Togaf

GA: Organisation (2), Ziele (1), Prozesse (2) ISA: Geschäftsobjekte (3), Fachliche Dienste, Anwendungen (4) TA: Plattformdienste (5), Netzwerkdienste (5)

- Organisation und Prozesse hängen oft zusammen.
- Fachliche Dienste meist mit Prozessen identisch

Architektur: Gegenüberstellung Prozesse, Anwendungen, Geschäftsobjekte - Welche Anwendung in welchem Prozess? - Welche Prozesse brauchen welche Geschäftsobjekte? - Welche Anwendung macht wann / was / wie mit den Geschäftsobjekten?

Anforderungen: - L1 zwischen GA und ISA - L2 zwischen Geschäftsobjekte und Fachliche Dienste, Fachliche Dienste und Anwendungen - L3 TA, zwischen Plattformdiensten / Netzwerkdiensten

Systems Engineering

Bemerkung: Big Bigger Biggest (BBC Serie)

Einleitung

System: systema: zusammengesetztes, Komplex interagierender Elemente, drei Dimensionen: Identität (bleibt auch unter Änderungen stabil, Identität durch Symbole / Traditionen), Organisation (Struktur, Regeln, Gesetzmässigkeiten, def. Beziehungen), Zielgerichtetheit

Nach INCOSE: System Kombination von interagierenden Elementen, welches einem def. Zweck zu erfüllen hat. HW, SW, Firmware, Menschen, Information, Techniken, Einrichtungen, Dienste und andere Hilfsmittel, künstliches / technisches System

Kybernetik: Allgemeine Theorie der Maschinen

Enterprise Architecture Management: TOGAF (nur subset)

Aufgaben Architekt: Ordnung, Positionierung im richtigen Layer (Welche Logik wohin)

Hierarchie eines Systeme

Systemhierarchie, Architektur: Neue Anforderunge: wohin gehört diese? Gliederung / Anordnung, Problem: immer mehrere Teilelemente betroffen,

Service-Baum im Configuration Management (kennt Mehrfachverknüpfungen, entspricht nicht der Systemhierarchie)

Beispiel: Nicht relevant

Systemmodel als Basis

Architektur betrifft das ganze, keine Trennung SW / HW

Flash-Back

- System-Begriff: Bertallan... / INCOSE
- Artikel lesen:

1. Abstract lesen
2. Zusammenfassung lesen
3. Titel lesen, interessante markieren
4. Zeichnung anschauen
5. Entscheid: Weiterlesen Ja / Jein / Nein

- 2 wesentliche Zeitschriften (1 Jahr Vorsprung)
- Communications of the ACM (www.acm.org)
- IEEE Computer

###Systemdenken - Systemtypen - Systems Engineering Problem in Teilprobleme aufsplitten bis Teilproblem lösbar

1 Innerer, 3 Äussere Zyklen

...

####Transparent IT: Wenn nicht durchsehbar, für andere: wenn durchsehbar

####Komplexität Komplexität != kompliziert, kompliziertes Problem: Zerlegung in Teilprobleme, Komplexität: u.U. nicht lösbar, Schranken / Eckpfeiler: System wird an diesen Punkten instabil, Lösung: im stabilen Bereich

Def. von Webster: Aus Sicht Engineering falsch, damit können wir umgehen

###Flash-Back - Was ist ein System? (Big Bigger Biggest) - Tatsachen rund um Systeme (Ursprünglich für Bau komplexer technischer Systeme, Verschmelzung mit SWE, da System of Systems (Alles vernetzen)) - Systemdenken (Zerlegung, IT kann alle Probleme lösen, gegeben durch Denke, Unverständnis bei Kunden) - Systemtypen - Komplexität (vs kompliziert, Reduktion möglich oder nicht?, Komplexität verstecken, Transparenz)

System der Systeme (System of Systems, SoS)

Heute: Leute mit Req., 1 Technologie, 1 System, 1-n Rechner Zukunft: Leute mit Req, Viele Systeme, die bereits verbunden sind, Problemstellung: Verknüpfung mit der Systeme mit einem neuen System, N-Technologien, benötigt: ganzheitliches Engineering, Verständnis der Interaktion zwischen den Systemen

SoS Prinzipien / Eigenschaften 1. Unabhängigkeit (Dezentrale Verwaltung) 2. Evolution (System verändert sich mit Daten, die es verarbeitet) 3. Eigendynamik (Systeme: Wasserstände messen, Wetter messen - Hochwasser + Schlechtwetter vorhersage, nur in Kombination sinnvoll - Überwachungsdrohne für Überprüfung, SoS für Koordination / Monitoring) 4. Geografische Verteilung

Systems Engineering Prozesse

Systems Engineering Vorteile gegenüber SWE: IEEE: jedes System hat einen Lifecycle (Lifecycle und Produkt gehören zusammen), Mensch ist Teil des Umsystems | INCOSE: 5 Prozesse: Technical Management, Acquisition & Supply, System Design, Product Realization, Technical Evaluation - Einige Teilprozesse noch nicht etabliert

SEP: Naturgemäss umfangreicher als beim SWE, Standards auch umfangreicher, wesentlicher Standards: IEEE, INCOSE, DAU gibt es nicht (da Mensch teil von System, nicht für System zugelassen)

Adaptive Systeme

Konzeptionelle Grundidee aus 90er Jahren, Engineering Ansatz

Konzepte: - Ubiquitous Computing (Allgegenwärtig) - Autonomous Computing - Semantic Web

Ubiquitous Computing

Computing erst nützlich, wenn wir es nicht mehr sehen

Kontext: Cyber-Physical-Systems (Smart Devices mit Sensor oder Aktor, Netzwerk), Lokationsmodelle (Location Based, Object Identification, Situation Based)

Autonomous Computing

Grundlage: Agent, dynamisches System, können kommunizieren

Semantic Web

Inhalt verstehen, Strukturierung Inhalt via Onthologien (Wissensbäume), Reaktion auf Wissensbäume, hat sich nicht durchgesetzt

Computational Reflection

Introspection: Selbstbeobachtung, Intercession: Aktionen auf bestimmte Beobachtungen - via Metaobject Protocols

Architecture Basics

Einleitung

Enterpriseebene: Prozesse, Organisation **Architektur:** Enterprise -> Anforderungen an Anwendungen, Abbildung Enterpriseebene auf Anwendungsebene, von aussen schauen für Architekturbild, z.B. Layering
Anwendungsebene: Wie aufbauen? Wo kommt was hin?

Architektur Architektur beeinflusst wie sich System und Systemteile verhalten

Werkzeugkasten: - Prinzipien - Basics - Standards - Theorie - Industrie - Know-How - Realität - #User - #Transaktionen - #Branche (Line of Business) - #Grösse der Firma

Cross-Cutting-Concerns: Anforderungen an allgemeines Verhalten

Triage der Anforderungen (Relevant für Architektur oder nicht), Mengengerüst wichtig

2.2 Architektur und Systemeigenschaften

Messbare Eigenschaften

Zur Laufzeit messbare Eigenschaften: - **Performance (Garantierte Antwortzeiten):** ~0.7s (von User tollerabel), Vermeidung garantien Antwortzeiten zu Beginn (schwierig zu messen, nicht beeinflussbare Faktoren), TCP / IP: Best Effort, zentrale Systeme: einfachere Verbesserung Performance, Verteilte Systeme: Schwierig - **Security (unautorisierter Zugriff, mutwilige Zerstörung):** User Management (keine lokale User-Verwaltungen!), Rollenbasierte Sicherheitskonzepte, Lokalitätsabhängig, Regulatorische Anforderungen hinsichtlich Daten - **Availability (Verfügbarkeit):** 3 Parameter: Uptime (% der Gesamtzeit oder Arbeitszeit), nicht redundante Systeme: ~96%, > 96%:

Redundante Systeme, Time-to-Recovery-Objective, Recovery-Point-Objective (Wie viele Daten dürfen max. verloren gehen) - **Usability (Verwendbarkeit):** Zweideutig: Ergonomie oder Verwendung für vorgesehenen Zweck - **Robustness (Stabilität):** Schwierig, mehrere Umgebungen, bei Einführung: Probleme wegen anderen Datenmengen, Betrieb: Schwierig zu managende Fehler, keine Reproduzierbarkeit auf Testumgebung, Cloud: vermehrte Auslagerung, Self-Recovery

Zur Laufzeit nicht messbare Eigenschaften: - **Scalability:** 90% der Fälle: Irrelevant, da für gewisse Anzahl User / Transaktionen konzipiert, z.T. sehr teuer in der Umsetzung (Lizenzen, Softwareentwicklung, etc.), Verlagerung in die Infrastruktur - **Integrability:** Was passt zusammen? Technologie- / Plattformentscheide, Integrationsschicht: Austausch von Daten zwischen Anwendungen - **Portability:** Verliert an Relevanz, BU-Kritisch (Produktentwicklung / High-End-Bereich), o.ä. von Entwicklerteam Portabilität verlangen -> bessere Qualität - **Maintainability:** DevOps, keine Dinge einbauen, die es nicht braucht / nicht zum kern gehören - **Testability:** Sinnvolle Logs und Traces einbauen, Änderung Modus ohne Neustart, Defensive Programming (Parameter-Range-Prüfung, One-Exit-Point) - **Reusability:** analog Portability, jede Komponente wird im Scnitt 1.3 x verwendet, nützlich für Qualität

