur: Verteilung der BOs auf logische Ebenen

- **Identifizierung unterschiedlicher Typen von BOs und Verteilung über die logischen Ebenen**
- **Definition der unterschiedlichen BOs und deren Zusammenspiels**

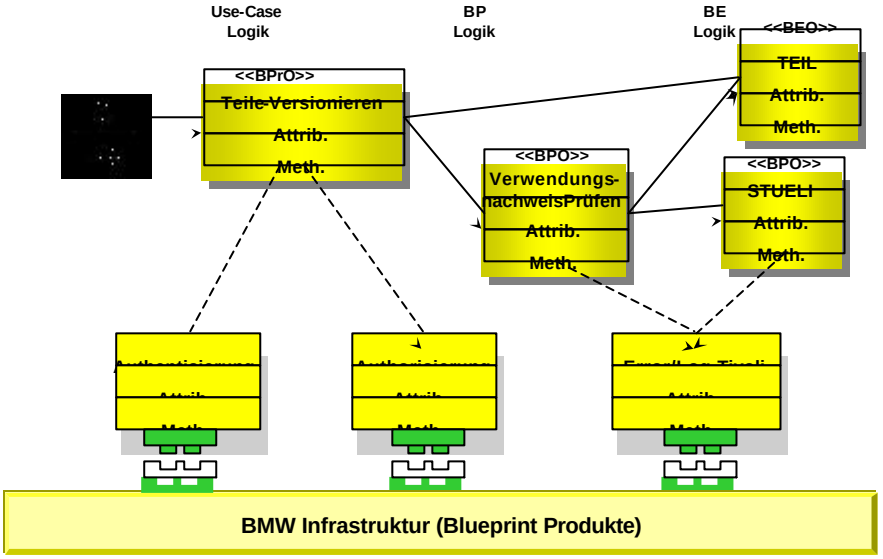
➢ **Guidelines für Bildung sowie Ein-/Verteilung von BOs**



BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Feinarchitektur: Einbinden in die technische Infrastruktur

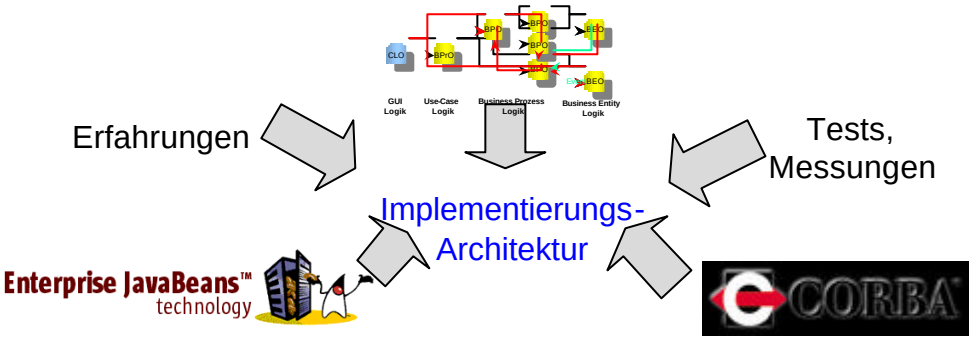
➢ **Guidelines (und Frameworks) zum Einbinden in die technische Infrastruktur**



Anwendungsarchitekturen bei BMW
Spezifische Aspekte

- **Reduzierung von Remote Communication, DB-Zugriffen**
  - Value Objects
  - Keine BEOs bei Queries mit großen Treffermengen
- **Möglichst geringer Ressourcenverbrauch**
  - BEO nicht ständig im Zugriff des Clients
  - BPrO gibt BPOs/BEOs wieder frei
  - Keine Abhängigkeiten zwischen BPrOs
- **Transaktionen**
  - Long Running Transactions vermeiden
  - Client stößt Transaktionen an
  - Service eines BPrO setzt Transaktionsklammern

Anwendungsarchitekturen bei BMW
Implementierungsarchitektur



Bei Abbildung auf konkrete Technologie:
Zusammenspiel mit konkreter Infrastruktur beachten!

- Application Server: Stand-alone vs. Cluster
- O/R-Mapping, Caching, DB-Replikation
- Realisierung von Business Events
- Integration bestehender Anwendungen

BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Ziel: Architekturzentrierter Entwurfsprozess

- **Zielsetzung**
  - Beschleunigung Time-To-System
  - Entwicklung gemäß Musteranwendungsarchitektur

- **Durchgängige Toolunterstützung**
  - BOs bereits in Modellierung kennzeichnen
    Stereotypen, tagged values
  - Generierung eines Implementierungsrahmens gemäß Architekturdefinition
  - Technologiespezifisches Ergebnis
    z.B. Interfaces, Klassen, Deployment Descriptors für EJB 1.1

BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Leitthema: Beschreibungstechnik

- **Eingesetzte Beschreibungstechniken**
  - Analyse: Use-Cases, Class + Activity Diagram,
    auch: Sequence Charts, Collaboration Diagram
  - Design: Detaillierung, zusätzlich StateTransition Diagram

- **Defizite von UML**
  - Fachliche Abläufe mit Activity Diagram + Use-Cases?
    - ⇨ High-level, Low-Level Abläufe
  - Infrastrukturarchitektur mit Deployment Diagram?
  - “Bedienungsanleitung” für Architektur?
    - ⇨ Scope of BO
    - ⇨ Transaktionsverhalten
    - ⇨ Ressourcenschonende Verarbeitung (Queries aus BPO, kein direkter Zugriff auf BEO vom Client, ...)

BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Leitthema: Methodik und Vorgehen

- **Methodisches Vorgehen**
  - ITPM, Vorgehensmodelle und Guidelines
  - Architekturzentrierte Entwicklung

- **Einbettung Architekturentwurf in Entwicklungsprozess**
  - Architekturprototypen vor Entwicklung
  - Proof of concept, Durchstich
  - Risiken möglichst früh erkennen
  - Entwicklung von Musteranwendungsarchitekturen ist projektübergreifend, iterativ

BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Leitthema: Wiederverwendung

- **Blueprint, Standardarchitekturen**
  - Blueprint für Produkte und Infrastrukturarchitektur
  - Geplant für Anwendungsarchitekturen
    Business Objects, Native Client, Web Client, Embedded Workflow,
    Authentifizierung / Autorisierung, Error Handling & Logging
    Application Integration (CICS, IMS, SAP) mit XML, JMS, Web Services, …
    Integration von Standardkomponenten (PDM, …)
  - Ziel: Verbindlicher Einsatz

- **Ebene der Wiederverwendung**
  - Code
  - Design
  - Spezifikation: Konzept BO in frühen Phasen der Modellierung

BMW Group
M. Podolsky
FI-21
05.12.2000

Anwendungsarchitekturen bei BMW
Leitthema: Qualitätsmerkmale und Entwurfsprinzipien

- **Qualitätsmerkmale**
  1. Performance, Skalierbarkeit, Wiederverwendbarkeit
  2. Anpassbarkeit, Erweiterbarkeit, Wartbarkeit

- **Entwurfsprinzipien**
  - lose Kopplung
  - Modularität
  - Aufsetzen auf Standardschnittstellen, Austauschbarkeit

Architektur für die Kopplung von IT - Systemen mit Simulationssystemen auf Basis HLA

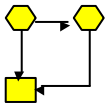
Dr. Andreas Dabelstein
ESG GmbH
Einsteinstraße 174
81657 München
Tel.: +49 89 9216 - 2646
Fax: +49 89 9216 - 2732
E-Mail:adabelst@esg-gmbh.de

Inhaltsübersicht

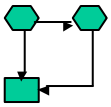
- **Einleitung & HLA Anwendungen**
- **Führungsinformations- und Simulationssysteme**
- **Kopplung mit High Level Architecture**
- **Ergebnisse**

Verteilte IT-Systeme und Simulationssysteme
Þ High Level Architecture

OR Modelle



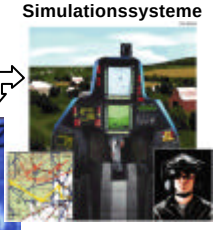
Konstruktive Simulations systeme



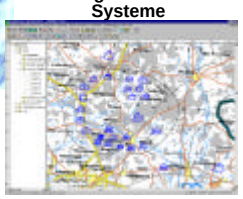
- Entwicklungsbegleitende Simulation
- Entscheidungsunterstützung

- Analyse und Planung
- Ausbildungs- und Trainingssysteme

Virtuelle Simulationssysteme



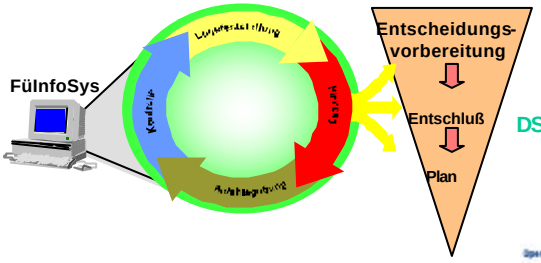
Führungsinformations Systeme



Herausforderungen

- Interoperabilität
- Wiederverwendbarkeit
- Standardisierung
- hohe Flexibilität
- Kopplung von Realzeit- mit Simulationszeit-Ereignissen

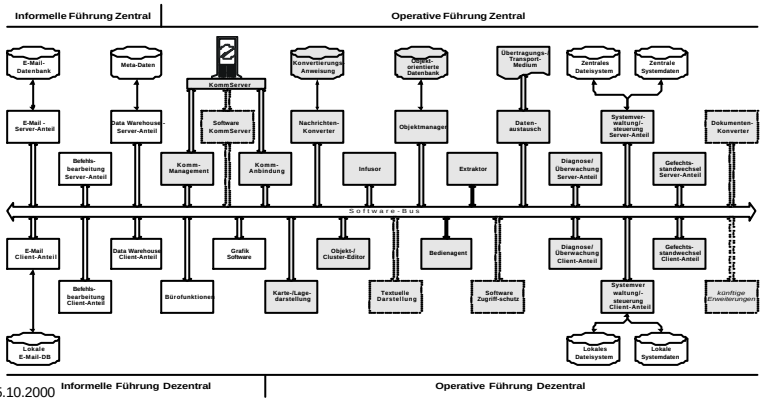
Führungsprozess und Führungsinformationssysteme



- bewertet komplexe Zusammenhänge
- reduziert Streßsituationen und hohe physische Belastungen
- entlastet von Routineaufgaben
- unterstützt bei Abwesenheit von Experten

FüInfoSys HEROS

- OO Architektur und OO Datenbankmanagementsystem
- Nutzung von kommerziellen SW Produkten
- Middleware CORBA

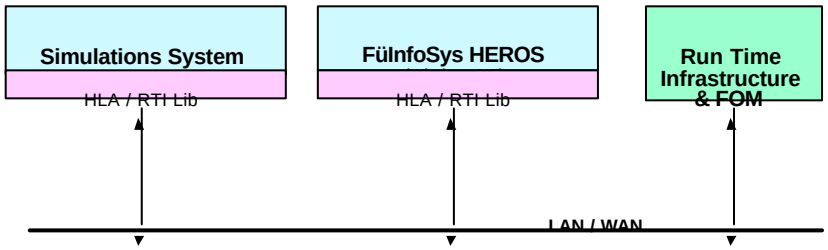


High Level Architecture

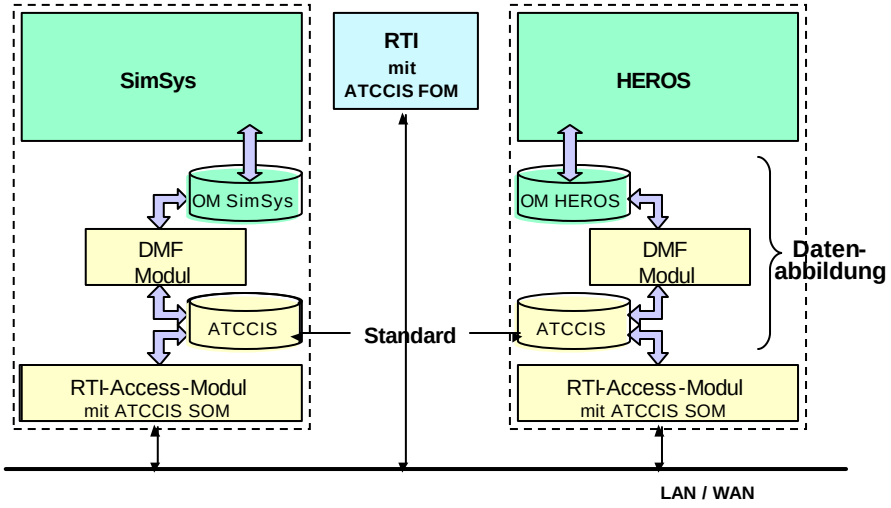
- **Ziel: Kopplung der zu integrierenden Systeme und Module**
- Verbindliche Vorgabe des US DoD
- Teilnehmer (Federates) schließen sich zu einem Verbund (Federation) zusammen.
- **HLA spezifiziert durch**:
  - **Regelwerk mit 10 Regeln** zur Beschreibung des Verhaltens der Federates und der Federation. **IEEE 1516 Framework and Rules**
  - Interface Spezifikation unterteilt in **sechs Managementbereiche** **IEEE 1516.1 Federate Interface Specification**
  - **Object Model Template (OMT),** legt das Format für die zwischen den Federates auszutauschenden Objekte fest **IEEE 1516.2 Object Model Template OMT Specification**

Eigenschaften der HLA

- Was ein Federate zur Federation beitragen kann, wird in der Objektsichtweise der Federates (Simulation Object Model - SOM) und der der Federation (Federate Object Model - FOM) festgeschrieben.
- Zur Laufzeit treten die Federates über die **Run Time Infrastructure (RTI)** in Kontakt

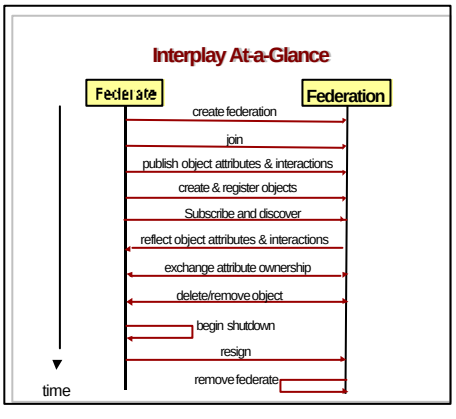


Gesamt Architektur



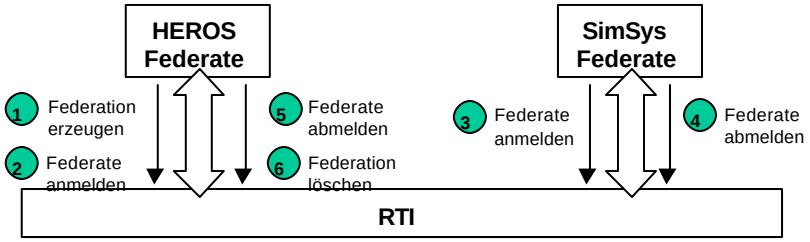
6 HLA Managementbereiche

1. **Federation Management**
   (wer nimmt teil ?)
2. **Declaration Management**
   (wer gibt ? Wer nimmt ?)
3. **Object Management**
   (was gibt es neues ? Wer bekommt es ?)
4. **Ownership Management**
   (Wer ist Besitzer eines Objekts ?)
5. **Time Management**
   (Uhrenvergleich)
6. **Data Distribution Management**
   (wie kommen die Daten effektiv vom Sender zum Empfänger ?)



