









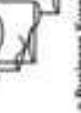


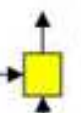




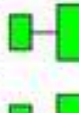
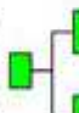



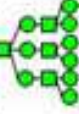

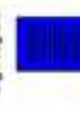






	DATA	What	FUNCTION	How	NETWORK	Where	PEOPLE	Who	TIME	When	MOTIVATION	Why	
SCOPE (CONTEXTUAL)	List of Things Important to the Business		List of Processes the Business Performs		List of Locations in which the Business operates		List of Organizations Important to the Business		List of Events/Cycle Significant to the Business		List of Business Goals/Strategies		Planner
BUSINESS MODEL (CONCEPTUAL)	Entity = Class of Business Thing e.g. Semantic Model		Process = Class of Business Process e.g. Process Model		Node = Major Business Location e.g. Business Logistics System		People = Major Organizations e.g. Work Flow Model		Time = Major Business Event/Cycle e.g. Master Schedule		End/Means = Major Business Goal/Strategy e.g. Business Plan		Owner
SYSTEM MODEL (LOGICAL)	Entity = Data Entity Relationship = Data Relationship e.g. Logical Data Model		Process = Application Function IO = User views e.g. Application Architecture		Node = Distributed System Architecture e.g. Distributed System Architecture		People = Human Interface Architecture e.g. Human Interface Architecture		Time = System Event Cycle = Processing Cycle e.g. Processing Structure		End = Business Objective Means = Business Strategy e.g. Business Rule Model		Designer
TECHNOLOGY MODEL (PHYSICAL)	Entity = Table Segment Relationship = Key/Pointer e.g. Physical Data Model		Process = Application Function IO = Data Elements/Data e.g. System Design		Node = Hardware/Software Link = Line Specification e.g. Technology Architecture		People = User Work = Screen Format e.g. Presentation Architecture		Time = Execute Cycle = Component Cycle e.g. Control Structure		Time = Execute Cycle = Component Cycle e.g. Control Structure		Builder
DETAILED REPRESENTATIONS (OUT OF CONTEXT)	Entity = Field Relationship = Address e.g. Data Definition		Process = Language Statement IO = Control Block e.g. Program		Node = Address Link = Protocol e.g. Network Architecture		People = Identity Work = Job e.g. Security Architecture		Time = Interrupt Cycle = Machine Cycle e.g. Timing Definition		End = Subordinate Means = Step e.g. Rule Specification		Sub Contractor
FUNCTIONING ENTERPRISE	E.g. DATA		E.g. FUNCTION		E.g. NETWORK		E.g. ORGANIZATION		E.g. SCHEDULE		E.g. STRATEGY		

Software Architektur - Standards

Daniel Liebhart, 21.9. & 19.10.2015

Verfasser: Daniel Liebhart
11. Auflage: 2015
Version 11.0

© by Daniel Liebhart

Inhalt

Referenzen und Abkürzungen	3
Einleitung.....	6
1.1 Standards als Wegweiser.....	6
1.2 Architekturstandards	6
1.2.1 Client Server	6
1.2.2 Common Object Request Broker Architecture	6
1.2.3 Information System Architecture	6
1.2.4 Open Distributed Processing.....	6
1.2.5 Telecommunications Management Network	6
1.2.6 Weitere Standards.....	7
Client - Server.....	8
2.1 Einleitung	8
2.2 Aufbau des Modells.....	8
2.3 Definitionen Client-Server	8
2.4 Einbettung	9
2.5 Dreistufiges Strukturmodell	10
2.6 Die Rolle des Netzwerkes in einem Client Server Modell.....	11
Java Platform, Enterprise Edition (Java EE)	12
3.1 Einleitung	12
3.2 Aufbau des Modells.....	12
3.3 Architektur von Java EE 7	13
3.4 Die Dienste von Java EE 7.....	13
3.5 Zentrale Konzepte.....	14
3.6 Java EE Application Servers	15
.NET Framework.....	16
4.1 Einleitung	16
4.2 Framework	17
4.3 Common Language Runtime (CLR)	18
4.4 Base Class Library	19
4.5 Microsoft .NET Servers	19