Jede Architekturentscheidung hat Einfluss auf eine der Eigenschaften

Definition

1 Zeichnung reicht nicht für alles (bei Gebäuden funktioniert das), mindestens 3 Darstellungen: Komponenten, Datenflüsse, Prozesse, , Hierarchisch: Unterstrukturen –> Konstruktionselemente

Formale Definition

- Architecture
  - Elements
  - Processing Elements (Führend Transformationen auf Data Elements aus)
  - Data Elements (Daten)
  - Connecting Elements (Entweder P.E. oder D. E., z.B. Procedure Calles, Shared Data, Messages)
  - Formale
  - Weighted Properties (Gewichtung Eigenschaften Element, Unterscheidung zentral / dekorativ, Mittel zur Definition Rahmenbedingungen, minimale Anforderungen)
  - Relationships (Definition Platzierung Element in best. Kontext von Elementen)

  - Rationale (Motivation zur Auswahl best. Architektur-Elemente, Ausdruck Abbildung Systemanforderungen, funktional nach allgemeine Systemanforderungen.) ###Zusammenfassung
- Werkzeugkasten der Architektur
- Architektur hat Ordnungsfunktion, beeinflusst Art und Weise wie eine Anwendung oder Systemlandschaft geordnet wird (mit Werkzeugkasten)
- Architektur hat direkten Einfluss auf allgemeine Systemeigenschaften
- Messbare / Nicht-Messbare Eigenschaften

2.4 Architektur und Moduleigenschaften

Gute Architektur in konkretem SW-Design sichtbar, baasiert auf Reihe klar definierter Prinzipien. Prinzipien beziehen sich auf einzelne Module oder Verhältnis der Module zueinander

Architekturbild

Webseite: Logik: z.B. nach Bereichen, oder nach Use Cases, Cross-Cutting-Concerns auf Schichten aufteilen (1 Schicht -> 1 CCC)

1. In Schichten einteilen
2. Ordnungskriterium auswählen (Use Cases, etc.)

Moduleigenschaften

- **Modularity:** Eigenständige, in sich geschlossene Komponente, Arbeitsteilung, je grösser System, desto wichtiger, Typisch: User-Mgmt, Rule-Engine, SW ist Modular: wenn vernünftig sortierbar
- **Portability:** Auch in anderen Umgebungen lauffähig, Organisation Gesamtarchitektur
- **Changeability:** System verändert sich während Lebenszeit (Umfeld, UI, ...),Wie weit kann System verändert werden?
- **Conceptual Integrity:** Was zusammen gehört, muss zusammen wachsen, ähnliche Gestaltung ähnlicher Funktionalitäten, Referenzimplementationen, zentral
- **Intellectual Control:** Verständnis / Unterstützung durch Beteiligte, zentral
- **Buildability:** geht aus Conceptual Integrity und Intellectual Control hervor, System muss so spezifiziert werden, dass es von gegebenen Team in gegebener Zeit realisiert werden kann, zentral
- **Coupling and Cohesion:** Gemäss Programmierunterricht, Idealform: Lose Koppelung, je nach Situation: sinnvoll Koppelung aufheben (Performance)

  - Data Coupling
  - Stamp Coupling (Austausch Datenstrukturen)
  - Control Coupling (Austausch steuert Kontrollfluss)
  - Content Coupling (verändert Daten von anderem Modul)
  - Coincidental Cohesion (Gruppierung durch Zufall)
  - Logical Cohesion (Fkt. in Modul zusammengefasst, bezieht sich aber nich aufeinander)
  - Temporal Cohesion (Zeitpunkt Verwendung bestimmt Gruppierung)
  - Procedural Cohesion (Aufrufreihenfolge bestimmt Gruppierung)
  - Communications Cohesion (Gruppierung durch gemeinsamen I/O)
  - Sequential Cohesion (Abfolge Datenbearbeitung best. Gruppierung)
  - Functional Cohesion (Gruppierung hat Ziel dass Modul Logik und Daten lokal halten - Information Hiding)
  - Erreicht mit Independece of Design, Small Interfaces, Low Interface Traffic, Unity, Encapsulation

Übergreifende Themen

- **Design for Change:** Robust gegenüber Veränderungen, verschiedene Änderungsklassen:
  - Domain Specific Changes: Fachliche Änderungen, z.B. SAP
  - Analytical Changes: Nachbesserungen
  - Downsizing Changes: Reduktion gewisser Funktionalitäten, bei Agile: bereits eingebaut

2.5 Schwierigkeiten SW-Design:

```
- Complexity
- Conformity: Kein stabiler Untergrund
- Changeability: SW kann immer verändert werden
- Invisibility: keine visualisierung, keine geometrische Repräsentation
```

2.6 Vorteile / Ziele einer Architektur

- SWA Grundlage für Kommunikation: Verständnis, Detail-Fragen
- SWA treibende Kraft des System-Designs
- SW Rahmen für Wiederverwendung SW-Artefakte
- Träger von Qualitätscharakteristika (messbare allgemeine Systemeigenschaften) und nichtfunktionaler Eigenschaften
- Invariante über mehrere SW-P, Vereinfachung Entwicklungsprozess
- Analyse bestimmter Systemeigenschaften

Architektur Style

Einleitung

Allgemein nach Detail: - Enterprise - Application - Module

- EAM (Enterprise Architecture Management)
- Standards
- Style
- EA Pattern
- Pattern for SE (z.B. SOLID)

Architektur Stile

Fymilie von Software-Systemen, welche aufgrund Struktur und Semantik verwandt sind.

Definition Bestandteile: Vocabulary (Design-Elemente), Rules and Constraintes (Design-Regeln / -Einschränkungen), Semantics (Bedeutung Design-Elemente eindeutig), Analytics (Prüfung Vokabular, Regeln, Einschränkungen)

Architektonischer Stil weniger eingeschränkt und weniger vollständig als definierte Architektur

Typen: - Independent Components: Set unabhängiger (unabhängig lauffähiger) Komponenten, Kommunikation über Nachrichten - Communicating Processes (Parallelverarbeitung), Kommunikation (synchron / asynchron) via Kommunikationskanal zwischen Elementen - Event-Systems (GUI's, Real Time Systems), Prozesskommunikation via Event-Systems, 2 Dimensionen (Zeit, Regeln), Zeitnahe Verarbeitung, CEP: Complex Event Processing System (z.B. ESYER) = FAST-DATA, MVC als Event-System - Implicit Invocation: Linda (Urform implizites Eventsystem), Event-Space, Meldung Event, dann vergessen, einzelner Event spielt keine Rolle, eignet sich für viele Events - Explicit Invocation: MQ Series (bekanntestes System, garantiert Übermittlung), Event & Prozess: gemeinsamer Vertrag (z.B. Typ), jeder einzelne Event spielt eine Rolle, eignet sich für zählbare Events

- **Call-and-Return:**
  - Main Programm & Subroutine: Klassisches Programmierparadigma, eindeutiger Kontrollfluss, debugbar, Urform: Remote Procedure Call
  - Object Oriented: folgt Main Programm & Subroutine, Zugriffe auf Objekte via Schnittstellen, Fokus: Information Hiding, Kapselung Daten
  - Layered: Aufgaben logisch verteilen

  - Hierarchische Layers
  - Nicht-Hierarchische Layer, Zugriff via Interfaces
- **Virtual Machine:**
  - Interpreter: beschreibt abstrakte Maschine, definierte Elemente für inneren Aufbau, Java VM (simuliert Prozessor, Method Area, Heap, Java Stacks, PC Registers, Native Method Stacks),
  - Rule-Based System: Hat Wissensbasis (entweder fix oder trainiert), Generalisierung Interpreter, Trennung Maschine und Regeln zur Beschreibung Maschine, Backward- und Forward-Chaining
- **Data Flow:**
  - Batch Sequential: Transformation auf Daten von einander unabhängigen Elementen, Host- / Grossrechner-Technologie, wörtliche Umsetzung EVA
  - Pipes and Filters: z.B. Unix Pipes, flexible Verareitung Input, Streaming
- **Data Centered:**
  - Repository: DB
  - Blackboard: Künstliche Intelligenz, Datenbestand meldet sich bei Änderung bei Systemen, z.B. Kanban-Board

Hinweise: Innerhalb Stilfamilie mischen nicht zu empfehlen

Architektur Standards

Enterprise Architecture Management

Anwendungslandschaft zentraler Unternehmenswert, Ziel EAM: Sicherstellung der effizienten IT-Unterstützung, Umfeld: Geschäftsprozesse, Produktportfolios, Vertriebskanäle

Zusammenarbeit Unternehmens- / IT-Strategie

Architekturtypen Beispiele: - Business Architecture - Information Architecture - Technology Architecture - Solution Architecture

Methoden & Frameworks

- ISO
- Information System Architecture (ISA): 30 Modelle
- Open Distributed Processing (ODP): Verschiedene Betrachtungswinkel, Zusammenhänge

Architektur Stil	Anwendung	Vorteil	Nachteil
Independent Components			
Communicating Processes	Parallelverarbeitung	Einfache Modellierung, Skalierbarkeit	Komplexität der einzelnen Elemente
Event Systems	GUI's, Real Time Systems	Unabhängigkeit der Elemente, Änderungs-Freundlichkeit	Non-Deterministisches Verhalten der Elemente
Call-and-Return			
Main Program & Subroutine	Structured Programming, Client-Server (RPC)	Definierter Kontrollfluss	Skalierbarkeit, Erweiterbarkeit
Object Oriented	Allgemeines Design, Client-Server	Universell	Komplexität, Anwendungsfreiheit
Layered	SOA, Multi-Tier Architectures	Konzeptionelle Integrität, Lokalität der Änderungen	Performance, Komplexität
Virtual Machine			
Interpreter	Prozessor- und Betriebssystem-Simulation	Portabilität, Flexibilität	Performance
Rule-Based Systems	Expertensysteme	Flexibilität durch Regelwerk	Komplexität, Performance
Data Flow			
Batch Sequential	Host Systeme	Datensteuerung	Flexibilität, Interaktion
Pipes and Filters	Software Converter, Compiler	Flexibilität, Verteilung	Komplexität
Data Centered			
Repository	Stammdatenverwaltungen	Einfach	Single Point of Failure
Blackboard	Datengesteuerte Kontrollsysteme	Skalierbar	Anwendung eingeschränkt

Figure 1:

- TOGAF
- GERAM
- CIMO
- PERA
- ...