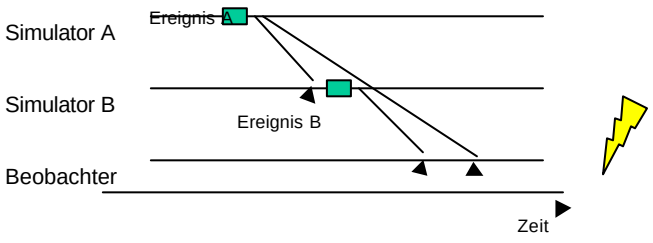
Federation Management

- Start eines Simulationsslaufes
  - Services: Federation erzeugen, Federates anmelden
- Ende eines Simulationslaufes
  - Services: Federates abmelden, Federation löschen



Time Management

- **Ereignisse treten in logischer Reihenfolge auf**

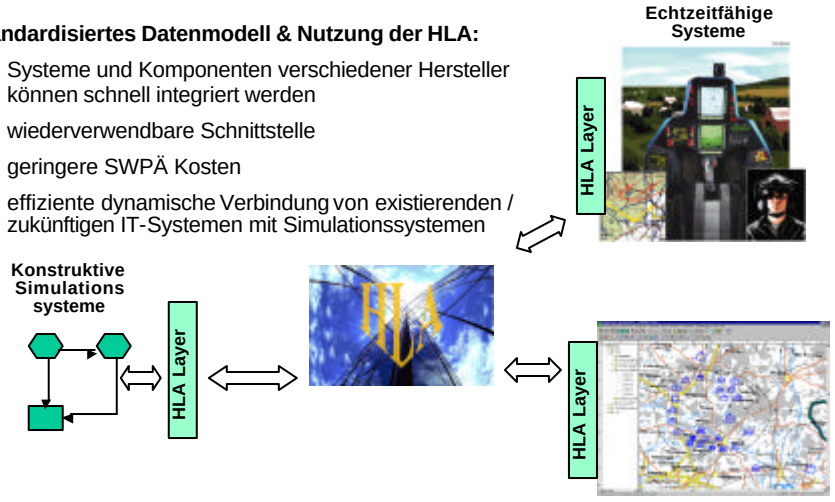


- **Servicefunktionen (Lookahead, Time Regulation, Time Constrained)**

Zusammenfassung

Standardisiertes Datenmodell & Nutzung der HLA:

- Systeme und Komponenten verschiedener Hersteller können schnell integriert werden
- wiederverwendbare Schnittstelle
- geringere SWPÄ Kosten
- effiziente dynamische Verbindung von existierenden / zukünftigen IT-Systemen mit Simulationssystemen



Beschreibungstechnik

- Welche Beschreibungstechniken werden beim Entwurf eingesetzt und was wird beschrieben ?

  Die Beschreibungstechniken sind IEEE Standards:

IEEE 1516	Framework and Rules
IEEE 1516.1	Federate Interface Specification
IEEE 1516.2	**Object Model Template OMT Specification**

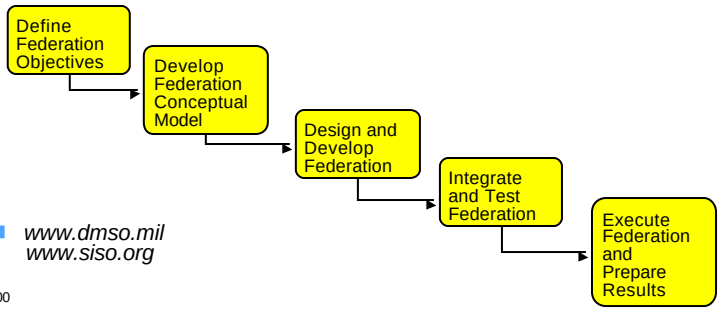
- In wie weit wird die UML als Architekturbeschreibungssprache verwendet und welche Defizite tauchen dabei auf ?

  UML wird z.Z. **nicht** durch HLA unterstützt. Das Datenmodell wird durch die OMT, d.h. ein hierarchisches DM mit einfacher Vererbung beschrieben.

Methodik und Vorgehen

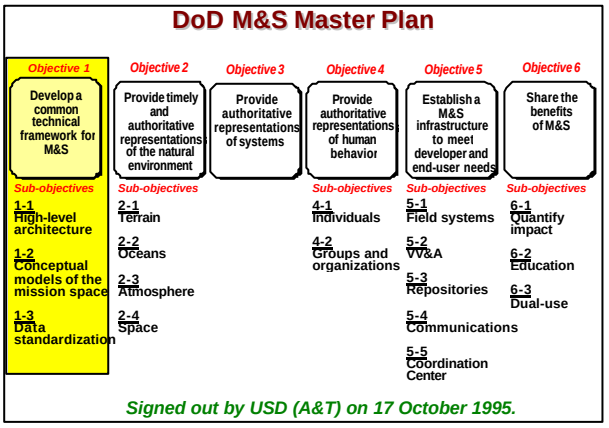
- Gibt es ein methodisches Vorgehen beim Entwurf und wie sieht es aus ?
- Wie weit ist der Architekturentwurf in den restlichen Entwicklungsprozess eingebettet ?

  Als Entwicklungsprozess für HLA ist der **Federation Execution & Development Process (FEDEP)** etabliert.



- *www.dmso.mil*
  www.siso.org

Modelling & Simulation Master Plan



Wiederverwendung

- Gibt es unternehmensweite Blueprints und Standardarchitekturen und in wie weit fliessen diese beim Entwurf mit ein ?
- Auf welcher Ebene wird Wiederverwendung von Architekturen betrieben (Code-, Design-, Spezifikationsebene) ?

  Wiederverwendung wird auf Codeebene sichergestellt. Entscheidend ist die semantische Interoperabilität der zu verbindenden Systeme, d.h. das **FOM Design**.

Wiederverwendung

- Gibt es unternehmensweite Blueprints und Standardarchitekturen und in wie weit fliessen diese beim Entwurf mit ein ?
- Auf welcher Ebene wird Wiederverwendung von Architekturen betrieben (Code-, Design-, Spezifikationsebene) ?

  Wiederverwendung wird auf Codeebene sichergestellt. Entscheidend ist die semantische Interoperabilität der zu verbindenden Systeme, d.h. das **FOM Design**.

Qualitätsmerkmale und Entwurfsprinzipien

- Welche Qualitätsmerkmale werden beim Entwurf angestrebt und wie ist die Priorisierung (Wartbarkeit, **Anpassbarkeit, Wiederverwendung**,...)

- Welche Entwurfsprinzipien werden beim Entwurf eingesetzt, um bestimmte Qualitätsmerkmale zu erreichen (**Modularität, lose Kopplung**, klare Abhängigkeit, ...)

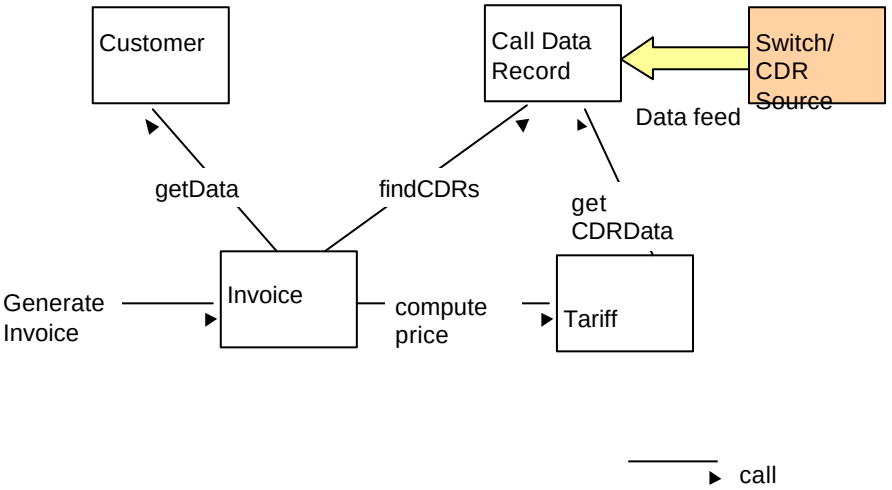
  Aufgrund der hohen Investitionen in Simulations- und IT-Systeme hat bei HLA **WIEDERVERWENDUNG** höchste Priorität. Darüber hinaus ist die Architektur flexibel auf die Anforderungen der Simulation ausgelegt. Hohe Datenübertragungsraten sind wünschenswert.

Gedanken zur Architektur von Informationsystemen

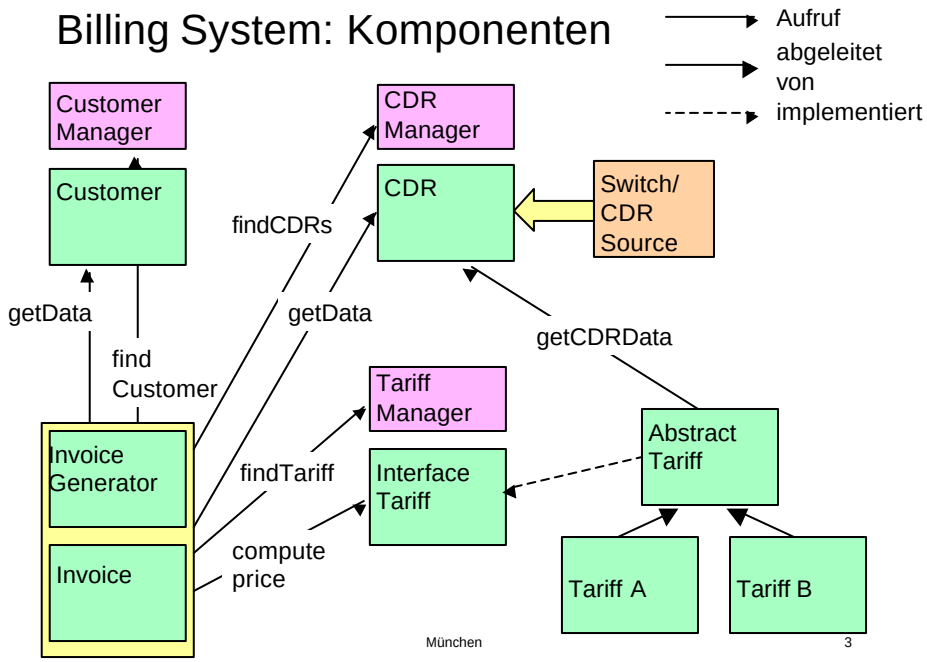
- Denken in Komponenten
- Schnittstellen
- Definierte Abhängigkeiten
- Variabilitätsanalyse
- Generische Programmierung
- Konsequenzen für den Entwicklungsprozeß

J. Siedersleben
Dezember2000
sd&m Münchem

Thinking in Components: Billing System



Billing System: Komponenten



Schnittstellen: Vision

1. Exakte Definition der Semantik (mathematische Hilfsmittel)
2. Qualitative Zertifizierung von Interface-Implementierungen gegen die Interface-Definition
3. Quantitative Zertifizierung von Interface-Implementierungen auf der Basis von Lasttests (wieviele PS hat unsere Implementierung in definierter Umgebung)

Vision

Zertifizierte Interfaces als berechenbare, belastbare Träger von Software-Architektur

Maximale und minimale Schnittstellen

- Jedes Produkt, jede Norm bietet maximale Interfaces:
  die Vereinigungsmenge aller vorhersehbarer Anforderungen (z.B. Swing)
- Aus Sicht der Anwendung interessieren mich minimale Interfaces: Was muß ich MINDESTENS sagen, damit mein Partner arbeiten kann.
- Die IQModel-Interfaces sind minimal:

```
public interface IQModel
{
    String getName();
    boolean isEnabled();
}

public interface IQFormModel extends IQModel
{
    Vector getValues();
    Vector getFields();
    Vector getChecks();
}
```

```
public interface IQFieldModel extends IQModel
{
    boolean isEditable();
    boolean isOk( String str );
    void setValue( String str );
    String getValue();
}
```


Definierte Abhängigkeiten

Software kann sein

- 0 bestimmt von gar nichts (Behälter, Strings)
- A bestimmt von der Anwendung (Kunde, Konto, Buchung)
- T bestimmt von mindestens einem technischen API (z.B. Datenverwaltung)
- AT bestimmt von der Anwendung und mindestens einem technischen API.
- R Trafo-Software (milde Art von AT)

Blutgruppen A + 0 = A; T + 0 = T; A + T = AT

Bewertung

- 0 ideal wiederverwendbar, für sich alleine nutzlos
- A dafür werden wir bezahlt
- T muß sein
- AT um jeden Preis zu vermeiden (bis auf R)
- R kann generiert werden

Variabilitätsanalyse

Klassifizierung von Änderungen

- R-Änderungen betreffen externe Repräsentation fachlicher Objekte
- T-Änderungen betreffen die Technik
- A-Änderungen betreffen die Anwendung.