4.6	Version 4.5 – Heute .NET 2015 / .NET Framework 4.6.....	19
4.7	Gartners Vergleich	20
Common Object Request Broker Architecture.....		21
5.1	Einleitung	21
5.2	Definition	21
5.3	CORBA Referenzmodell	21
5.4	Corba Services.....	22
5.5	Corba Facilities.....	22
5.6	Object Request Broker	23
Information Systems Architecture		24
6.1	Einleitung	24
6.2	Das ISA Framework	24
6.3	Perspektiven und Fokus.....	25
6.3.1	Perspektive	25
6.3.2	Fokus	25
6.4	ZIFA	25
Open Distributed Processing		26
7.1	Five Viewpoints	26
7.2	ODP Vorgehen.....	27
TMN: Telecommunications Management Network		28
8.1	TMN Definitionen.....	28
8.1.1	Synonyme für TMN	28
8.1.2	Funktionale Bereiche.....	28
8.1.3	Managed Object.....	28
8.2	Funktionale Architektur.....	29
8.3	TMN Layers.....	30
8.4	Innerer Aufbau eines OSF.....	31
8.5	Physische Architektur.....	32
Anhang A: Architekturentwicklung mit TOGAF.....		33

Referenzen und Abkürzungen

Referenzen

[Box 2002]	D. Box: Essential .Net Volume 1: The Common Language Runtime, Addison-Wesley 2002
[Chinnici, Shannon 2009]	R. Chinnici, B. Shannon: Java Platform, Enterprise Edition (Java EE) Specification, v6, Sun Microsystems, 13.1.2009
[CIMOSA 1993]	CIMOSA Open System Architecture for CIM, Research Reports ESPRIT, Springer Verlag, 1993
[DoDAF 2009]	US Departement of Defense: DoD Architecture Framework Version 1.5 & 2.0 28 May 2009
[Fliss 2005]	Ch. Fliss: Vergleichsmethoden für Vorgehensmodelle, Technische Universität Dresden 23. März 2005
[Guy 1991]	N. K. Guy: Community Networks: Building real communities in a virtual space?, Master of Arts Thesis, University of Waterloo, 1991
[Hornford et al. 2011]	D. Hornford, T. Paider, Ch. Forde, A. Josey, G. Doherty, C. Fox: TOGAF® Version 9.1, 2011, The Open Group
[Nell 1997]	J.G. Nell: An Overview of GERAM, ICEIMT'97 International Conference on Enterprise Integration Modelling Technology 1997
[ODP 2006]	ITU-T: Reference Model for Open Distributed Processing Part 1 - Part 4 ITU-T X.901 / ISO 10746-1
[OMG 2002]	OMG: Common Object Request Broker Architecture: Core Specification, Version 3.0, 6.12.2002
[Scheer, Jost 2003]	A.-W. Scheer, W. Jost: Aris in der Praxis, Springer Verlag 2002
[Schussel 1995]	G. Schussel: Client/Server: Past, Present and Future , 1995
[Schwichtenberg 2008]	H. Schwichtenberg: Microsoft .NET 3.5 - Crashkurs: Presentation, Communication, Workflow, Visual Basic 2008, Visual C# 2008, ADO.NET 3.5, ORM, LINQ, WPF, WCF, Microsoft Press 2008
[Sellin 1995]	R. Sellin: TMN Die Basis für das Telekom-Management der Zukunft, R.v.Decker's Verlag Heidelberg 1995
[Wiggins 2003]	D. Wiggins: .Net vs. Java: Competition or Cooperation, Gartner Audio Teleconference, 6. March 2003
[Zachmann 1987]	J.A. Zachmann: A Framework for Information Systems Architecture, IBM Systems Journal, vol. 26, no. 3, 1987

Abkürzungen

ADO	ActiveX Data Objects
ARIS	Architecture of Integrated Information Systems
ASP	Active Server Pages
BML	Business Management Layer
CIMOSA	Computer Integrated Manufacturing Open System Architecture
CORBA	Common Object Request Broker Architecture
DAS	Directory Access Function
DCF	Data Communication Function
DCN	Data Communication Network