Business Architecture **Die Idee:** Formalisierung Geschäftstätigkeit, Problematik: Formalisierung IT sehr weit fortgeschritten, aber: geschäftliche Tätigkeit kennt Formalisierung nur im Bereich Business Process Engineering –> ein Stück fehlt

Die Elemente: Wertschöpfungsnetzwerk, Geschäftsmodell, Prozesslandkarte und Wertschöpfungsketten, funktionenmodell, Informations- und Datenmodell, Bebauungsplan zur Strategieumsetzung

- Wertschöpfungsnetzwerk: Was macht wer in der Wertschöpfung? Visualisierung im Markt mit Position und Umfeld, Darstellung Marktrollen
- Geschäftsmodell: Visualisierung, Beschreibung Fkt, Schnittstellen, Soll-Bruchstellen, strat. Optionen
- ...

####IT Governance Ausgangsbasis: Gesamtbetrachtung

Betrachtungsweisen: - Information Architecture - Information Quality - Information Security

Herausforderungen

- Trennung Anwendungen und Daten (Ablauf & Aufbau)
- Verschiedene Systemtypen

Allgemeine Hinweise

Heute eingesetzte Standards (flächendeckend, Industrie): Java EE, .NET, TOGAF

Java + .NET

- Zweck: Wie baue ich eine Enterprise-Lösung als Individual-Software
- Inhalt: Bewährte Methoden, Libraries, Best Practices, Do's, Don't's
- Eignen sich grundsätzlich für das selbe (Enterprise Anwendungen)

Individual-Software-Entwicklung

- Local (meist nicht relevant für Architektur)
  - Big: C, C++
  - Small / Medium: Viele Teillösungen, spezialisierte Sprachen, VB, PHP, Phyton
- Distributed (Architekturrelevant)
  - Big / Medium: .NET / Java
  - Small: Swift, Objective C, ...

Java Platform, Enterprise Edition

- Ursprung: Programmiersprache Java (Green Project), Haushaltssteuerung, Objecct Application Kernel (OAK)
- Interpretierbar via VM
- Aufbau Modell
  - Web Browser: HTML
  - Web Server: Servlets, JSPs (Rendering)
  - Application Server: EJBs, POJOs
  - Backend Systems
- Libraries / Methoden um Daten zwischen verschiedenen Tiers auszutauschen

.NET Framework

- Jünger als Java
- Interpretierbar via VM (Common Language Runtime: CLR, Classloader, Managed Native Code, Execution & Security Checks, JIT Compilation with optional verification)
- Aufbau Modell (grundsätzlich analog Java)
- Libraries / Methoden um Daten zwischen verschiedenen Tiers auszutauschen

Information Systems Architecture (ISA) - Historisch

Formale Darstellung Geschäftstätigkeiten in allen Aspekte, einfachere Ableitung Systeme, Verschiedene Perspektiven, Fokus, 3 Teilbereiche der Formalisierung haben sich durchgesetzt: - Prozesse - Organigramm / Aufbauorganisation - Logisches & Physisches Datenmodell - Geschäftsregeln (teilweise) - State / Event Mechanismen (teilweise)

Open Distributed Processing (ODP) - Historisch

Universelles Framework, vollständige Darstellung Anwendung, 5 Viewpoints (Enterprise, Information, Computation, Engineering, Technology), Transitionen zwischen Viewpoints

Common Object Request Broker Architecture (CORBA) - Historisch

Ur-Standard für Verteilte Systeme, Idee: Infrastruktur für Aufbau transparente (nicht sichtbar auf Ebene Einzelobjekt) verteilte Systeme, Verteilung transparent, jedes Objekt ist lokal (auch wenn es physisch nicht lokal ist), Umleitung durch Infrastruktur, Kernstück: ORB (Object Request Broker), Programmiersprachenunabhängig

- Corba Facilities: Branchenspezifisch, eigentlich nicht umgesetzt
- Corba Services: umgesetzt (äquivalente in WebServices zu finden)
  - Naming Service: Objekte via Name finden
  - Life Cycle Service
  - Persistence Service
  - Concurrency Control Service
  - Transaction Service
  - Time Service
  - Security Service
  - Licensing Service

  - ...

Eignet sich als Checkliste für Architektur

Referenzarchitektur (Branchenstandard) - Telecommunications Management Network

Sämtliche beteiligte Komponenten standardisiert (M3100), 90er Jahre, Interoperabilität für Telefonie-Komponenten, Grundelement: Managed Object (4 Charakteristika, Verhalten, Benachrichtigungen, Funktionen, Eigenschaften), Architektur in Funktionale Bereiche aufgeteilt (Security, Accounting, Fault, Performance-Management)

- Funktionale Architektur: Operation Systems Function, Network Element Function, WorkStation Function, ...
- Layering: Business-, Service-, Network, -Element Management Layer, Network Element Layer

Ablauf Verbindung: - 0: "Ich bruache eine Leitung", Verbindung zur Zentrale (Schweiz: 2x16) - 00: Landesvorwahl, Verbindung nach Internationale Zentrale (Schweiz: 3) - 0041: Schliessung Verbindung IZ - 0041 44: 2 Redundante Verbindungen zu 2 zentralen in Zürich

- Zentrale: Netzwerk Element (OSF)
- CDR: Core Detail Record (Kostentransparenz)

TOGAF

- ABB: Architecture Building Blocks
- SBB: Solution Building Blocks
- 3 Ebenen
  - 1: Geschäftsarchitektur (Formale Darstellung Geschäftstätigkeit, Prozesse, Organisation, Treiber / Ziele, logische Informationsobjekte)
  - 2: Informationssystemarchitektur (Systeme, Daten)
  - 3: Technologiearchitektur (HW + Verbindung)

Architecture Operational Systems

Systemtypen

- Anwendungsgebiet

- Funktionaler Umfang
- Standard Software
  - Typischer Aufbau

Typen: - Operational Systems (Unterstützung Leistungsprozesse) - Industry Independant Systems: Standardlösungen, z.B. ERP, CRM, SCM) - Industry Specific Systems: Speziallösungen - Intercompany Systems: Unterstützung Firmenübergreifender Prozesse (z.B. Datenaustausch) - Data Exchange - Electronic Markets - Dispositive Systems (Unterstützung Führungs- / Entscheidungsprozesse, BI, Reporting) - Management Information Systems: Unterstützung Führungskräfte für Entscheidungsfindung - Corporate Planning Systems: Planung / Simulation zukünftiger unternehmerischer Tätigkeiten - Systems for unstructured Data (Anderes) (Anteil in Unternehmen: 50-80%) - Office Automation: Backofficeprozesse - Multimedia Systems - Knowledgebased Systems

Operative Systemtypench

Unterstützung Leistungsprozesse (Kernprozesse Wertschöpfung, z.B. Einkauf, Herstellung, Vertrieb), Unterstützende Prozesse (Personal, Finanzen, Controlling), am "Jetzt" orientiert

- Industry Independant Systems
  - ERP
  - CRM
  - SCM
- Industry Specific Systems
  - Production
  - Retail
  - Banking
  - Insurance
- Intercompany Systems
  - Data Exchange
  - Electronic Markets

Enterprise Resource Planning Betriebliche Planung, Buchführung, Verwaltung Unternehmensressourcen, hervorgegangen aus MRP (Manufacturer Resource Planing)

Grundfunktionalität: - Finanzwesen - Controlling - Herstellung - Materialwirtschaft - Produktionsplanung - Vertrieb - Personalverwaltung