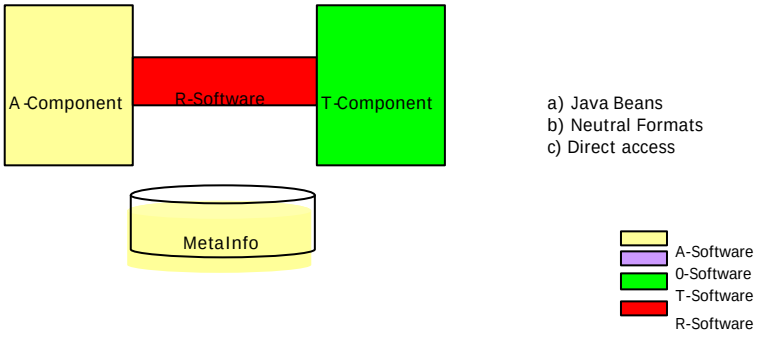
Ziel (Quasar)

X-Änderungen betreffen nur X-Software (X = R, T, A)

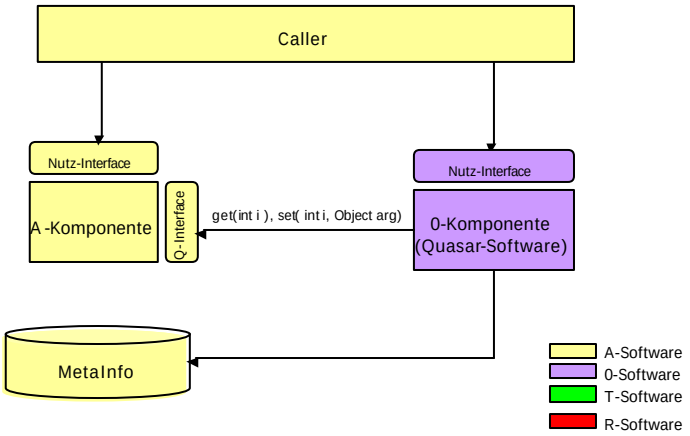
Generic Programming

A- and 0/T-Software communicate by means of

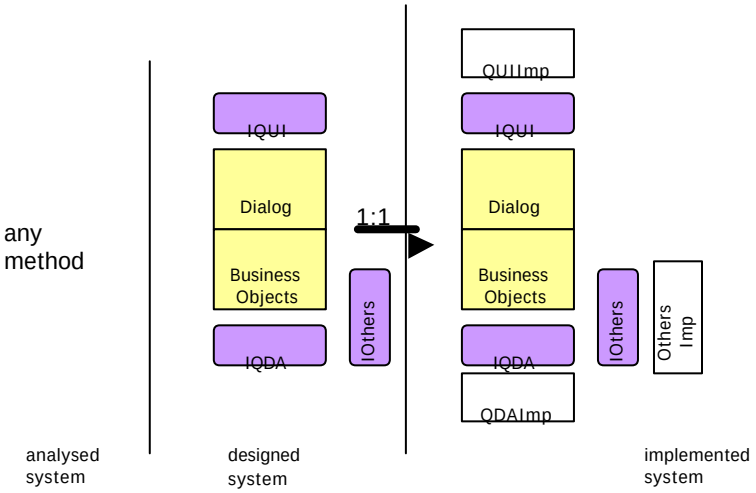
R-Software and (optionally) meta-information



Generische Programmierung: Direkter Zugriff



Quasar: Development Process



I Beschreibungstechnik im Entwurf

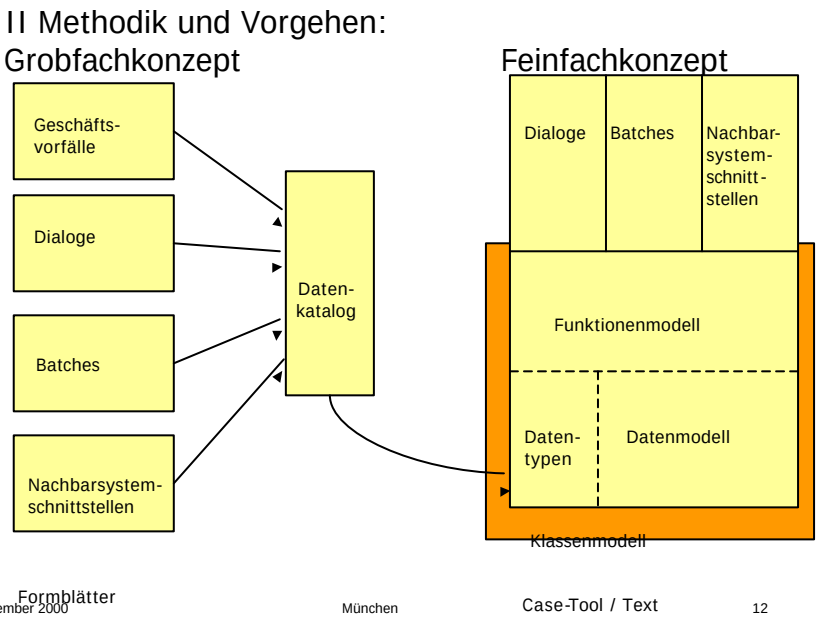
Die eingesetzten Beschreibungstechniken sind Texte und Bilder.

frei gestaltet/ informal	halbformal	formal

UML-Defizite

- wenig/kein Zusammenhang der verschiedenen Diagrammtypen
- schwache Integration Text/Bild
- keine Unterscheidung Datentyp/autonome Klasse
- keine Unterstützung bei den Standardproblemen (z.B. Historie, Mandantenfähigkeit)
- keine Unterstützung bei der GUI-Modellierung
- OCL ungeeignet
- in der Praxis oft: Klassendiagramm steht im Vordergrund; Komponenten und Schnittstellen kommen zu kurz

II Methodik und Vorgehen:



III Wiederverwendung

- **SINUS**: sd&m-Wiederverwendungsarchiv
  -- enthält Code und Konzepte
  -- wird wenig verwendet (schlechte Doku, zu komplizierte/zu spezifische Lösungen)
- Einige wenige rühmliche Ausnahmen (DB-Loader)
- **Quasar**: Definition von wiederverwendbaren Konzepten. Das heißt
  -- Idee
  -- Komponenten und Schnittstellen (sprachunabhängig)
  -- Beispielimplementierungen (Java, C++)
- **Bestandteile** von Quasar:
  -- QDI (Quasar Database Interface)
  -- QUI (Quasar User Interface)
  -- QMI (Quasar Middleware Interface)
  -- QDT (Quasar Data Typpes)
  -- QUT (Quasar Utilities)
  -- QAP (Quasar Applications)

IV Qualitätsmerkmale und Entwurfsprinzipien

- Denken in Komponenten, nicht in Frameworks
- Denken in Interfaces: Der Aufrufer ist wirklich unabhängig von der Implementierung
- Variabilitätsanalyse
- Trennung der Zuständigkeiten: 0/A/T
- Minimale/maximale Schnittstellen
- So einfach wie möglich, aber nicht einfacher.

> Ein Entwurf ist nicht dann fertig, wenn Du nichts mehr hinzufügen kannst, sondern dann, wenn Du nichts mehr weglassen kannst.
>
> Antoine de Saint-Exupery

Entwicklung von Frameworks bei iteratec

iteratec

Geschäftsinhalt

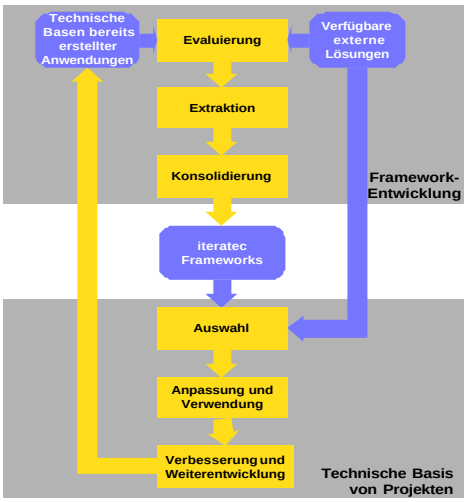
- Entwicklung individueller betrieblicher Management- und Informations-Systeme
- Unterstützung der DV-Abteilung in Fragen der IT-Strategie und im Projektmanagement
- Kernkompetenzen:
  - iteratives Vorgehensmodell
  - Entwicklung von IT-Architekturen
  - OO-Analyse und Design
  - Migration von Altverfahren
  - Management großer Projekte

iteratec

Entwicklung von IT-Architekturen E-Business-Anwendungen

- **Erfolgskriterien**
  - Anforderungen der Anwender
  - Strategische Erfolgsfaktoren des Unternehmens
- **Aspekte**
  - Benutzerkreis
  - Sicherheitsanforderungen
  - Systemanforderungen
  - Time to market
  - Neue Technologien
  - Skills und Ressourcen
  - ...

Entwicklung von IT-Architekturen

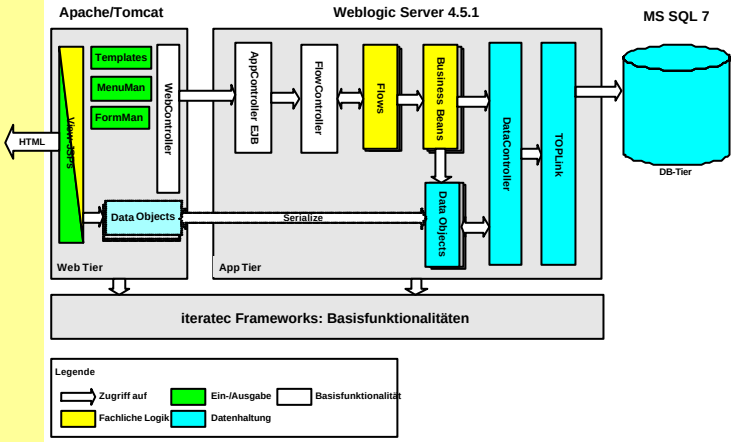


Entwicklung von IT-Architekturen

- **Anforderungen an Architekturen**
  - Angemessenheit
  - Einfachheit
  - Verständlichkeit
  - Erweiterbarkeit
  - Komponentenbildung
- **Inkrementell iterativer Prozess**
- **Design**
  - Design erfolgt kontinuierlich
  - Lernen während der Entwicklung
  - Refactoring
- **Selbstüberprüfende automatisierte Tests**
- **Dokumentation**

iteratec

Technische Architektur E-Business- Anwendung (J2EE)



iteratec

Technische Architektur Designentscheidungen

- **Konfiguration und Parametrisierung**
  - Reflection Pattern
  - Meta-Daten
  - Generierung
- **Kommunikation zwischen den Schichten**
  - schmale Schnittstelle
  - lose Kopplung
  - Command Pattern
- **Dialogsteuerung**
  - Zustandsautomat
  - fachlicher Status <-> Dialogstatus
- **Datenzugriffsschicht**
  - objektrelationales Mapping
  - Persistenz
  - Transaktionen

iteratec

Beschreibungstechniken beim Entwurf

Ziel: verständlicher und intuitiver Entwurf als Basis für eine effiziente zwischenmenschliche Kommunikation

Intuitive halbformale Diagramme
- Veranschaulichung von Zusammenhängen
- Beschreibung von Architektur und Informationsfluß

UML-Diagramme (Klassendiagramme, Sequenzdiagramme)
- Dokumentation von Designentscheidungen
- Verwendung bei größerer Detaillierung

Defizite von UML
- Fülle von Diagrammtypen - bewußte Selektion notwendig
- Notation suboptimal (teilweise unübersichtlich)
- Beschreibung des Informationsflusses nicht möglich
- Unzureichende Werkzeuguntersützung

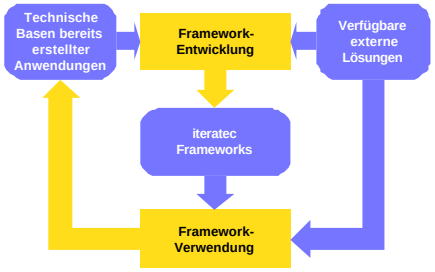
iteratec

Methodik und Vorgehensweise

- **iteratec Vorgehensmodell**
  - Prototypen
  - Leitlinien, Checklisten...
  - Projektorganisation (z.B. Technische Basis Team)
  - Designmeetings und Reviews in Projektplanung
- **Design**
  - Design erfolgt kontinuierlich
  - Designmeetings
  - Reviews
  - Lernen während der Entwicklung
  - Refactoring

Wiederverwendung

- **iteratec Frameworks**
  - Referenzarchitekturen für E-Business Systeme
  - Basisfunktionalitäten und Basiskomponenten
  - Leitlinien, Checklisten, Namenskonventionen, ...
  - Allgemeine Designdokumente
- **Wiederverwendung bei iteratec**
  - Trennung zwischen Projekt - Framework-Entwicklung
  - iteratec Frameworks stehen für Projekte zur Verfügung
  - Kein Muß für Wiederverwendung in Projekten
  - Aktives Wissensmanagement
- **Vorgehen bei iteratec**



Qualitätsmerkmale und Entwurfsprinzipien

- **Qualitätsmerkmale generell:**
  Angemessenheit, Einfachheit, Verständlichkeit, Komponentenbildung, Flexibilität, Erweiterbarkeit, Portabilität, Robustheit, Nutzungskomplexität
  Im Projektkontext anwendungsabhängig zu priorisieren
- **Entwurfsprinzipien**
  - Einsatz von Pattern
  - Komponentenbildung mit klar definierten Schnittstellen
  - Lose Kopplung zur Anbindung von Legacy-Systemen
  - Klare Zuordnung von Verantwortlichkeiten
  - Schichtenarchitektur

iteratec

Komponentenbasierte Softwarearchitektur für CAM Systeme

Dr. Wolfgang Daxwanger
SEKAS *GmbH*
München
daxwanger@sekas.de

PROQUIS
Components for CAM

Einführender Gedanke

Die Software-Industrie versucht jetzt nachzuholen, was die Hardware-Industrie seit langen Jahren erfolgreich vorlebt:

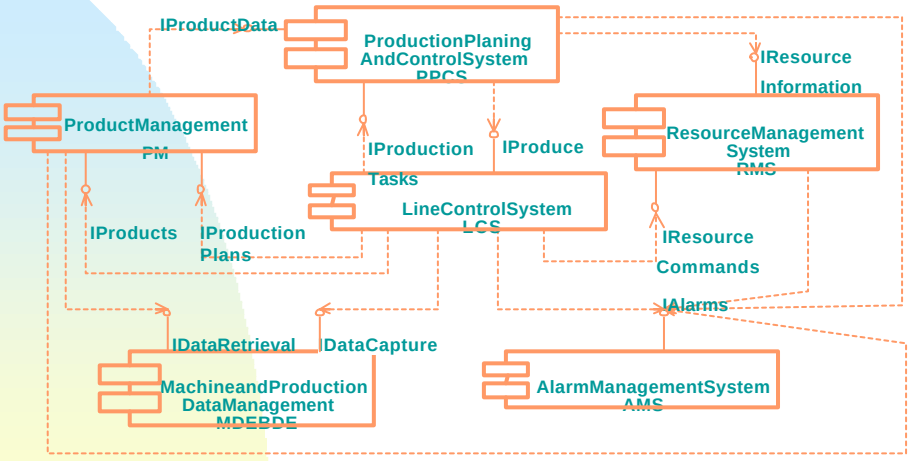
- Definition von Standards für die Interoperabilität von Komponenten.
- Integration von Komponenten in Module auf verschiedenen Komplexitätsebenen.