DGSA	DoD Goal Security Architecture
DLL	Dynamic Link Library
DODAF	Departement of Defense Architecture Framework
DOM	Document Object Model
DSF	Directory Service Function
EJB	Enterprise JavaBean
FEAF	Federal Enterprise Architecture Framework
GIS	Geographical Information Systems
HTTP	Hypertext Transfer Protocol
ICF	Information Conversion Function
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineers
ISA	Information System Architecture
ITU	International Telecommunication Union
JACC	Java Authorization Contract for Containers
JASPIC	Java Authorization Service Provider Contract
Java EE	Java Enterprise Edition
Java ME	Java Micro Edition
Java SE	Java Standard Edition
JAXR	Java API for XML Registries
JAX-RPC	Java API for XML-Based Remote Procedure Calls
JAX-RS	Java API for RESTful Web Services
JAX-WS	Java API for Web Services
JCA	Java Connector Architecture
JDBC	Java Database Connection
JIT	Just In Time Compiler
JMS	Java Message Service
JMX	Java Management Extensions
JSF	JavaServer Faces
JSP	Java Server Page
JSTL	Standard Tag Library for JavaServer
JTA	Java Transaction API
MAF	Management Application Function
MCF	Message Communication Function
MF	Mediation Function
MF-MAF	Mediation Function - Management Application Function
MIME	Multipurpose Internet Mail Extensions
MSIL	Microsoft Intermediate Language
NEF	Network Element Function
NEF-MAF	Network Element Function - Management Application Function
NEL	Network Element Layer
NEML	Network Element Management Layer
NML	Network Management Layer
ODP	Open Distributed Processing
OMG	Object Management Group
OMT	Object Modeling Technique
ORB	Object Request Broker
OS	Operation Systems
OSF	Operation Systems Function
OSF-MAF	Operation System Function - Management Application Function
OSI	Open Systems Interconnection
QAF	Q Adaptor Function
QAF-MAF	Q Adaptor Function - Management Application Function

RDBMS	Relational Database Management System
RIA	Rich Internet Applications
SAAJ	SOAP-with-Attachments API for Java
SAX	Simple API for XML
SF	Security Function
SML	Service Management Layer
SOAP	Simple Object Access Protocol.
SPI	Service Provider Interface
SQL	Structured Query Language
TMN	Telecommunications Management Network
WCF	Windows Communication Foundation
WF	Workflow Foundation
WFP	Windows Presentation Foundation
WS Metadata	Web Services Metadata
WSF	WorkStation Function
WSSF	WorkStation Support Function
XAML	EXtensible Application Markup Language
XML	eXtended Markup Language

Einleitung

1.1 Standards als Wegweiser

Die Architektur von Informationssystemen orientiert sich an etablierten Architekturstandards. Die gekonnte konkrete Umsetzung eines bestimmten Systems beginnt mit der Auswahl des richtigen Standards. Leider gibt es eine Vielzahl solcher Standards. Die hier skizzierte Auswahl der Standards, die noch nicht im Rahmen der Vorlesung behandelt worden sind, gibt eine Übersicht und ist bei weitem nicht vollständig. Sie erlauben jedoch die Illustration des Umfangs und des Wirkungsbereiches von Standards. Neben den wichtigen historischen Standards wie beispielsweise Client Server und CORBA sind die zwei wichtigsten Industriestandards heute – die Java Enterprise Edition und das Microsoft .NET Framework zentrale Grundlagen für den Bau von Systemen.

1.2 Architekturstandards

1.2.1 Client Server

Die Client – Server Architektur ist bereits in den frühen 80er Jahren entwickelt worden. Sie ist der Basisstandard für den ganzen Bereich Distributed Processing.

1.2.2 Common Object Request Broker Architecture

CORBA ist eine Entwicklung der OMG (Object Management Group). Diese Organisation wurde 1989 gegründet und hat heute über 600 Mitglieder (Firmen, Organisationen und Universitäten). Ihre Selbstgestellte Aufgabe ist die Förderung und Entwicklung Objektorientierter Technologie, die "Erstellung von Industrienormen und Spezifikationen zur Unterstützung einer allgemeinen Plattform zur Anwendungsentwicklung". Sie möchte dabei ausdrücklich die Entwicklung heterogener Rechenumgebungen ermöglichen. Das CORBA Referenzmodell und dessen Herzstück, der ORB (Object Request Broker), ist eine der wichtigsten verteilten sprach- und plattform-unabhängigen Architekturen.

1.2.3 Information System Architecture

Der ISA Standard ist ein Beispiel einer so genannten Enterprise Architecture, einer Systemarchitektur, die über die technologische Umsetzung hinaus die Gesamtbetrachtung der betrieblichen Informationssysteme und deren Einfluss auf eine Unternehmung darstellt.

1.2.4 Open Distributed Processing

Eine Möglichkeit zur konzeptionell getrennten Betrachtung einer Informationsarchitektur. ODP wird heute vor allem indirekt im Engineering von Architekturen angewendet. Die Betrachtung verschiedener Views auf ein und dasselbe System ist einer der zentralen Modellierungs-Schritte bei der Definition einer Architektur.