SAP: FI, CO (meisterverkaufte), struktur ähnlich einem Betriebssystem

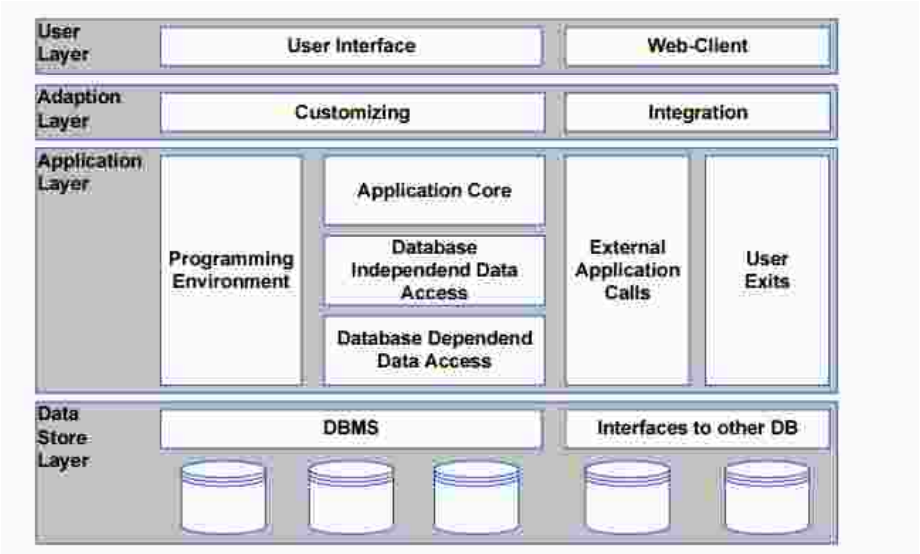
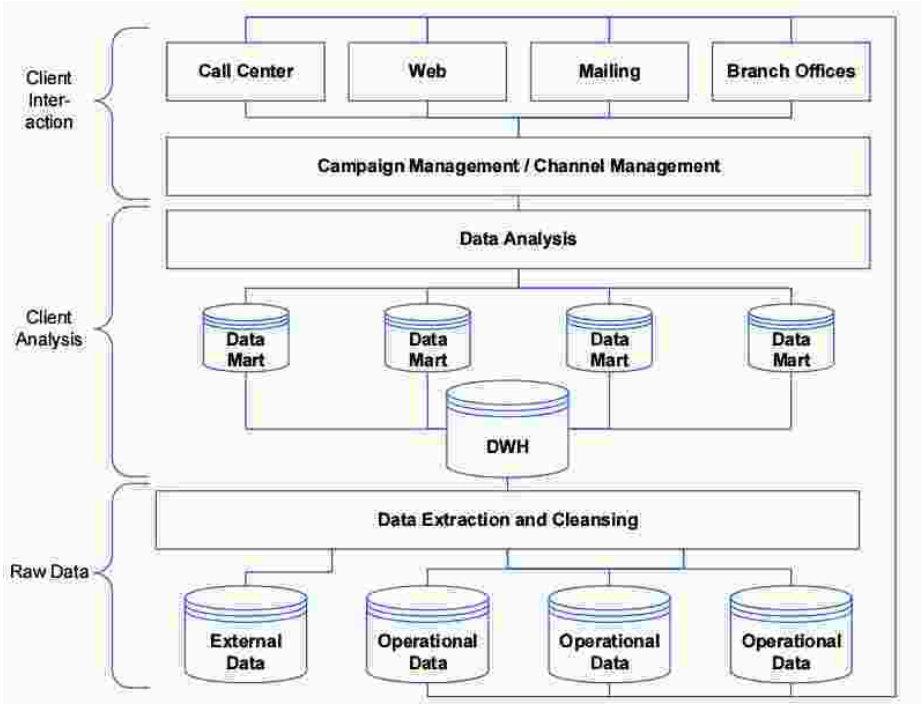


Figure 2:

Customer Relationship Management Markt - Target - Interessierter Kunden - Potenzielle Kunden - Kunde, CRM: Interessierter Kunde bis Kunde

Lead - Opportunity (Sales Chance) - Offering - Contract / Deal

- Sales force
- Siebel (Wirtschaftlich am erfolgreichsten) - Kunden: u.a. ZKB



Standardaufbau
- Operatives CRM (wie beschrieben) - Analytisches CRM (Wenn Kunde Kunde ist, was möchte er noch?) - Kolaboratives CRM (Für Inserateverkäufer wichtig)

Fragen

- Was hat der Kunde bereits gekauft?
- Was braucht der Kunde noch?
- Markt bis Kunde: 1 Jahr (Gesundheit: 3 Jahre)
- 10 Öffentliche Ausschreibungen: 8 Verloren, 2 Gewonnen

Logistik System (Supply Chain Management) Meistens individual Systeme (Fragementierung der Branche), Zwischen Lastwagen und Kunde: ca. 2-5 Zwischenhändler, Korrekte Menge, an bestimmten Ort, in bestimmter Qualität zur bestimmten Zeit abliefern

- Direct Supply Chain (Lieferant - Unternehmen - Kunde)
- Extended Supply Chain (1. Lieferant (Rohstoff) - Endkunde)
- Ultimate Supply Chain (Einfluss Finanzdienstleister, Transport- und Marktforschungsunternehmen)

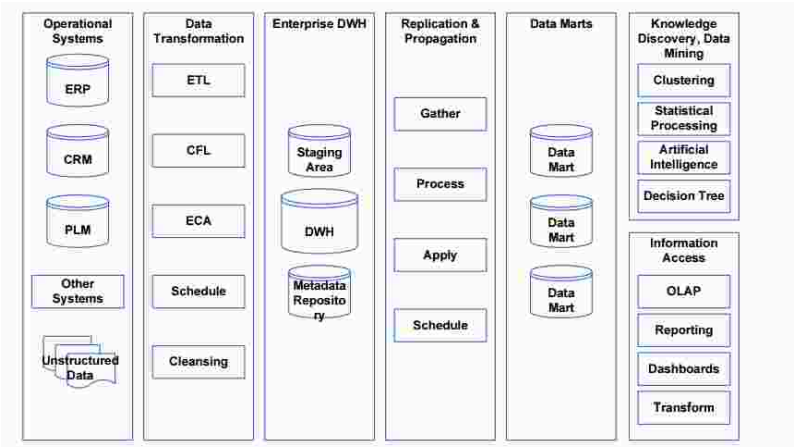
Übereinstimmung Finanzierungs- / Warenfluss

Managementphilosophie: - Alle Beteiligten verhalten sich integriert (miteinander) - Informationsaustausch zwischen allen Beteiligten - Risiko- / Gewinnverteilung - Zusammenarbeit - Ein Ziel: Kundenzufriedenheit - Prozessintegration - Langzeit-Partnerschaft

Standardisierungs: SCOR (Supply-Chain Operations Reference-model), Supplier, Enterprise, Client - Kernprozesse: Plan, Source, Make, Deliver, Return - Plan: Gesamtplanung Prozesse, Verwaltung Geschäftsregeln, Performance, Inventar, Transportkapazitäten, Regulatorien, Risiken - Source: Terminierung / Prüfung Lieferung, Auslösung Zahlung, Auswahl Lieferanten - Make: Terminierung Produktion, Produktfreigabe, Verpackung, Schlussprüfung - Deliver: Verwaltung Bestellprozess, Auswahl & Kontrolle Lieferung - Return: Rückgabe, Freigabe Rückgabe, Abwicklung Unterhaltsarbeiten / Reparaturen, Behandlung Garantiefälle

Dispositive Systemtypen

- Management Information Systems
  - DWH
  - EDWH (Enterprise DWH)
  - CPM (Corporate Performance Management System, KPIs)
- Corporate Planning Systems
  - Mining: Explorative Analyse
  - Analytics: Aufbereitung
  - Simulation



Systems for unstructured Data

- Office Automation: Backofficeprozesse

  - Communication
  - DMS: Document Management System
  - WMS: Workflow Management System
  - CMS: Content Management System
- Multimedia Systems
  - Video Converencing
- Knowledgebased Systems
  - Expert Systems
  - Language Processing

CSCW (Computer Supported Cooperative Work) Zusammenarbeit von Menschen in Gruppen unterstützen, E-Mail, Workflow-Systems, Video Conferences, ...

- Immer für Gruppenarbeit (2 oder mehr interagierende Personen, die einander beeinflussen) ausgelegt.

Enterprise Content Management Architektur: - Clients: Portals & Web Applications, LOB & ISV Solutions, Desktop Applications - Services: Capture, Classification & Taxonomy, Records-, Document-, Image-, Archive-, Web Content-, Forms-, Reports-Management, Search & Discovery, Content Centric BPM, Collaboration - Repositories: File Server, Image / Movie DB, Archive

```
- Suchbarkeit
- Workflow
- Verknüpfung: Dokumente mit Metadaten, Verknüfpung zu Geschäftsfall
- Sicherheit
- Nachvollziehbarkeit
- Versionierung
- Sicherung
- Mehrsprachigkeit
- Komposition
```

CQ - Distributer: Dynamisch Inhalt herstellen - Render: Aus dynamischen Inhalten statischen Inhalt erzeugen - Guardian: Überwachung System

Hints

Neuen MA einstellen: ca. 50'000 .-

Zusammenfassung

- Verschiedene Systemtypen
- Breites Spektrum im operativen und distributiven Bereich, sehr wenige Daten (nur strukturierte), 80 % der Systeme für 20% der Daten zuständig
- Wenige Anbieter für Bearbeitung unstrukturierter Daten
- Viele Anbieter für Bearbeitung strukturierter Daten
- Gesamtarchitektur besteht aus mehreren Vieews / Blickpunkten
- Neues System: Was ist es für ein System? Wie haben es die anderen gemacht?

Service Oriented Architecture

Ganzheitliche Betrachtung IT-Systemlandschaft, Unterstützungsfunktion betriebliche Prozesse, Standardarchitektur, logische Teilung Applikationen, Integrationsmechanismen, Dienste und Orechestrierung

Ganzheitliche Betrachunt IT-System Architektur kann auf Entitäten, Objekten oder Services aufgebaut werden. SOA: Betrachtungsweise IT-Landschaft, im Bereich DevOps einzige auf Basis Services, Service hat Service-Implementation (Wird geleistet), technisch oder "Mensch" ist egal, Daten: elektronische Daten oder Informationen welche für Umgang mit Service benötigt werden.