Einführung

- Ein Hauptbetätigungsfeld der SEKAS GmbH ist die Entwicklung von Softwaresystemen und Softwarekomponenten für die Bereiche
  - Automatisierungstechnik (CAM-Bereich)
  - Kommunikationstechnik
  - Testsysteme
- Im CAM-Bereich variieren Projektanforderungen stark aufgrund
  - unterschiedlichster Systemumgebungen
  - unterschiedlicher Automatisierungsgrade
  - unterschiedlicher Zielsetzungen der Kunden

Dezember 2000 Komponentenbasierte Softwarearchitektur für CAM Systeme, © SEKAS GmbH 3

Systemarchitektur für CAM-Systeme



Dezember 2000 Komponentenbasierte Softwarearchitektur für CAM Systeme, © SEKAS GmbH 4

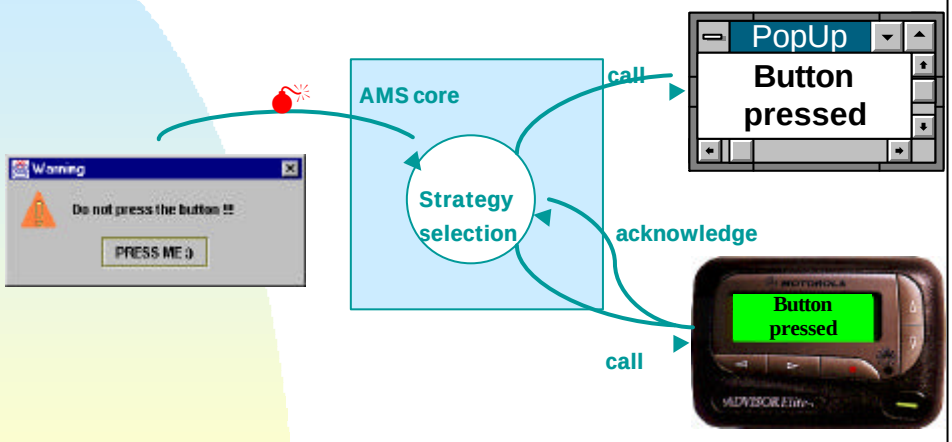
Alarm Management System-Anforderungen

- Kein Verlust von Alarminformation
- Asynchrone Alarmbehandlung
- Frei konfigurierbare Alarmbehandlung mit Eskalationsstrategie
- Plattformunabhängigkeit
- Transparente API
- Kontextsensitive Alarmklassifizierung

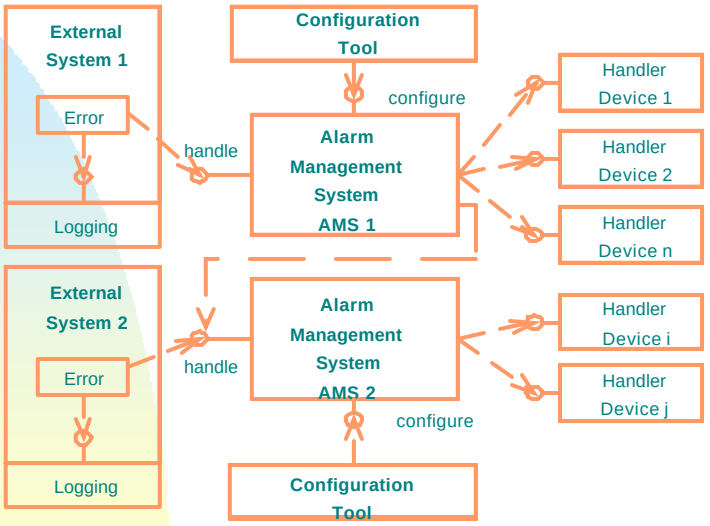
Entwurfsentscheidungen

- Einsatz von CORBA als Middleware
- Alarmklassenhierarchie
- Strategiemuster

Alarm Management System Anwendungsbeispiel



Alarm Management System Architektur

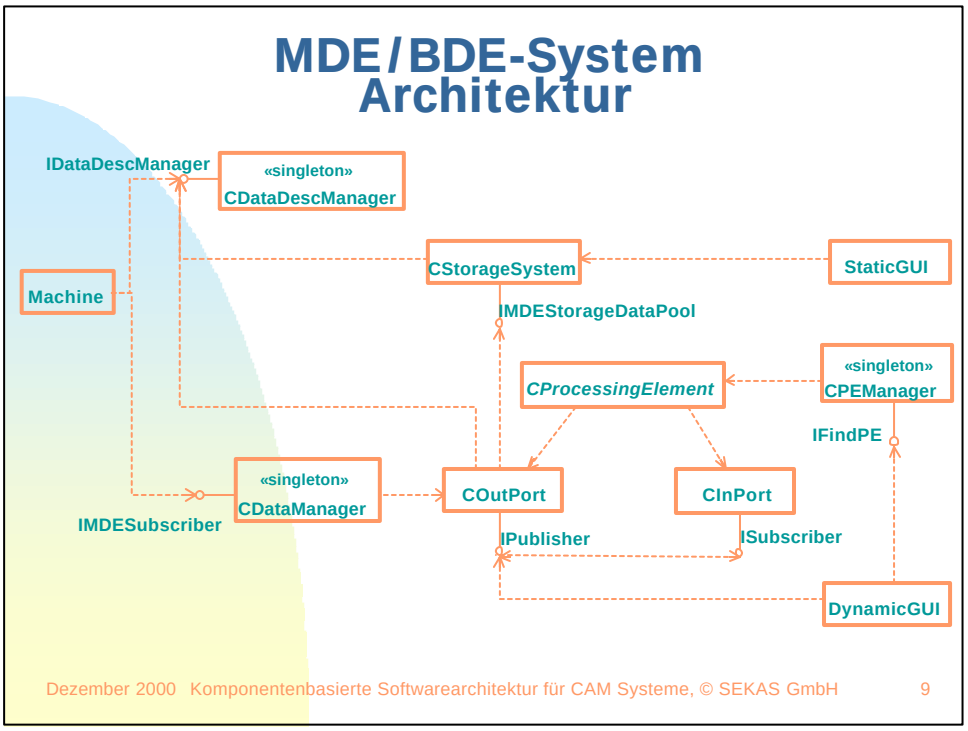


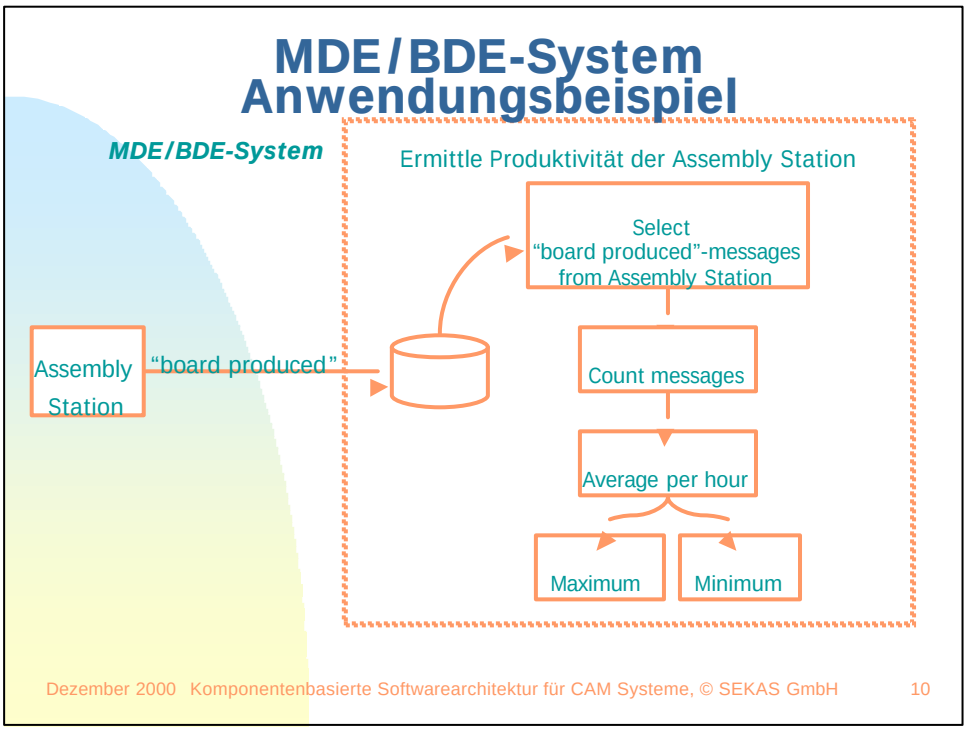
MDE/BDE-System Anforderungen

- Nichtblockierende Weiterleitung von Daten
- Kein Datenverlust
- Erweiterbares Datentypenmodell
- Datentypsicherheit
- Unterschiedliche Persistenz-Stufen
- Einfache Erweiterbarkeit
- Konfigurierbarkeit

Entwurfsentscheidungen

- Datenbeschreibung und Datentypbeschreibung mittels XML
- Netzwerk einfacher aber effizienter und zuverlässiger Verarbeitungselemente


MDE/BDE-System
Architektur
IDataDescManager
«singleton»
CDataDescManager
CStorageSystem
StaticGUI
Machine
IMDEStorageDataPool
«singleton»
CPEManager
CProcessingElement
IFindPE
«singleton»
CDataManager
COutPort
CInPort
IMDESubscriber
IPublisher
ISubscriber
DynamicGUI
Dezember 2000 Komponentenbasierte Softwarearchitektur für CAM Systeme, © SEKAS GmbH 9


MDE/BDE-System
Anwendungsbeispiel
MDE/BDE-System
Ermittle Produktivität der Assembly Station
Select
“board produced”-messages
from Assembly Station
Assembly
Station
“board produced”
Count messages
Average per hour
Maximum
Minimum
Dezember 2000 Komponentenbasierte Softwarearchitektur für CAM Systeme, © SEKAS GmbH 10

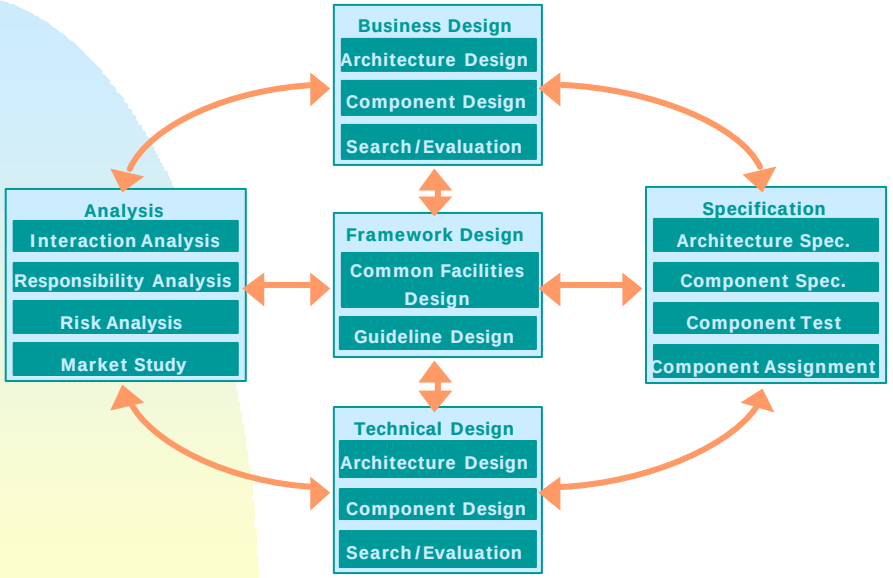
Beschreibungstechniken

- Komponentenmodell
  - Komponentendiagramme (grafische Darstellung der Relationen)
  - textuelle Beschreibung der Zuständigkeit
- Schnittstellen
  - textuelle Beschreibung
  - Schnittstellen-Klassenmodellierung
- Datenstruktur
  - E/R
  - Klassenmodellierung
- Dynamisches Verhalten
  - Sequencecharts/ Kollaborationsdiagramme
  - Aktivitätsdiagramme/ Zustandsautomaten
  - ereignisgesteuerte Prozessketten

Beschreibungstechniken Defizite mit UML

- Keine zeitlichen Randbedingungen einbringbar
- Trägt hauptsächlich den Gedanken der technischen Komponente (dll, exe, jar)
  - Schnittstelle vs. implementierende Komponente in Sequenzdiagrammen
- Unterstützer Sprachumfang der CASE Tools entspricht oft nicht der Spezifikation

Methodik und Vorgehen



Dezember 2000 Komponentenbasierte Softwarearchitektur für CAM Systeme, © SEKAS GmbH 13

Wiederverwendung

- (Noch) keine Standardarchitektur
- Aus der Verwendung der Komponenten soll schrittweise eine Standardarchitektur entstehen
- Wiederverwendung findet statt im Bereich
  - Komponenten
    - Beispiele: AMS und MDE/BDE
  - Code
    - Beispiel: Ankopplung des AMS
  - Design
    - Beispiel: Alarmhierarchien