1.2.5 Telecommunications Management Network

TMN ist von ITU (International Telecommunication Union) als Standard zur Realisierung komplexer Telekommunikations-Systeme definiert worden. Dieser Standard ist eine wesentliche Grundlage zur Realisierung von Systemen durch verschiedene Hersteller. Nur aufgrund der Anwendung der standardisierten Schnittstellen sind Basisfunktionen der Telekommunikation, wie beispielsweise das internationale Roaming, überhaupt möglich.

1.2.6 Weitere Standards

Standard	Anwendung
OASIS (Organization for the Advancement of Structured Information Standards)	Document Management Systems und Archivsysteme
DoDAF (Department of Defense Architecture Framework)	Das Referenzmodell des US Departement of Defense für die Organisation der Enterprise Architektur und der Systemarchitektur in verschiedene Views, ähnlich den ODP Viewpoints [DoDAF 2009].
ARIS (Architecture of Integrated Information Systems)	Das ARIS Framework ist eine ganzheitliche Betrachtung des Unternehmens und seiner Informationssysteme und wurde von August-Wilhelm Scheer entwickelt [Scheer, Jost 2003]
CIMOSA (Computer Integrated Manufacturing Open System Architecture)	CIMOSA ist ein Framework zur Modellierung von CIM Unternehmen und basiert auf einem Lifecycle Modell, einer Modellierungssprache sowie einem Architekturmodell [CIMOSA 1993].
GERAM (Generalised Enterprise Reference Architecture and Methodology)	Unternehmensweite Sicht auf eine Architektur basierend auf dem ISO 15704 (2000 - Requirements for enterprise-reference architectures and methodologies) Standard [Nell 1997]
TOGAF (The Open Group Architecture Framework)	TOGAF sieht eine Trennung zwischen fachlichen ABB's – Architecture Building Blocks – und deren konkrete technische Umsetzung durch SBB's – Solution Building Blocks vor. Als Architecture Building Blocks gelten Organisation, Ziele als Treiber, Prozesse, Geschäftsobjekte und fachliche Dienste [Hornford et al. 2011].

Client - Server

2.1 Einleitung

Der Name "Client – Server" wurde in den 80er Jahren des letzten Jahrhunderts für die Anbindungen von Personal Computers (PCs) an ein Netzwerk verwendet. Als Architekturstandard hat sich Client-Server Ende dieser 80er Jahre durchgesetzt, dessen Ziel es war eine vielseitige und modulare Infrastruktur bereitzustellen, die skalierbar, interoperabel, flexibel und einfach zu betreiben war.

Das Client – Server Modell ist wie vor eines der wichtigsten Architektur Standards, die im industriellen Einsatz sind. Es teilt den beteiligten Komponenten zwei Rollen zu:

- **Der Client:** Benützer eines Services
- **Der Server:** Provider eines Services

2.2 Aufbau des Modells

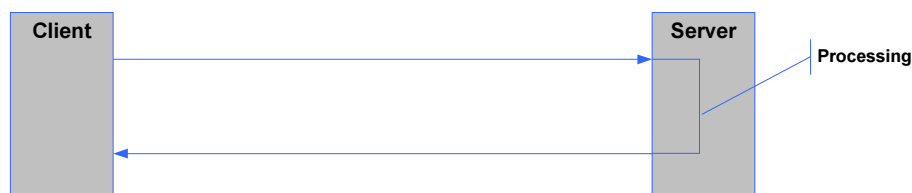


Abbildung 1: Client - Server

Das Modell legt die Rolle der Beteiligten und die zeitliche Abfolge der Interaktionsschritte fest. Ein Client ist ein Prozess mit allgemeinen Aufgaben, während der Server bestimmte Aufgaben, also Dienste die er anbietet, besonders effizient erledigen kann. Ein Server kann immer mehrere Clients bedienen, die Verwaltung der Anfragen und deren Bearbeitung ist seine einzige Aufgabe.

2.3 Definitionen Client-Server

Definition von Neil K. Guy [Guy 1991]

A paradigm used in computing architecture, based on the idea that a client computer is used to access information stored on a server. The World Wide Web is based around this type of client-server architecture. The opposite is peer-to-peer networking.

Das Online GIS-Tutorial (www.gis-tutor.de) führt in seinem Glossar folgende Definition auf:

Client-Server ist i.a. eine kooperative Datenbank, bei der verschiedene Aufgaben unter verbundenen Rechnern aufgeteilt werden: Datenverwaltung, Transaktionsverarbeitung, Netzwerkmanagement, Oberflächengestaltung. Dabei kommuniziert die Hardware über Netzwerke, meist Ethernet. Daten können im einfachsten Fall über "file transfer" ausgetauscht werden. Eleganter, wenngleich software-technisch wesentlich aufwendiger, ist eine Programm-Programm-Kommunikation zwischen Client und Server.