- Service: Standardisierte Darstellung von Funktionalitäten
- Service Oriented Computing (SOC): Paradigma, Dienste als Basis für Applikationen, Basisdienst + Basisoperationen
- Service hat definierte Schnittstelle (von aussen zugreifbar)
- Einzige Interaktion via Schnittstelle
- Keine Vererbung
- Nur Abhängigkeitsbäume
- Service Contract: Informelle Spezifikation Service-Funktionalität
  - Praxis: Keine Möglichkeit zur formalen / maschinellen Spezifikation, Ausnahme: Onthologien / Semantisches Web (Nur Forschung)
- Bereitgestellt von Service Provider
- Servicenehmer: Software, die einen Dienst in Form eines Interface Proxy intern darstellt.

Dienste statt Applikationen
Wiederverwendung ganzer Systeme durch Kapselung und definierte Service-Schnittstellen

Service Oriented Architecture: Basis: Kein Beginn auf grüner Wiese, Weiterverwendung bestehender Landschaft / Systeme, lässt Anwendungen länger

leben -> zementiert Anwendungen, Inflexibilität bezüglich neuen Technologien / innovationen, Architektur ist nicht hierarchisch, Logik wird in 2 Schichten geteilt: 1x statische Logik auf Ebene Application, 1x veränderliche / dynamische Logik (z.B. Prozesse, Regeln) in der orchestration Schicht, Einsatz standardisierter Schnittstellen

Layering: - Managed Services (Certification, Rating, SLA's, Assurance, Support): fast nur organisatorische Aufgaben - Composite Services (Coordination, Conformance, Monitoring, Planning): Dienste für Bereitstellung zusammengesetzter Services, organisatorische Aufgaben - Basic Services (Publication, Discovery, Selection, Binding, Capability, Interface, Behaviour, Quality of Service): Basis-Services für Komposition weiterer Services

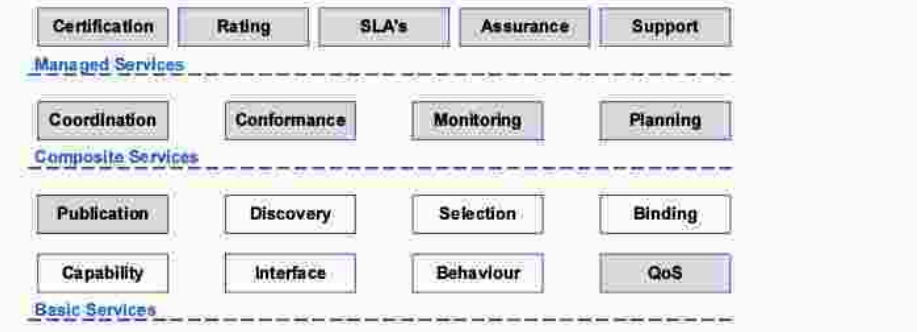


Figure 3:

Eigenschaften:
- Trennung Business-Logik in dynamische (Prozesse / Regeln) und statische (applikatorische Umsetzung) Bereiche - Integration Architecutre: logische Komponente, nicht unbedingt durch ESB, minimal: Netzwerk - Weiterverwendung: integraler Bestandteil - Service Layer: SOAP und WSDL, weitere nicht zwingend

Referenzmodelle: - W3C: Fokus: globales Netz an Funktionalitäten, mehrere Metamodelle (Policy-, Action-, Message-, Ressource-Modelle), wie interagieren Services? Servie Oriented Model - GeneriCo: Idee: standardisierte Branche, Abbildung standardisierter Prozesse, Referenzarchitektur für jedes Unternehmen (wenn generisch genug), besteht aus Enterprise Applications, Security Layer, External Access via Portals, Custom Applications -> führt zu Service-Landkarte (auf Basis generalisierter Architektur), ähnelt Architektur eines Legacy-Systems (da generisch), Abbildung von Prozessen 1:1 auf Services möglich

SOA Komponenten

Nicht betrachtet in Kurs: Virtualized Infrastructure, Presentation

- **Presentation:**

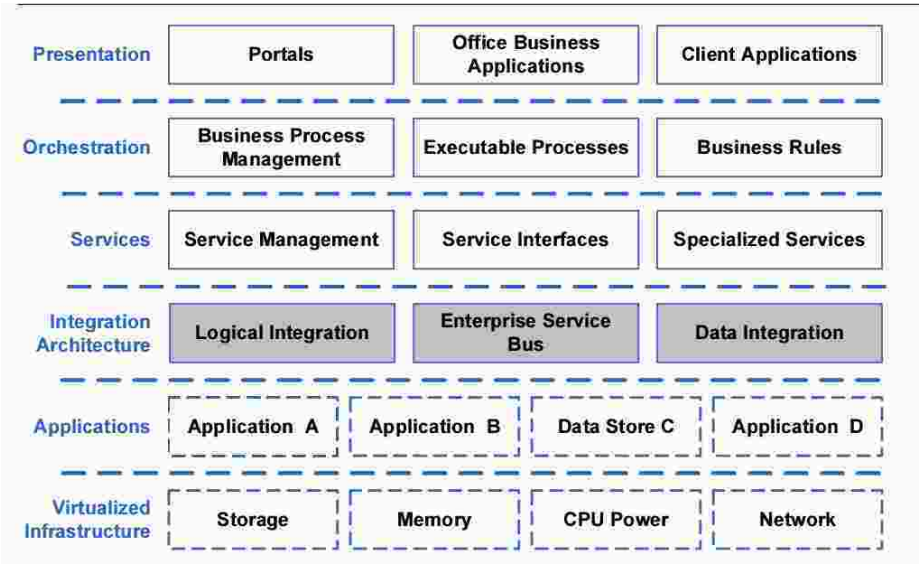
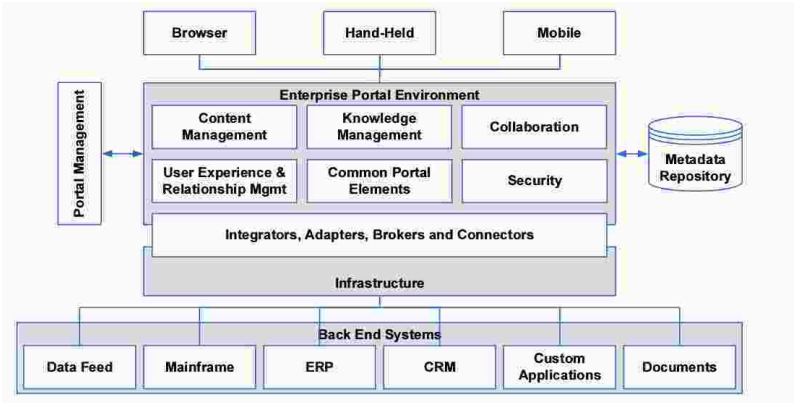


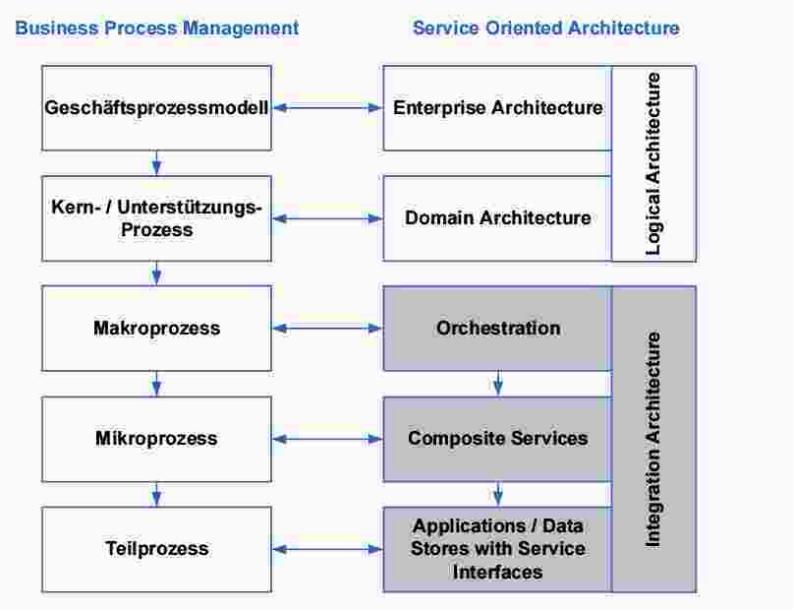
Figure 4:

  - Portale



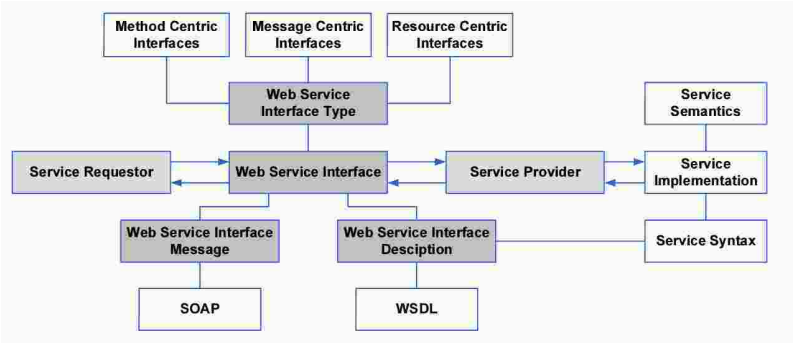
  - Referenzarchitektur: outdated, nicht mobile tauglich (zu umfangreich), Trick: verschiedene UIs für verschiedene Geräte, Mehrfachimplementation Portal, Alternative: HTML5
  - Office Business Applications: Dokument als Ressource
  - Client Applications: z.B. Java oder .NET