Dezember 2000 Komponentenbasierte Softwarearchitektur für CAM Systeme, © SEKAS GmbH 14

Qualitätsmerkmale und Entwurfsprinzipien

Hauptmerkmale
- Funktionalität
- Zuverlässigkeit

Projektabhängige Merkmale
- Benutzbarkeit
- Übertragbarkeit

Untergeordnete Merkmale
- Effizienz
- Änderbarkeit

Unterstützende Entwurfsprinzipien
- Modularität
  - Unabhängigkeit
  - Abgrenzung
- lose Kopplung
- Kapselung der Komplexität

Flexible Unternehmensanwendungen

Integration eines EJB-Servers mit einer Workflow-Engine

Klaus Bergner
Olav Rabe

4SOFT

Architektur-workshop

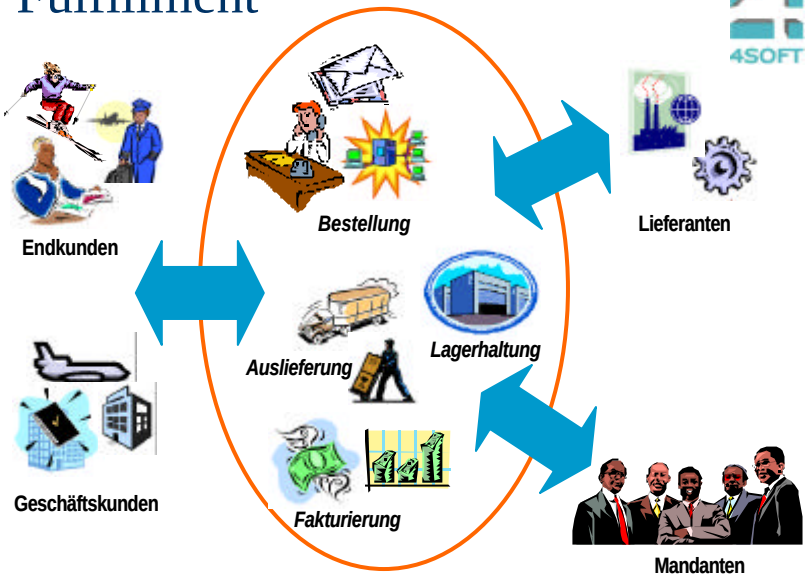
5. Dezember 2000

Inhalt

4SOFT

- Einführung
  - Anwendungsumfeld
  - Ziele und Motivation
- Vom Geschäftsmodell zum lauffähigen System
  - Modellierungskonzepte
  - Sicht des Benutzers
- Technik
  - Architektur
  - Implementierung
  - Worklets
- Reflektion
  - Beschreibungstechnik
  - Methodik und Vorgehen
  - Wiederverwendung
  - Qualitätsmerkmale und Entwurfsprinzipien

Fulfillment



Hauptziele für Dohmen

- Überlebe den steigenden Wettbewerb
  - Biete den Mandanten auf sie zugeschnittene Dienstleistungen
  - Verkürze die Time-to-Market für neue Dienstleistungen
- Bewältige das Wachstum
  - Verkürze die Zeit für die Umsetzung neuer Mandanten
  - Vermindere die Reibungsverluste beim Arbeiten und bei der Einarbeitung neuer Mitarbeiter

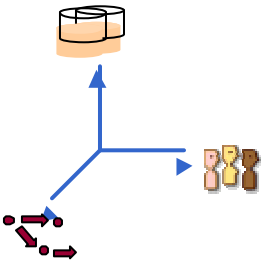
Viele traditionelle Warenwirtschaftssysteme bieten zwar sämtliche nötige Funktionalität, es mangelt ihnen jedoch an der nötigen Mandantenfähigkeit und Flexibilität.

Wie erreicht man Flexibilität?

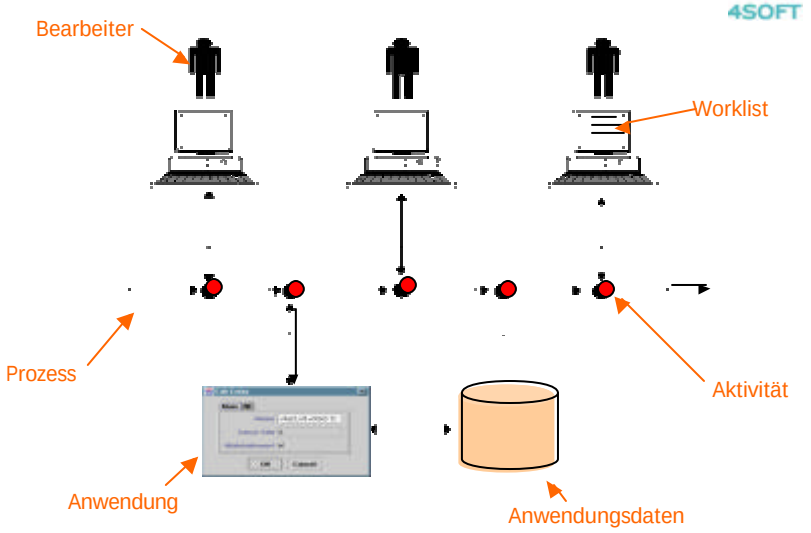
- Trennung von Fachlichkeit und Technik
  - Realisierung der fachlichen Daten und Abläufe erfordert möglichst wenig Wissen über die technische Infrastruktur
- Modellierung statt Programmierung
  - Änderungen bei der Fachlichkeit können ohne Programmierung geschehen
- Wiederverwendbare Komponenten
  - System besteht aus Bausteinen, die ausgetauscht, konfiguriert und angepasst werden können
- Flexible Architektur
  - Zusammenspiel und Schnittstellen der Bausteine sind klar definiert

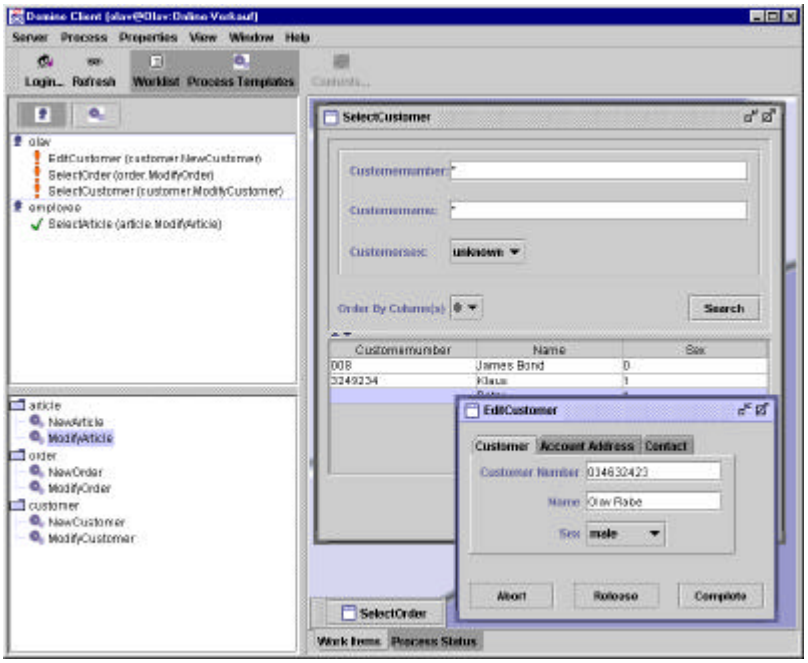
Modellierung der Fachlichkeit

- Geschäftsobjekte
  - UML-Klassendiagramme
  - Umsetzung durch EJB Entity Beans auf EJB Server
- Geschäftsprozesse
  - UML-Aktivitätsdiagramme
  - Umsetzung durch Workflow-Engine mit Prozess-Repository, EJB Session Beans für Aktivitäten
- Organisation
  - UML-Use-Case-Diagramme
  - Umsetzung durch Verzeichnis-dienst (z.B. LDAP, Active Directory, NDS)



Workflow-gesteuertes Arbeiten





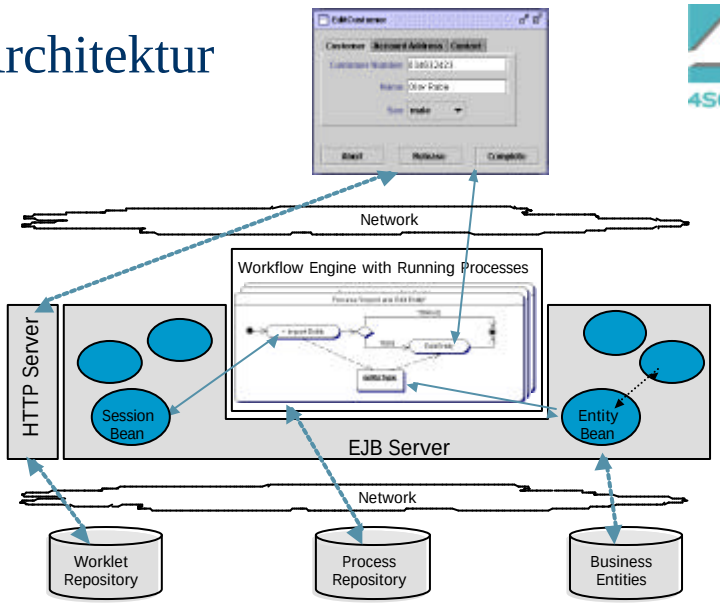
Ansatz bei der Architektur

- Anforderungen an die Architektur
  - Komponentenorientierung
  - Verwendung moderner Techniken
  - Klare Schnittstellen für Anwendungsprogrammierer
  - Möglichst weitgehende Verwendung von Standards
  - Vielfältige Möglichkeiten bei der Verteilung vorsehen
- Gewählter Ansatz: Dreischichtenarchitektur
  - Datenhaltungsschicht (Relationales Datenbank-Managementsystem Oracle bzw. DB/2)
  - Anwendungsschicht (EJB-1.1 Application Server und eingebettete Workflow Engine Verve)
  - Präsentationsschicht (Native Java Clients über RMI/CORBA)

Abbildung der Fachlichkeit

- Geschäftsobjekte
  - Abbildung von Geschäftsobjekten und Features gemäß dem CMP-Ansatz (Container-Managed Persistence) auf Entity EJBs
  - Realisierung von beliebigen Assoziationen über ein selbstgeschriebenes Framework mittels BMP (Bean-Managed Persistence)
- Geschäftsprozesse
  - Abbildung von Prozessen auf Prozess-Objekte der Workflow-Engine
  - Abbildung von autonomen Aktivitäten auf EJB Session Beans, die automatisch vom EJB Server ausgeführt werden
  - Abbildung manueller Aktivitäten auf „Worklets“, die auf dem Client ablaufen und mit der Anwendungsschicht kommunizieren
- Organisationsmodell derzeit nur rudimentär realisiert

Architektur

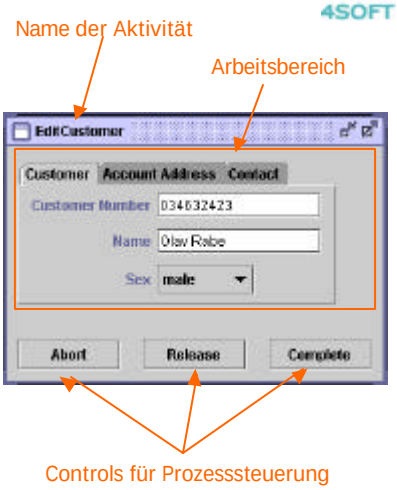


Implementierung

- Erweiterungen für Lücken im EJB-1.1-Standard
  - Automatische Erzeugung eindeutiger Primary Keys
  - Persistente Beziehungen zwischen Entity Beans
  - EJB-Komponentenvererbung
  - Erweiterte Laufzeitinformationen für Beziehungen
- Anpassung der Workflow Engine Verve
  - Zugriff auf prozessrelevante Geschäftsobjekte
  - Aufruf von Session Beans für autonome Aktivitäten
  - Einheitlicher Zugriff auf JNDI und ORB
- Java-Client
  - URL Class Loader für **Worklets**
  - Generische GUI-Komponenten zum Zugriff auf beliebige Entity EJBs

Worklets

- Wiederverwendbare, einbettbare EJB-GUI-Komponenten
  - Entsprechen den modellierten Aktivitäten
  - Ablaufumgebung wird vom GUI-Client gestellt
  - Selbst wiederum aus wiederverwendbaren Komponenten aufgebaut
- Server-seitige Verwaltung
  - Speicherung im Dateisystem
  - Laden über HTTP-Zugriff analog zu Applets



Generische GUI-Komponenten

- Zugriff auf beliebige EJB-Komponenten
  - Greifen zur Darstellung der GUI-Elemente auf Java Runtime Type Information sowie erweiterte Typ-Information für Beziehungen zu
  - Weitgehende Konfigurierbarkeit später vorgesehen
- Beispiel: Entity Editor / Viewer
  - Editiere / zeige Features und Attribute eines Geschäftsobjekts internationalisiert an
  - navigiere zu assoziierten Geschäftsobjekten

Beschreibungstechnik

- Modellierung der Software-Architektur und der Mechanismen mit UML
  - Klassendiagramme für Schnittstellen und Klassen
  - Sequenzdiagramme für Beispielabläufe und Mechanismen
- Defizite von UML als Architekturbeschreibungssprache
  - Ungenügende Beschreibungstechniken für Grobarchitektur
  - Keine Abstraktions- und Verfeinerungskonzepte – die Zusammenhänge zwischen einem abstrakten und einem konkreten Modell können nicht sinnvoll gezeigt werden
  - Component Diagrams und Deployment Diagrams können informelle „Architektur-Blockdiagramme“ nicht ersetzen

Methodik und Vorgehen

- Schritte beim Entwurf der Software-Architektur
  1. Evaluierung von Techniken und Komponenten
     - Recherche im Internet
     - Experimentieren mit Inselprototypen
  2. Erarbeitung der Grobarchitektur
     - Erstellung Architekturdokument
     - Erarbeitung eines Durchstichs zum Beweis der Machbarkeit
  3. Ausarbeitung von Mechanismen
     - Verfeinerung Architekturdokument
     - Erarbeitung eines evolutionären Prototyps
     - Schrittweise Erweiterung des Prototyps
- Einbettung in restlichen Entwicklungsprozess
  - Anschließend an Anforderungsanalyse
  - Paralleler Entwurf von Fachlichkeit und Technik
  - Fließender Übergang in Implementierung

Wiederverwendung

- Blueprints und Standardarchitekturen
  - Keine starren Vorgaben für Projekte
  - CORE: **Co**mmon **R**oots of **E**ngineering als Baukasten von Techniken, Prozessen und Komponenten, die wiederverwendbar sind
- Wiederverwendung von Architekturen
  - Auf Code-Ebene
    - Wiederverwendung von Komponenten (Beispiele: generische GUIs, Logging)
  - Auf Design-Ebene
    - Dokumentation und Verwendung bewährter Architekturmuster (Beispiel: Konfiguration von Komponenten)
    - Verwendung existierender Standards und Komponenten
  - Auf Spezifikations-Ebene
    - Nur Wiederverwendung von Textbausteinen

Qualitätsmerkmale / Prinzipien

- Qualitätsmerkmale
  - Flexibilität, Anpassbarkeit und Erweiterbarkeit
  - Einfache Schnittstellen für Anwendungsprogrammierer
  - Verwendung existierender Komponenten
- Entwurfsprinzipien
  - Trennung von Fachlichkeit und Technik
  - Modulare Komponenten-Architektur
  - Lose Kopplung durch Verwendung von Adaptern (im Projekt für Datenbank und WFE)
  - Modellierung und Generierung statt Codierung

FE Architecture

1 Big Picture
2 Components
3 Persistency
4 Presentation
5 Middleware

THINKING AHEAD FOR YOUR EFFICIENCY.

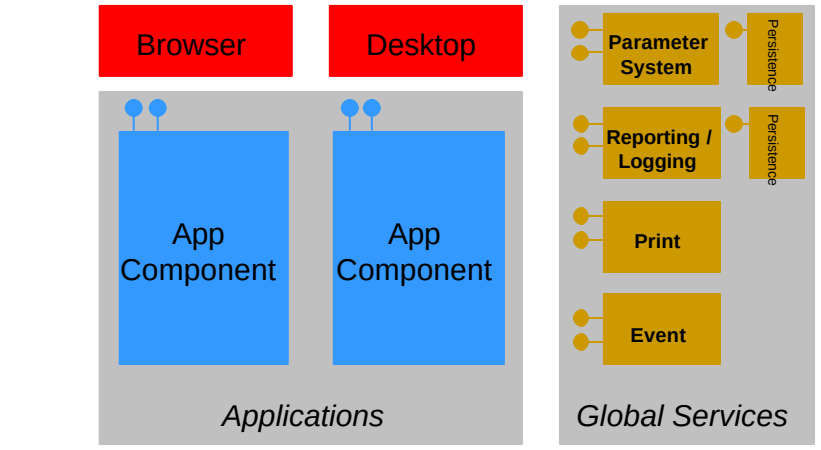
FE Architecture: Big Picture

Scope

- Support for multiple sales channels
- Scalability
- Reduce software deployment effort
- Open system standards
- Multiple language and currency support
- Support for multiple databases, middleware and operating systems

THINKING AHEAD FOR YOUR EFFICIENCY.

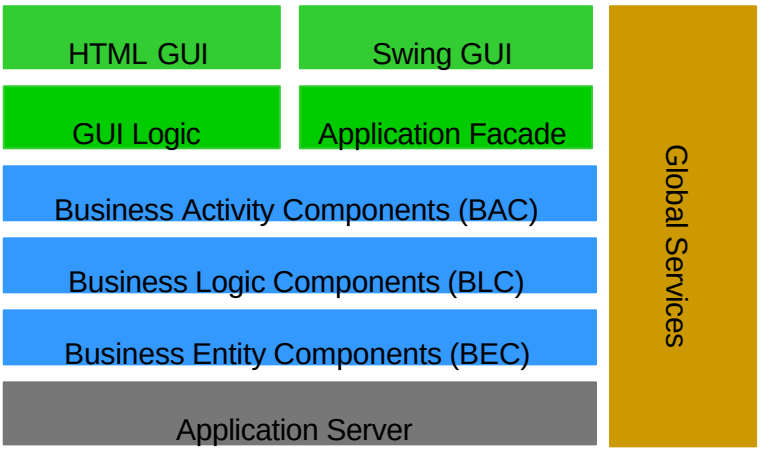
FE Architecture: Big Picture



THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Big Picture

Layers



THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Big Picture

BAC (Business Application Component)

- The BAC components provide the interfaces for the front-end development
- All BAC object methods need a transaction context
- Almost all BAC objects are stateless
- Any context kept across technical transactions is very small
- Instead of object hierarchies, the context only contains object IDs and pure data

FE Architecture: Big Picture

BLC (Business Logic Component)

- A BLC component contains persistent business objects
- These objects implement the business logic based on the persistent objects
- Typically, these objects do not maintain object hierarchies
- The state of these objects is only intended for short term usage

FE Architecture: Big Picture

BEC (Business Entity Component)

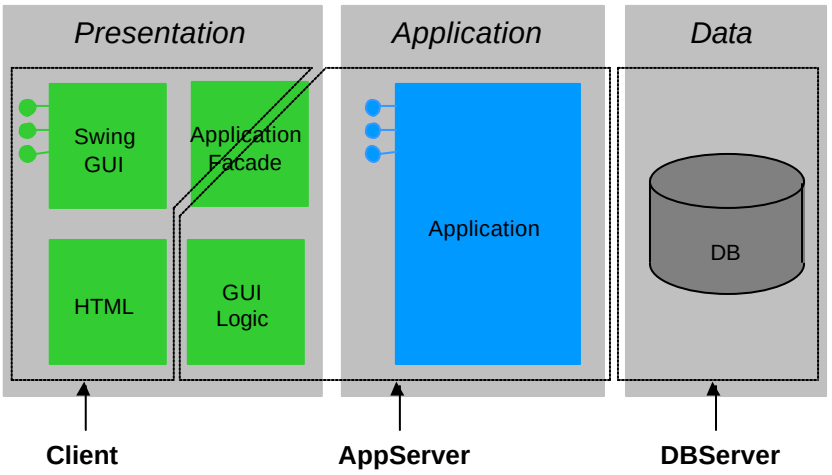
- Usually, there is one BEC component for each BLC component
- The BEC component contains all persistent objects needed for the BLC
- Persistent objects do not contain any business logic
- A BEC provides interfaces to create, update, remove, and query persistent objects

FE Architecture

1 Big Picture
2 Components
3 Persistency
4 Presentation
5 Middleware

FE Architecture: Components

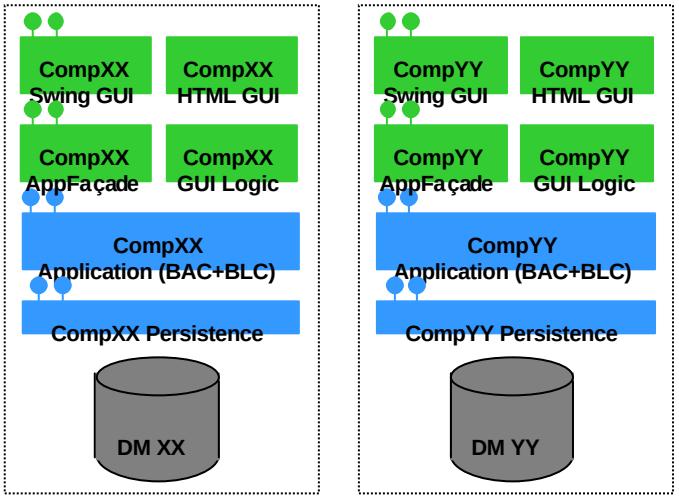
3 Tier Architecture (Logical and Physical Structure)



THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Components

Complete Component



THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture

1 Big Picture
2 Components
3 Persistency
4 Presentation
5 Middleware

THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Persistence

Persistence Features

- High data integrity (using DB constraints)
- Support for major DB Systems (MS SQL, Oracle, UDB)
- Code generation for entity objects and query functions
- Simple transaction handling

THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Persistence

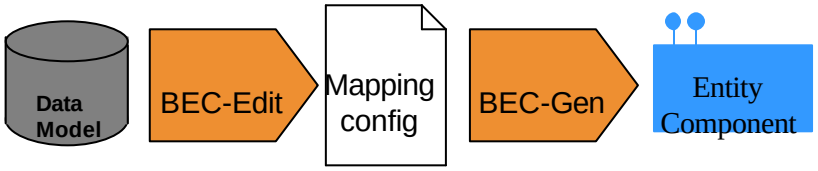
Persistence Strategy

- NSE O2R Tool (better R2O Tool)
- Entities are no side effects of an object model
- Bottom-up: On the base of a well designed data model, we create entity objects
- The data model is optimized for performace and queries
- Framework supports serialization of DB operations (allows the use of DB constraints)

THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Persistence

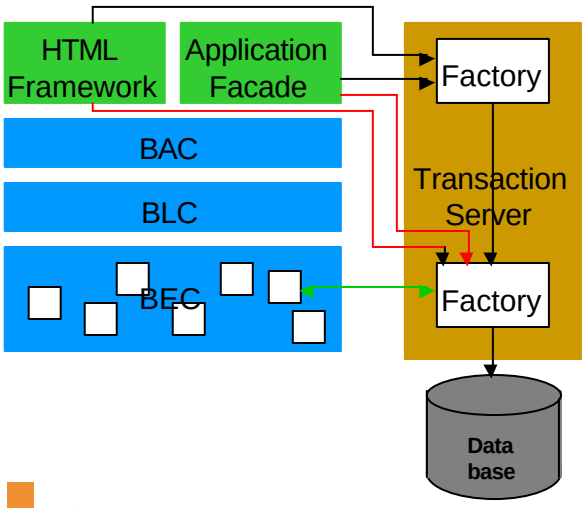
Persistence Development Process



THINKING AHEAD FOR YOUR EFFICIENCY.

NSE FINANCIAL SOLUTIONS

Transaction Handling



- Central transaction service
- Transactional behaviou managed by context ar BEC
- Developer uses contex control transaction
- Concurrency solved by optimistic locking

THINKING AHEAD FOR YOUR EFFICIENCY.

NSE FINANCIAL SOLUTIONS

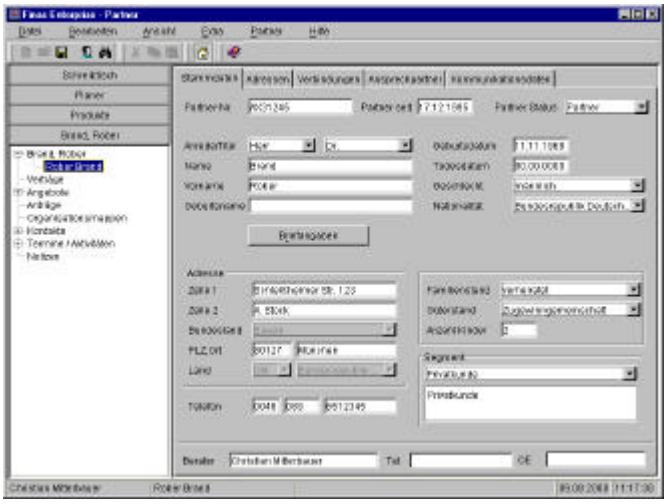
FE Architecture

1 Big Picture
2 Components
3 Persistency
4 Presentation
5 Middleware

THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Presentation

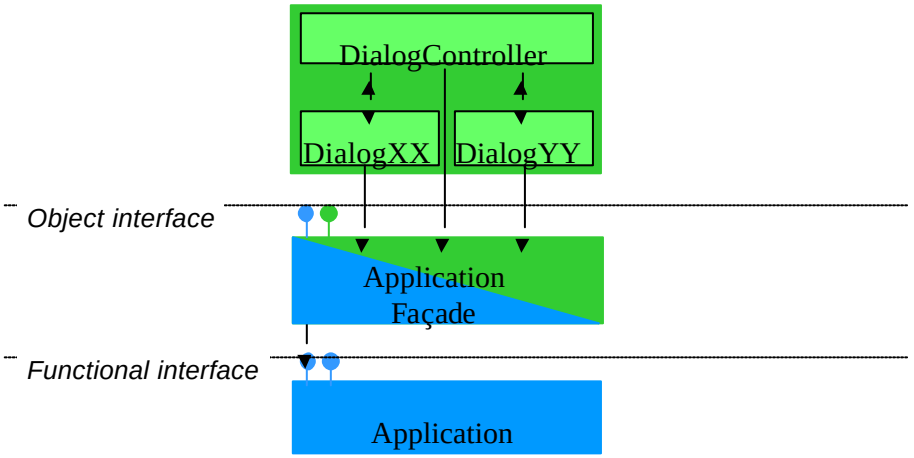
Sample Desktop



THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Presentation

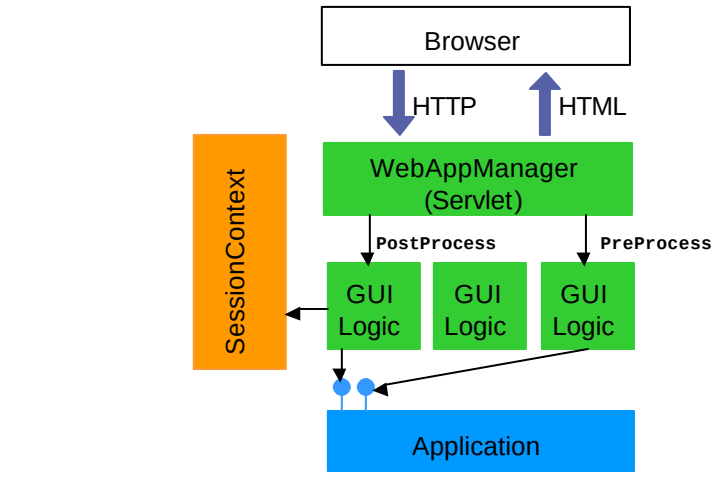
Swing GUI: Application Façade



THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Presentation

HTML GUI: Web Application Framework



THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture

1 Big Picture
2 Components
3 Persistency
4 Presentation
5 Middleware

THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Middleware

Remote Object Interface (ROI)

- The remote object interface is an abstraction layer for different middleware systems
- The ROI interface definition language (IDL) is a subset of the CORBA IDL
- This IDL reduces complexity and allows the support of different middleware systems
- The NSE implementation offers a complete lightweight broker

THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Middleware

ROI Features

- Language mapping for C++ and Java
- Supports Windows NT and various UNIX systems
- HTTP Tunneling for internet use
- Lightweight communication stubs (client base < 20K)
- Different process / thread models
- Load balancing
- Small footprint for standalone use

THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Middleware

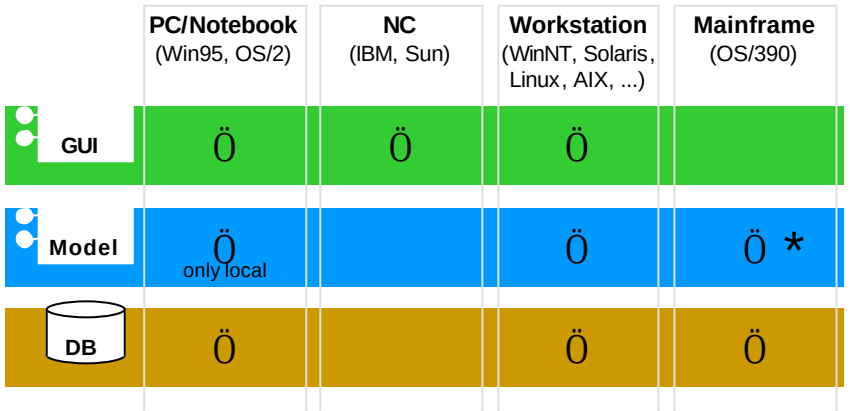
ROI Migration Scenarios

DCOM	migration for all C++ services (not of interest any more)
CORBA	migration for all Java and C++ implementations (important deployment scenario)
RMI	migration for all Java and C++ implementations possible (not implemented)
EJB	More restrictions on the IDL (important deployment scenario)

THINKING AHEAD FOR YOUR EFFICIENCY.