- **Orchestration:** bildet Teil der Businesslogik ab, dynamischer Teil
  - Business Process Management: Abbildung Enterprise Prozess auf technische ausführbare Organisation der Gesamtarchitektur, Domänenarchitektur: nur in Konzernen, pro Abteilung / Bereich



  - Executable Processes: Formale Beschreibung eines Prozesses zur Steuerung eines IT-Systems (BPMN, BPEL), Idee: Modellierung auf abstrakter Ebene, Steuerung Process-Engine / Interpreter, jeder Prozess ist wiederrum ein Service, beliebige Verschachtelung, SOAP, WSDL, BPEL, Integration z.B. via ESB und Process Engine
  - Business Rules: Rule Engines, Einsatz in 3 Bereichen: Konsistenzregeln, Fakten: Steuerung komplexer Aktionen auf Basis bestimmter Fakten, Aktionsregeln: Steuerung von Aktionen als Reaktionen auf bestimmte Ereignisse - Organisation z.B. nach Business Events zur Steuerung der Prozesse

- **Serivice-Ebene / Service Management**
  - Service Management
  - Version & Status Mgmt: Versionierte Schnittstellen, gleichzeitig in betriebliche
  - Testing
  - Billing / Verrechnung
  - Client-info
  - Administration: Service-Owner / -Manager, etc.



  - Service Interfaces
  - Service Semantics: inoffizieller Vertrag
- Integration Architecture
  - Später detailliert

3 Arten von Service-Schnittstellen
- Call & Return Schnittstellen (Aufruf, einfach, impliziert Funktionalität, getCustomer) - Message-Oriented-Interface (umfangreiche Meldung einpacken, an Service senden -> process this, Service führt Verarbeitung durch und gibt strukturierte Antwort zurück) - Ressource-Oriented-Interfaces (Manipulation auf Ressource hat Service Aktionen zur Folge, abhängig von Ressource, z.B. REST)

3 Arten von Prozessen:
- Human-Centric-Process: Abhängig von Interaktion Menschen (z.B. Approval-Prozesse) - System-Centric-Processes: Nur Systeme (z.B. Booking Confirmation oder Backup) - Document-Centric-Processes (z.b. Steuererklärung)

Fragen: Noch zeitgemäss? alternative? (iot, cloud), Micro-Service-Architecture

Wichtigste Punkte

- Schichten: Presentation, Orchestration, Services, Integration, App / Data
- 3 Grundeigenschaften: Standardisierte Services / Produkte (O, S, I, A/D), Trennung dynamische / statische Logik, Weiterverwendung bestehender Services (nicht Wiederverwendung)
- Typische Komponenten: O: Rule Engine, Process Engine - S: Aufbau Service Schnittstelle (Interface, Impl, Daten, informeller Vertrag, Semantik: informell, Syntax: formel), Cross Cutting Services - I: später, Choreographie

Distributed Real Time Systems

Entwicklung (EA-Games): ca. 20 Millionen, Real-Time oder Near-Real-Time, Simpler Unterschied zu normalen Anwendungen: Unabhängige Interaktion-

szyklen, Simulationstecnik ist älter als Game-Engines

Distributed Real Time Applications

- Simulations
- Virtual Environements
- Computer Games

Urform der Game-Engine: unreal

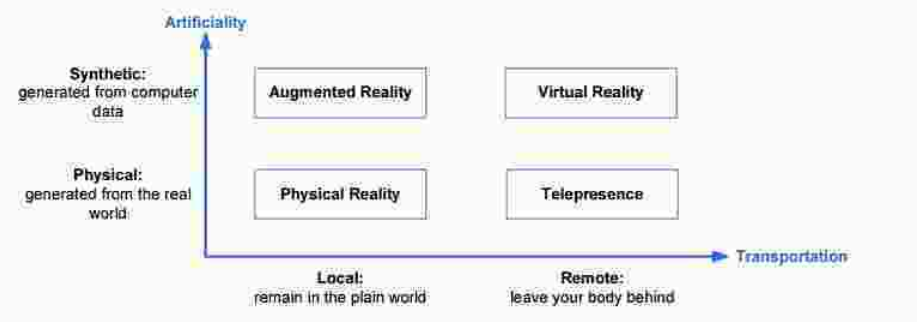


Figure 5:

- Augmented Reality: Erweiterte Realität, visuelle Überlagerung in Echtzeit
- Virtual Reality: berechnete Realität
- Physical Reality: Reale Welt
- Telepresence: Durch Verwendung von Technologie erscheint ein Individuum lokal präsent, obwohl physisch an einem anderen Ort.

Wann ist ein Spiel ein Spiel: 1. Es gibt Regeln (können nicht verletzt werden). 2. Es gibt Teilnehmende. Anzahl zu bestimmter Zeit gegeben. 3. Man muss gewinnen können.

Allgemeine Architekturkriterien

- Performance: Zwingend in Echtzeit
- Security: Schutz des Individuums, der verwendeten Daten
- Scalability: Wechselnde Anzahl User, kein Einfluss auf Performance
- Availability: 7x24h
- Usability: Spezielle Peripheriegerät in 3D Environement
- Flexibility: dynamisch neue virtuelle Umgebungen verwalten können
- Portability: Unterschiedliche Geräte

- Reusability: Replizierbarkeit verschiedener Schichten / Komponenten
- Testability: Spezielle Testgeräte und Testmaschinen
- Separation of Concern: Klare Trennung einzelner Komponenten, Vermischung führt zu Performance Einbussen
- Comprehension: Hohe Verständlichkeit der Aufgabe einzelner Komponenten
- Correctnesss / Completeness: Engpässe umgehen können, Verlust ganzer System- / Infrastrukturteile ausgleichen können
- Referencial Transparency: Hohe Granularität und Verteilung einzelner Komponenten erfordert hohe Abstraktion der Funktionsweise
- Buildability: Endgeräte einfach und billig
- Coupling: banal
- Cohesion: banal, Unterordnung an Gesamtanforderungen

Zentrale Faktoren

Hardware der Endgeräte + Beschaffenheit des Netzwerkes

Hintergrund

SIMNET: Erstes voll einsatzfähige Virtual Reality System überhaupt, heute werden diese Distributed Interactive Simulators (DIS) genannt, unterstützt werden bis zu 10'000 menschliche Spieler sowie 9000 Software Avatare

High Level Architecture

Einleitung

Generelle Architektur für Simulationssysteme, IEEE Open Standard 1516, Baseline Definition: HLA Rules (10 Regeln, definierten Verhalten des Systems), HLA Interface Specification (SST zwischen HLA Federates und Runtime Infrastructure), HLA Object Model Template (OMT): Voralge für Spezifikation Objektmodell, Simulation: Alle haben gleiche Sicht auf Zeit.

Runtime Infrastructure (RTI): Support Utilities, Simulation, Interface to life players

Federation

Sammlung von Federates, die über RTI ineragieren, einen Simulationslauf durchführen (Federation Execution), Federates: Komponenten

HLA Rules

Federation Rules: - Müssen mit Hilfe Object Model Template dokumentiert werden - Instanzierung von OM Objekten nur in Federates, nicht in RTI - Datenaustausch dzwischen Federates erfolgt durch RTI - Ein Attribut einer Objektinstanz darf nur einem Federate zugeordnet sein

Federate Rules: - Federades werden durch Simulation Object Model beschrieben - Federates publizieren Objektattribute in ihrem SOM, sie können diese auch wieder entfernen, Messages zwischen einzelnen SOM - Die Bedingungen für Aktualisierung von Objektattributen sind veränderbar - Federates übernehmen die zeitliche Koordination mit anderen Federates für den Datenaustausch

HLA Interfaces

Federate meldet sich bei RTI an

Runtime Infrastructure

Andere Architekturen

Distributed Interactive Simulation, Parallel Discrete Event Simulation

Networked Virtual Environements

Virtuelle Umgebungen, Echtzeit, spezifische Art von simulationen, grundsätzlich keine Unterschiede zur Architektur von Simulationssystemen, Anwendungen: Games, Telemedizin, virtuelle Laboratorien, Simulationen

Eigenschaften: - Alle Anwenderinnen sind in demselben virtuellen Raum präsent (evtl. nicht sichtbar) - TeilnehmerInnen sind durch Avatare repräsentiert - Aktionen anderer TeilnehmerInnen werden in Echtzeit wahrgenommen - TeilnehmerInnen können miteinander kommunizieren - Interaktion mit anderen TeilnehmerInnen und virtuellen Objekten

Rahmenbedingungen

- Zielsysteme
  - Anzahl TeilnehmerInnen
  - Kompleixtät Objekte und Verhaltensweisen
  - Ausmass Interaktion
- Netzwerk