FE Architecture: Development

Supported Platforms



* if appropriate application server is available

THINKING AHEAD FOR YOUR EFFICIENCY.

Reflektion: Beschreibungstechnik

- Systemarchitektur als Grundlage projektspezifischer Architekturen
- Beschreibung umfaßt PPTs, Prosa, HTML-Dokumentation des Frameworks sowie Beispiele
- UML tritt wo geeignet in der HTML-Dokumentation auf
- Klassen- und Sequenzdiagramme
- Defizit erkennbar bei dem Versuch die vorangegangenen Folien in UML abzubilden

Reflektion: Methodik und Vorgehen

- Keine spezielle Methodik da einmaliger Vorgang
- Konsequente Überarbeitung aufgrund der Erfahrungen aus den Projekten
- Änderung aufgrund des sich ändernden technischen Umfelds
- Systemarchitektur dient als Grundlage der projektspezifischen Architekturen
- Unterstützung verschiedener Architekturpatterns durch das Framework

Reflektion: Wiederverwendung

- Systemarchitektur als Basis sollte immer wiederverwendet werden
- Auch bei Nicht-Standard-Projekten dient die Systemarchitektur als Anhaltspunkt um evtl. Teile des Frameworks nutzen zu können

Reflektion: Qualitätsmerkmale und Entwurfsprinzipien

- Priorisierung liegt auf längerfristig tragender Basis
- Auch bei Nicht-Standard-Projekten dient die Systemarchitektur als Anhaltspunkt um evtl. Teile des Frameworks nutzen zu können

Entwicklung der Systemarchitekturen in der Bayerischen Landesbank

Architektur Workshop München 2000

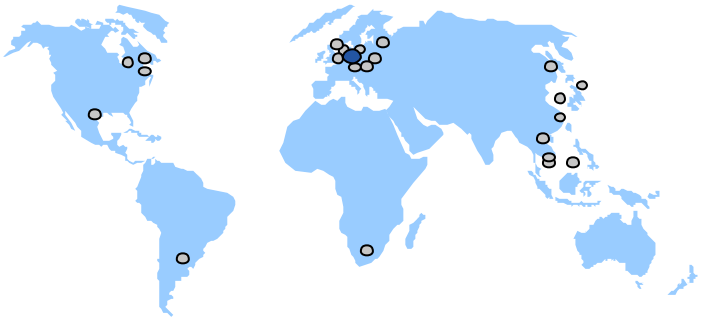
5.12.2000

Gerhard Klausen
Stefan Finkenzeller
UB1500 Informatik und Organisation

Bayerische Landesbank

Vorstellung

„Die Bayerische Landesbank in München ist eine der größten Universalbanken Deutschlands und als eine der wenigen Banken mit Triple-A-Rating an allen bedeutenden Finanzplätzen der Welt vertreten. Als Geschäftsbank bieten wir unseren Kunden das ganze Spektrum moderner Finanzdienstleistungen. Daneben ist die Bayerische Landesbank die Hausbank des Freistaat Bayern und die Zentralbank der Bayerischen Sparkassen."



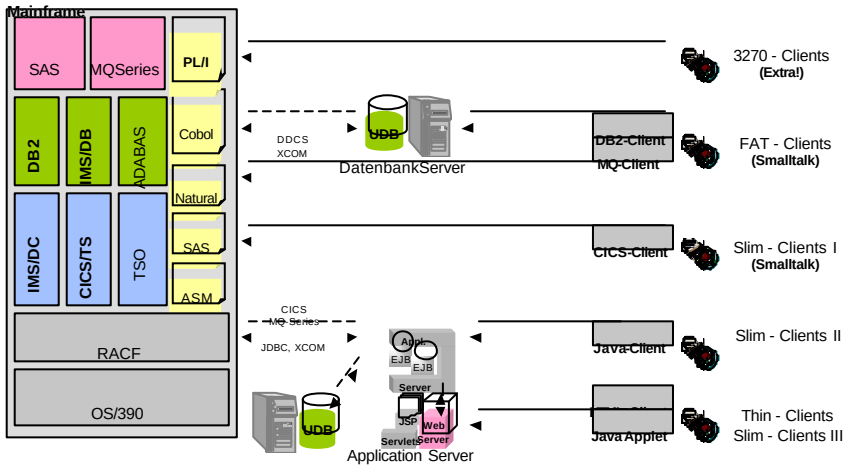
Entwicklung der System-architekturen in der Bayerischen Landesbank

Architekturgrundsätze

- So zentral wie möglich, so dezentral wie nötig
- Sicherheit und Nachvollziehbarkeit sind zentrale Bestandteile
- Thin Client Architektur, geeignet für Intranet, Internet, Extranet
- Datenverarbeitung dort, wo die Daten liegen
- Unnötiges Datenkopieren über Architekturgrenzen vermeiden
- Integration bzw. Anbindung der Backends (zentral/dezentral)
- Berücksichtigung von Mission Critical Applicationen bzgl.
  - ✓ Transaktionssicherheit
  - ✓ Verfügbarkeit
  - ✓ Skalierbarkeit
  - ✓ Managebarkeit

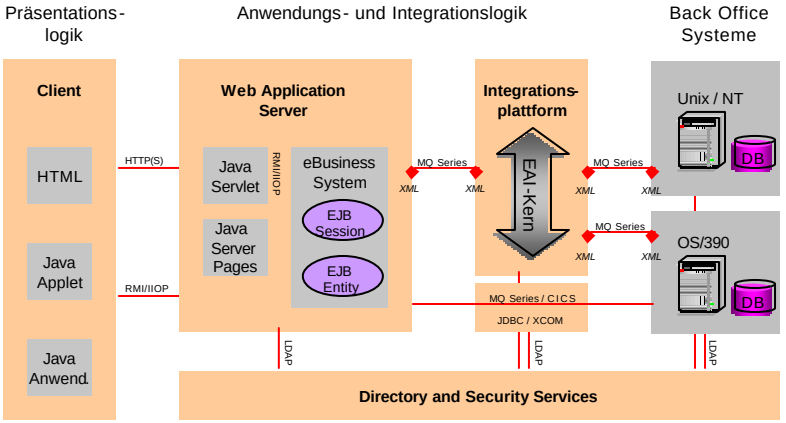
Entwicklung der System-architekturen in der Bayerischen Landesbank

Entwicklung der Systemarchitekturen



Entwicklung der System-architekturen in der Bayerischen Landesbank

Strategie Systemarchitektur



Entwicklung der System-architekturen in der Bayerischen Landesbank

Benefit

- Architektur für Mission Critical Anwendungen
- Saubere Trennung der Darstellung, Business Logik und Datenhaltung
- Transparente Anbindung über Standardschnittstellen
- Wiederverwendung auf Basis der technischen Komponenten
- Firewall-taugliche Kommunikation mit Verschlüsselungsmöglichkeit (HTTPS)
- Multikanalfähig (HTML, Java-Anwendung, WAP, ...)
- Steigerung der Produktivität durch den Einsatz von Frameworks und Wizards
- Integrierte SEU mit Debugging und Testclients (VAJava und Websphere)
- Unterstützung von multiplen Plattformen (NT, Unix, OS/390)

Fragen ?

Beschreibungstechnik für Systemarchitekturen

- **„Grobe“ Systemarchitektur**

  Erster Architekturentwurf, der die beteiligten Systeme, Komponenten, Programmier-sprachen, Middleware, Kommunikationsverbindungen und externe Anbindungen grob beschreibt.

  - PowerPoint mit eigenen Symbolen

- **Anwendungsarchitektur**

  Beschreibt die logischen Bestandteile der Anwendung, deren Beziehungsstruktur sowie die fachlich verarbeiteten Daten.

  - Rational Rose mit UML-Erweiterung ARCUS

- **„Feine“ Systemarchitektur**

  Technische Realisierung der Anwendungslandschaft. Sie besteht aus einer Hardware- und Softwaresicht.

  - Rational Rose mit UML-Erweiterung ARCUS

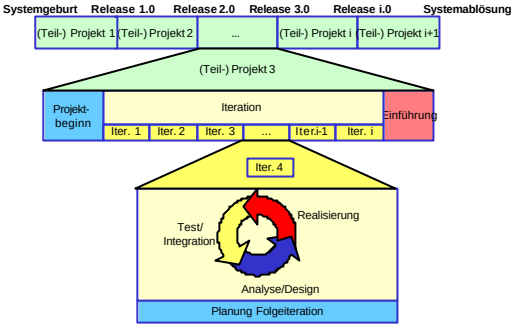
Methodik und Vorgehen

BLB Unified Process V2 (BUP)

Allgemeines Vorgehensmodell für die Anwendungsentwicklung mit Ergebnismodell und integrierten Qualitätsmanagement.

Durch den gesamten Entwicklungszyklus ziehen sich Aktivitäten zur laufenden Aktualisierung und Abstimmung der Systemarchitektur.

Es erfolgt eine enge Zusammenarbeit zwischen Projekt und den Systemarchitekten. Dadurch ist eine praxisbezogene Weiterentwicklung der Systemarchitektur gewährleistet.



Der Mythos „Wiederverwendung“

Seit jeher werden in der EDV Mittel und Wege gesucht Arbeitsergebnisse wiederzuverwenden. Hierbei haben sich zwei Hauptprobleme herauskristallisiert:

- ☞ Höherer Aufwand bei der Entwicklung
- ☞ Problem der Wiederfindung

Architekturen

- × Standardarchitekturen auf Basis des Technologieplans

Technische Wiederverwendung

- × Funktioniert in Teilen (z.B. bei zentralen Modulen)
- × Problematisch ist aber die Wiederverwendung von Anwendungsteilen, da hoher Abstimmungsaufwand und meist nicht allgemeingültig programmiert
- × Die Hoffnung ruht auf EJBs!

Fachliche Wiederverwendung

- × Fachliche Wiederverwendung ist ein Indiz, dass Anwendungen doppelt gemacht werden
- × Uneinheitliche Definitionen von Sachverhalten in den Fachbereichen

Entwicklung der System-architekturen in der Bayerischen Landesbank

Qualitätsmerkmale und Entwurfsprinzipien

Qualitätsmerkmale und -aktivitäten sind im Vorgenesmodell (BUP) fest verankert. Die Priorisierung erfolgt projektspezifisch nach Anforderung.

Unsere QS-Merkmale

- Wartbarkeit
- Benutzerfreundlichkeit
- Korrektheit
- Zuverlässigkeit
- Effizienz
- Sicherheit
- Wiederverwendbarkeit
- Performance
- Produktivität

Die grundlegenden Entwurfsprinzipien, *Schichtenmodell und Kapselung*, bieten eine hochwertige Basis um diese Qualitätsmerkmale zu erreichen.