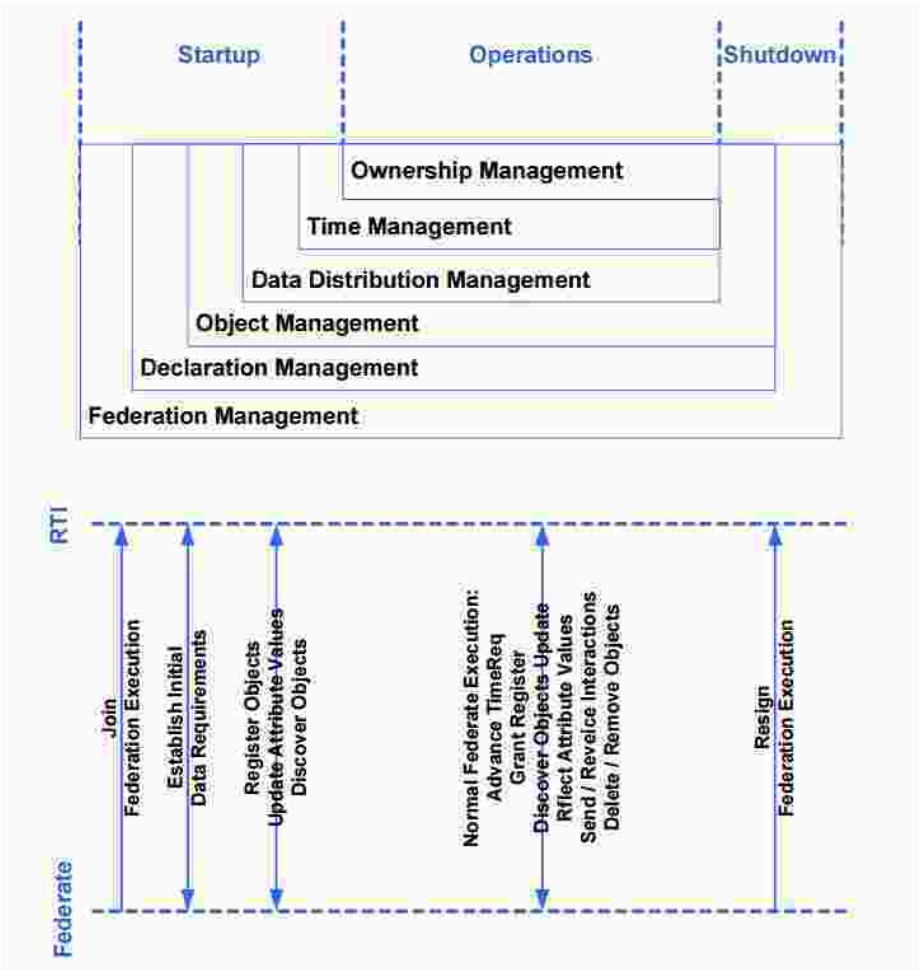

Startup
Operations
Shutdown
Ownership Management
Time Management
Data Distribution Management
Object Management
Declaration Management
Federation Management
RTI
Federate
Join Federation Execution
Establish Initial Data Requirements
Register Objects
Update Attribute Values
Discover Objects
Normal Federate Execution:
Advance TimeReq
Grant Register
Discover Objects Update
Rflect Attributo Values
Send / Reveice Interactions
Delete / Remove Objects
Resign Federation Execution

Figure 6:

– Connection zwischen Infrastrukturkomponenten
– Unicast: von bestimmten Sender an best. Empfänger
– Multicast

Toolkits und integrierte Architekturen

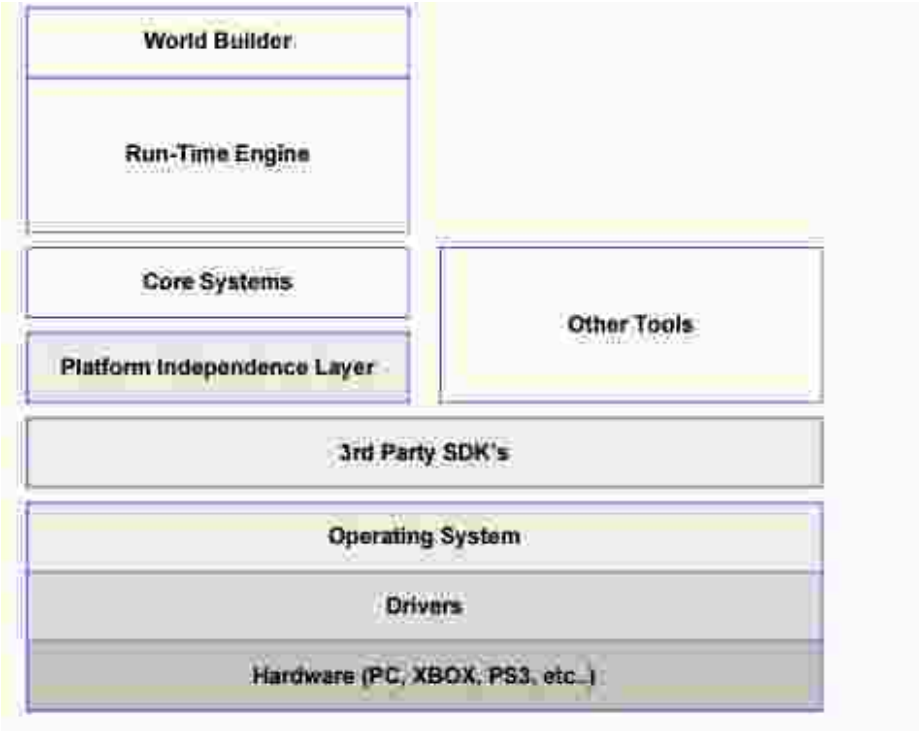


Figure 7:

Toolkits-basierte Architekturen umfassen Werkzeuge und Libraries zur Erstellung und Interaktion mit virutellen Umgebungen. Umfassen: Kontrolle der Objekte in Umgebung, Bewegung Repräsentation des Avators, Dynamische Viewpoints, Objekt-Relationship, Display-Management, Ressourcen-Synchronisation

Integrierte Architekturen: Liefern voll funktionsfähiges System, welches alle Basisaufgaben eines NVE realisiert, Grundsatz für Realisierung: Komponenten ersetzen oder ergänzen

Architekturen

Shared Scalable Server Welt zentral auf Server, nicht auf den Clients, eher für Online-Games

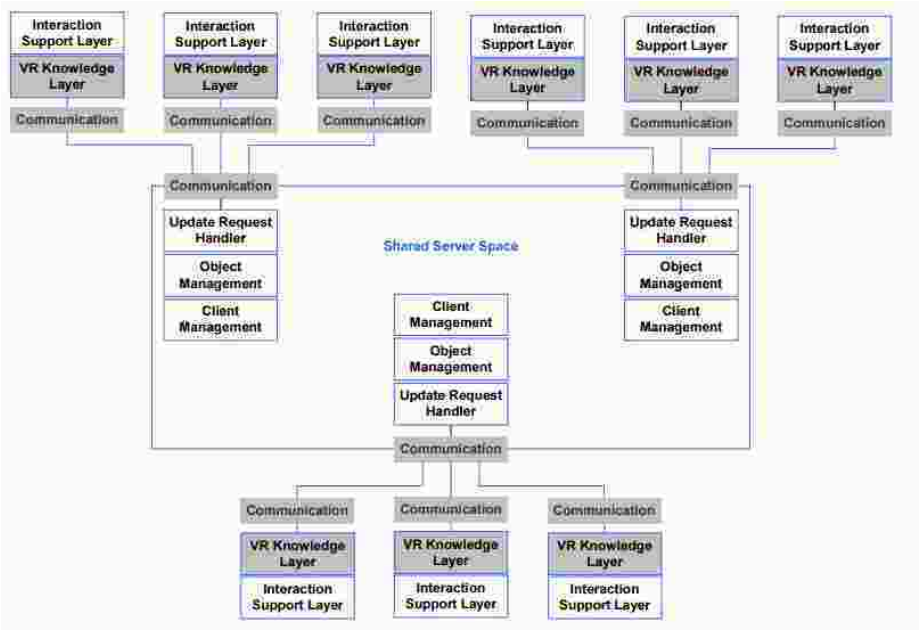


Figure 8:

High Interactive Distributed Real-Time Architecture (HIDRA)

Verteilung auf Clients, Message-Funktionalität auf Server, mehr für Einzelplatzspiele

Funktionale Architektur

Multiplayer Computer Games

- Networking Issues:
  - Bandbreite
  - Latenz
  - Rechenleistung
  - Message Compression and Aggregation
  - Interest Management (Area of Interest)
  - Dead Reckoning: Darstellung wird interpretiert ohne Update
- Scheduling
  - Lineares Scheduling
  - Priority Round-Robin Scheduling

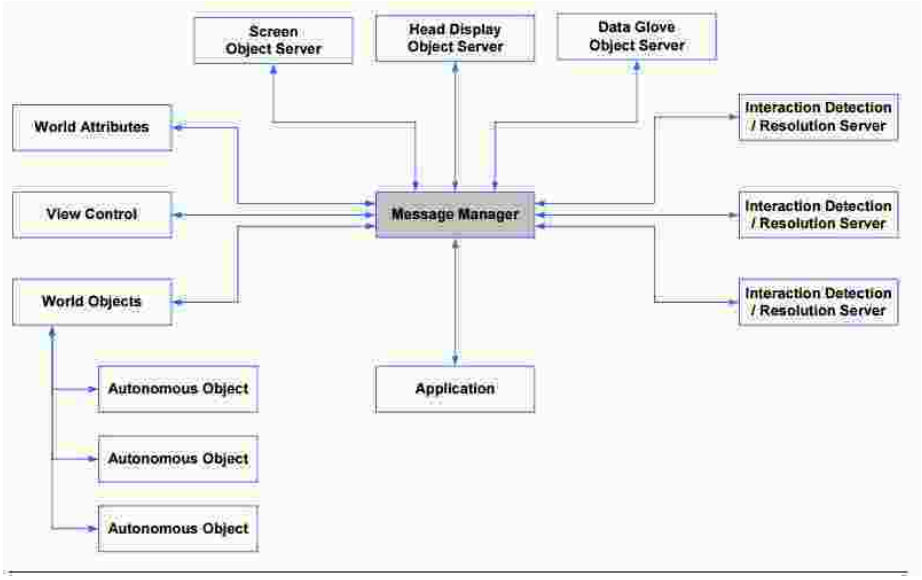


Figure 9:

  - Context Dependend Scheduling
  - Temporal Bounding Volume: Objekt in nichtsichtbarem TBV muss nicht nachgeführt werden.
  - Update Free Regions: Update nur notwendig, bei welchen Objekte sichtbar sind.