SIEMENS

Software-Architektur für Vermittlungsrechner in Telefonnetzen mit Internet-Konvergenz

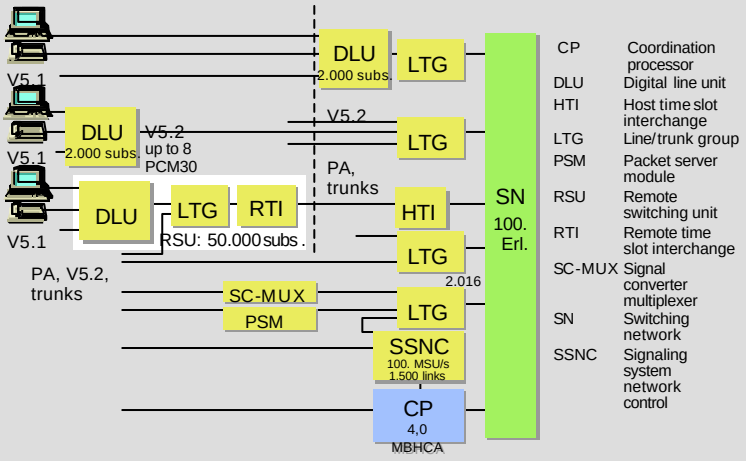
SIEMENS

Inhalt

- Software-Architektur für Telefonnetze
- Produktionsaspekte
- Software-Architektur für Server zur Internet-Konvergenz
- Resümee

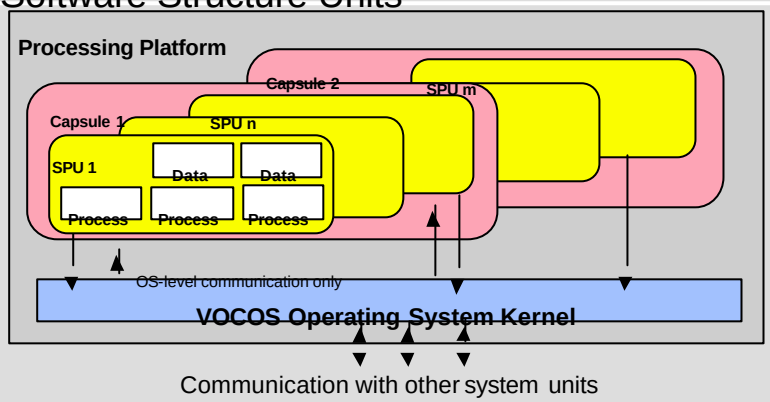
SIEMENS

EWSD Overview



SIEMENS

Software Structure Units

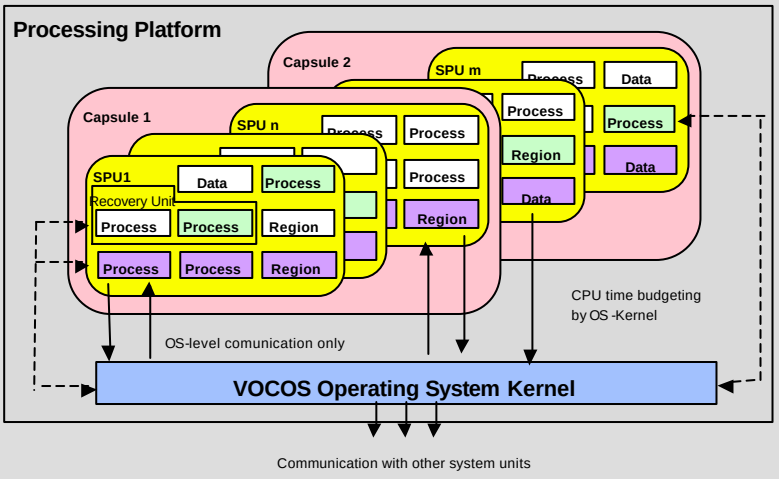


Communication with other system units

- Capsule: virtual address space, heap budget
- SPU (Service Provision Unit): location transparent design unit
- Message based communication

SIEMENS

Software Structure Units



SIEMENS

Service Provision Unit

- **Fundamental design unit, consists of modules, processes, and procedures**
- **Provides one or more functionally related services**
- **Shared memory interfaces possible within SPUs (same address space)**
- **External interfaces: only message-based communication (Inter Process Communication, RPC, Service Addressing)**
- **Relocatable unit i.e. can be moved from one HW platform or capsule to another at build time**
- **Can be replicated at build time on different HW platforms or on the same HW platform (e.g. with respect to given data partitioning)**
- **Unit of testing (SPU-testbed)**

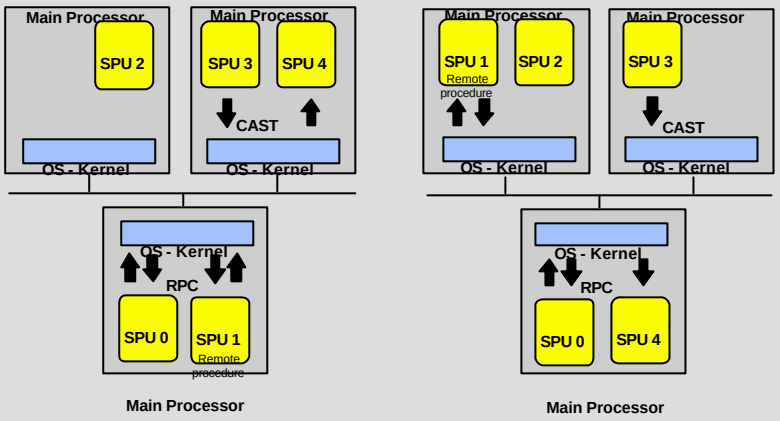
SIEMENS

Capsule

- Unit of resource allocation:
  - own protected virtual address space
  - own guaranteed heap budget
  - own patch area
- Unit of optimization: communication is optimized if it stays completely within the same capsule
- Capsule is a build time unit: at build time
  - SPUs are grouped together to form capsules
  - capsules are allocated to loadtypes
  - loadtypes are associated to platforms

SIEMENS

Location Transparency



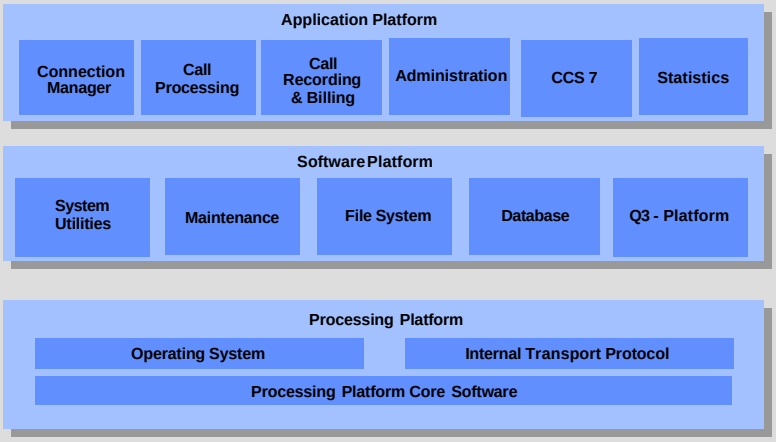
SIEMENS

Objectives for Software Structure

- Support Distributed System Architecture with High Complexity
- Scalable Performance
- High Reliability
- Realtime System with High Load
- Decouple hardware evolution from application software
- Flexible addition of new features and new services

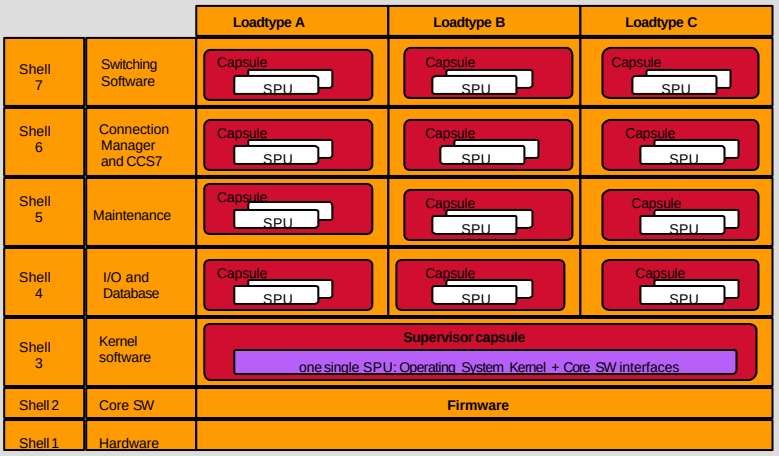
SIEMENS

Software Layers



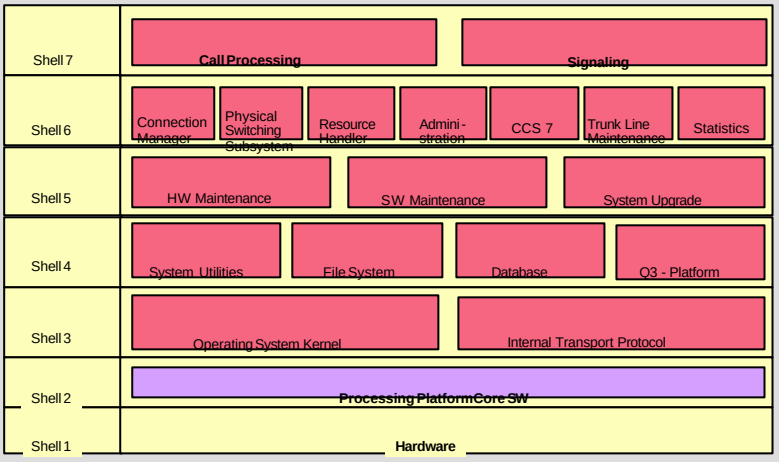
SIEMENS

Shell Model



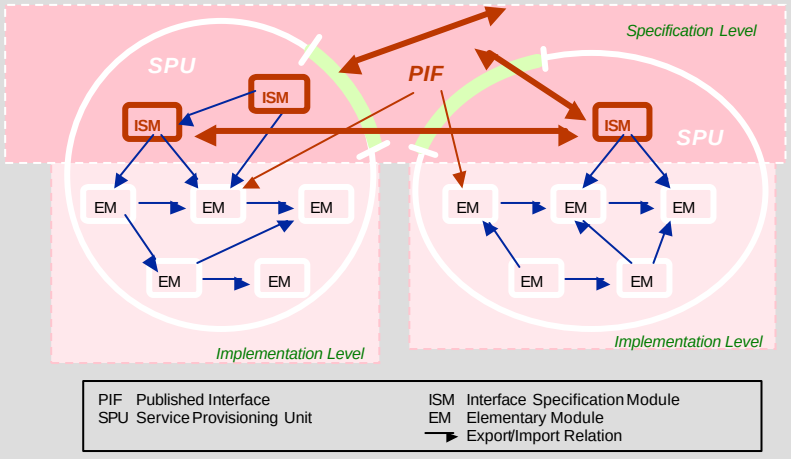
SIEMENS

Shell Contents



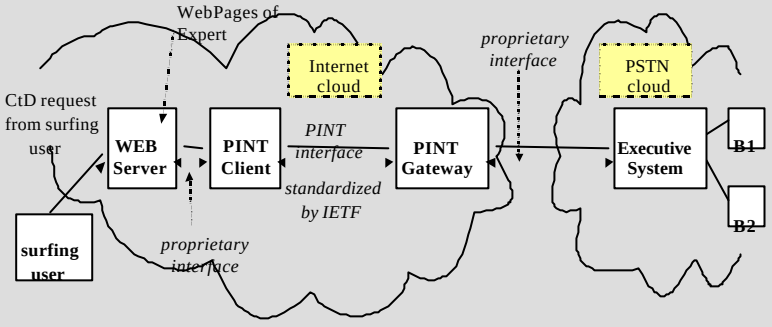
SIEMENS

Software Production and Global Interfaces



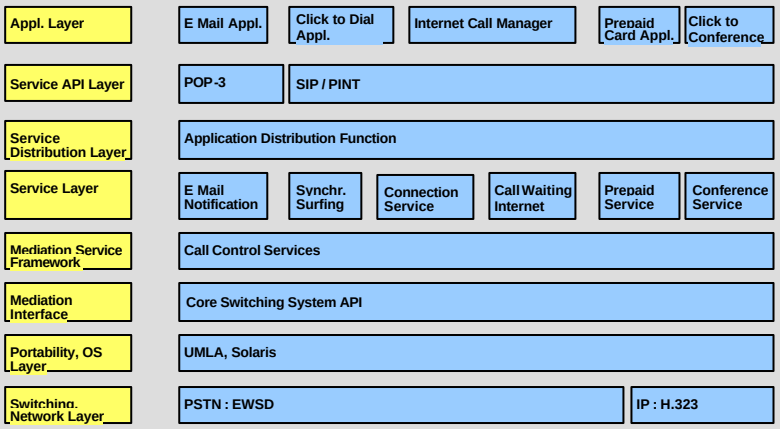
SIEMENS

Internet Supplementary Services Network



SIEMENS

Layer Architecture for Open Service Unit



SIEMENS

Application & Component Description

Application Description:

General Description
Use Cases / Main Tasks
Component Collaboration
Load Model, Memory/Processing Budget

Component Description:

General Description
Use Cases / Main Tasks
Relations to other Components
Load Model, Memory/Processing Share

Specification of Interfaces:
Static Component Interface Diagram
Main Execution Paths
Description of Methods

SIEMENS

Thema I: Beschreibungstechnik

1. Eingesetzt werden SDL, MSC, CHILL (ITU) ; UML (OMG).

   Beschrieben werden damit die Systemstatik, die Schnittstellen und die wesentlichen kausalen Systemabläufe.
   Kriterien sind Zuverlässigkeit, Verständlichkeit, Erweiterbarkeit, Skalierbarkeit und die Integration von Standards.

2. Defizite von UML sind unzureichende Strukturierungsmöglichkeiten für komplexe Systeme.
   Es müssen Behelfe geschaffen werden, etwa Textfelder zur Kennzeichnung von Prozeßgrenzen.

SIEMENS

Thema II: Methodik und Vorgehen

1. Methodisches Vorgehen beim Entwurf:
   Schritt 1: Schichtenmodell mit Komponenten
   Schritt 2: Verfeinerung zu SDL-Diagrammen bzw. Objektmodell, jeweils mit MSCs
   Schritt 3: Lastmodell, Memory und Processing Budget

2. Einbettung in restlichen Entwicklungsprozeß:
   Entwurf ist Grundlage für Implementierung und für Systemtest, für Systemtest insb. die MSC-Abläufe.
   Generierung von Programmgerüsten und Testfällen.
   Formale SDL-Verifikation mit automatischem Tool.

SIEMENS

Thema III: Wiederverwendung

1. Standardarchitekturen:
   Grundlage ist das Schichtenmodell.
   Die Kommunikation zwischen Komponenten erfolgt über klar definierte Schnittstellen, die zum Teil standardisiert sind.
   Die Systeme werden über einen sehr langen Zeitraum weiterentwickelt, wobei entweder auf der vorhandenen Architektur aufgesetzt wird oder die vorhandene Architektur gekapselt und in eine neue Architektur integriert wird.

2. Wiederverwendet wird
   - die wesentliche Strukturierung des Schichtenmodells
   - Komponenten.

SIEMENS

Thema IV: Qualitätsmerkmale und Entwurfsprinzipien

1. Qualitätsmerkmale sind:
   Ausfallsicherheit (< 3 Min in einem Jahr)
   Wartbarkeit über lange Zeiträume

2. Entwurfsprinzipien sind:
   Lose Kopplung von SPUs und Komponenten
   Datenkapselung
   Modularität
   Defensive Programmierung
   Verifizierbarkeit