- Referenzarchitektur ## Zusammenfassung Einfluss auf Anforderungen und somit Lösungsarchitektur: Spielregeln (Spielelogik), Nähe zur Realität / Darstellung der Physik (Performance Anforderungen), Ursprung aus (Kampf-)-Simulationstechnik, Erstes System: SIMNET, führte zu erster standardisierter Architektur (Wichtig: Taktgeber - Runtime Engine, Federate: Ein- / Ausklickbarer Bestandteil der Welt)

2 verschiedene Architekturen für Networked Virtual Environements (Shared Scalable Server für Online-Games, High Interactive Distributed Real-Time Architecture (HIDRA) - eher Einzelplatzgames), Funktionale Architektur, Historisches Datum: 15.Nov. 2001 (xbox), 30. Sept 2003 (PC), Unterscheidung erlebter Welt und physikalischer Simulation Gesamtwelt

Prüfungsvorbereitung

Ca. 15 Fragen (~2 Stunden) - Skizzieren Sie die Architektur einer HIDRA auf einer SOA - Was ist in TOGAF ein Solution Building Block? - Realisiert einen

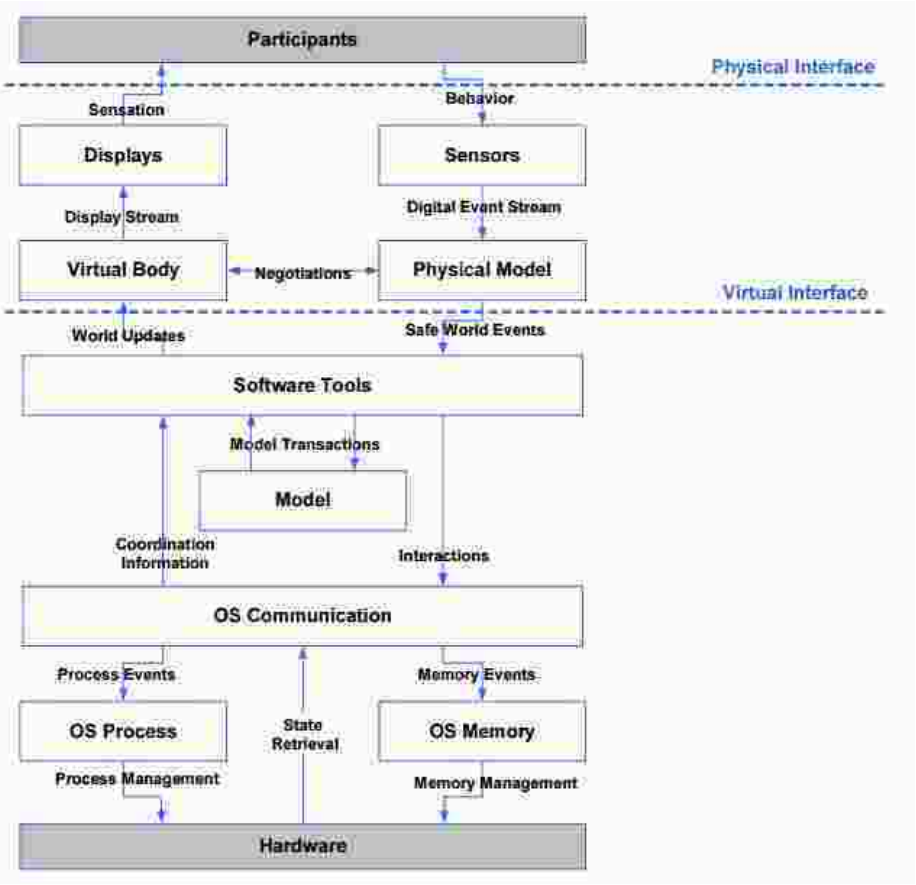

Participants
Physical Interface
Sensation
Behavior
Displays
Sensors
Display Stream
Digital Event Stream
Virtual Body
Negotiations
Physical Model
Virtual Interface
World Updates
Safe World Events
Software Tools
Model Transactions
Model
Coordination Information
Interactions
OS Communication
Process Events
Memory Events
OS Process
State Retrieval
OS Memory
Process Management
Memory Management
Hardware

Figure 10:

oder mehrere ABB, SBB ist konkret, ABB ist abstrakt - Welchem Systemtyp würden Sie Facebook auf Architektursicht zuordnen? - Mehrere Antworten möglich, gute Begründung (Systems for unstructured Data - Social Systems) - Erklären Sie den Begriff Virtual Reality (Auswahl aus mehreren Definitionen) - Begründung - Welches sind die heute wichtigsten Architekturstandards? - Java, .NET, Client-Server - Welcer Architekturstiel kann mit welcher Standardarchitektur umgesetzt werden?

Systemarchitektur

Einführung

Pro Firma:

- Client-Infrastruktur (ca. 50 Programme gemanaged), Server-Infrastktur:
- In der CH: meistens doppelte Buchhaltung (oft SAP FI und CO)
- ERP-System (oft SAP, nicht immer)
- 1-2 Zentrale Systeme für Hauptwertschöpfung (z.B. Steuerung, Produktion, ...)
- CRM
- 7-12 Zentrale Entitäten (Kunde, Offerte, etc.), werden zwischen Systemen hin und her geschoben
- Digitalisierung: Bestehende + Zukünftige GP durch IT unterstützen, Digitalisierung = IT-Unterstützung
- Anbindung von Things 7 Netzwerken via Netz an die Firma, kleiner Aufwand, < 1\$, weitgehende Konsequenzen
- Wirkung von Aussen auf die Firma (Fahrzeuge: Wegoptimierung, Umweltinformationen, Soziale Netzwerke)
- Klar Strukturierte IT: nicht geeignet, Grösstes Problem: Systeme in die Cloud (Sicherheit), Umbau der Architektur notwendig, Ziel: Netz der Netze
- Heute 3 Optionen:
    1. Alles selber machen (intern), Infrastruktur: Private Cloud, Big Data: Aufbau Infrastruktur, IOT: ?, Social Networks: via Big Data)
    2. Alles rausgeben (Clients intern, Rest in der Cloud), Problem: Datenverlust / -sicherheit bei Transfer

3. Kobination, intern: alles nahe beim Kernbusiness

- Heute: Innovation von Aussen (CES Las Vegas)
- Creative Destruction: Bestehende Systeme zerstören, um Platz zu machen

Probelm: Strukturierte Daten (15%), Restliche irgendwo verteilt (Nicht strukturierte Daten), Big Data: Versuch einer Lösung, Informationsgewinnung über mehrere Stufen,

Architektur: Was gehört wohin, Ordnungsrelation, Systeme oft grundlegende Architektur-Ideen, Standards, Strukturierung von individual, Standard SW, Hardware

Agenda / Ablauf

- System-Engineering
- Basis: Allgemeine Systemeigenschaften, allgemeine Architektur-Stiele
- Standards
- TOGAF: Längere Übung
- Enterprise Architecture Patterns
- Wichtigste Standard-Systeme (ERP, CRM, ...)
- Service Oriented Architecture
- Distributed Realtime-Systems
- Cloud-Computing
- Internet of Things

ArchitekturüBuilding

Ablauf

1. Einteilen
2. Unternehmen oder Abteilung
3. Informationen beschaffen
    - Organigramm
    - ...
4. Programme: Zeichnen + Tabellen (Gliffy, OO Calc)
5. Ordner
    - Zeichnungen für:
    - Organisation
    - Prozesse

- Anwendungen
- Tabelle: ABB's

6. Vorgehen
    1. Organigramm beschaffen (Top-Level: Abteilung)
    2. ABBs aus Organigramm ableiten
    3. 3 Ziele (selbstdefiniert)
    4. Prozesse definieren (Top-Level)

    - Mgmt Prozesse
    - Leistungsprozesse
    - Support Prozesse

https://www.3m5.de/fileadmin/user_upload/blog/2015/04/prozesslandkarte.jpg

http://www.mgm-cp.com/image/image_gallery?img_id=18927&t=1277716855489

http://www.wieczorekit.de/wp-content/uploads/2012/09/soa-%C3%BCbersicht.png