2.4 Einbettung

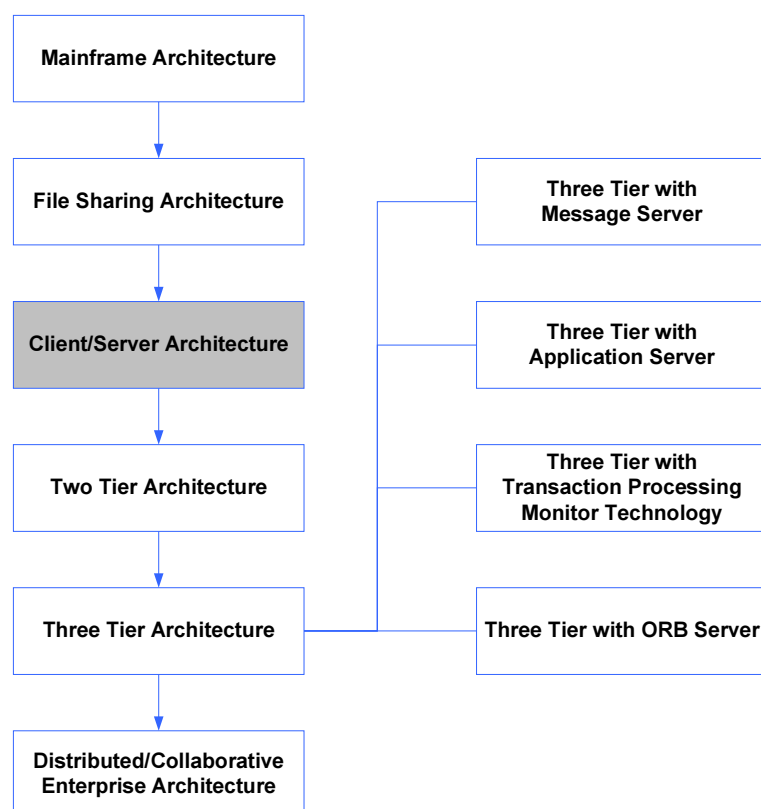


Abbildung 2: Client - Server

Das SEI (Software Engineering Institute) bettet die Client – Server Architektur in eine Hierarchie von strukturierten Architekturmodellen ein [Schussel 1995].

- **Mainframe Architecture:** Die Intelligenz eines Systems befindet sich zentral auf einem Host Computer. Die Interaktion mit dem Host erfolgt über ein so genanntes Terminal, welches die Tastatureingaben entgegennimmt und anschliessend direkt an den Host sendet. Diese Architektur eignet sich weder für GUI's noch für den Zugriff auf verschiedene Datenbanken auf geographisch getrennten Sites.
- **File Sharing Architecture:** Die ursprünglichen PC-Architekturen basieren auf File Sharing. Daten und Programme werden von Server an den PC gesendet, um dann auf dem PC ausgeführt zu werden. Diese Architektur war in den frühen 90er Jahren populär. Idealerweise konnten pro Server etwa 12 PCs bedient werden, da der Netzwerkverkehr für die Ausführung einer Aufgabe erheblich war.
- **Client/Server Architecture:** Die Einführung einer Relationalen Datenbank, die Anfragen direkt beantworten konnte statt die ganzen Daten auf einen PC zu transferieren, ersetzten die File Sharing Systeme. Hinzu kam die Verteilung der Logik zwischen Client und Server.
- **Two Tier Architecture:** Die Aufteilung auf genau einen Server und mehrere Clients. Das Prozess-Management des Servers erlaubt die gleichzeitige Abarbeitung mehrerer Client Requests. In der ursprünglichen Ausprägung enthält der Server lediglich die Datenbank. Diese Architektur eignet sich für 12-100 PCs.

- **Three Tier Architectures:** Diese Architektur erweitert die Two Tier Architecture durch einen Middle Tier, welcher Funktionalitäten wie Beispielsweise das Queuing von Anfragen, das Ausführen von Applikationslogik. So stellt ein Transaction Processing Monitor Mechanismen für den priorisierten und sicheren Zugriff auf Datenbanken bereit.
- **Distributed/Collaborative Enterprise Architecture:** Sämtliche Funktionen eines solchen Systems sind verteilt. So können mehrere Server und Backendsysteme zu Einsatz kommen. SOA ist heute der wichtigste Ansatz für diese Architektur.

2.5 Dreistufiges Strukturmodell

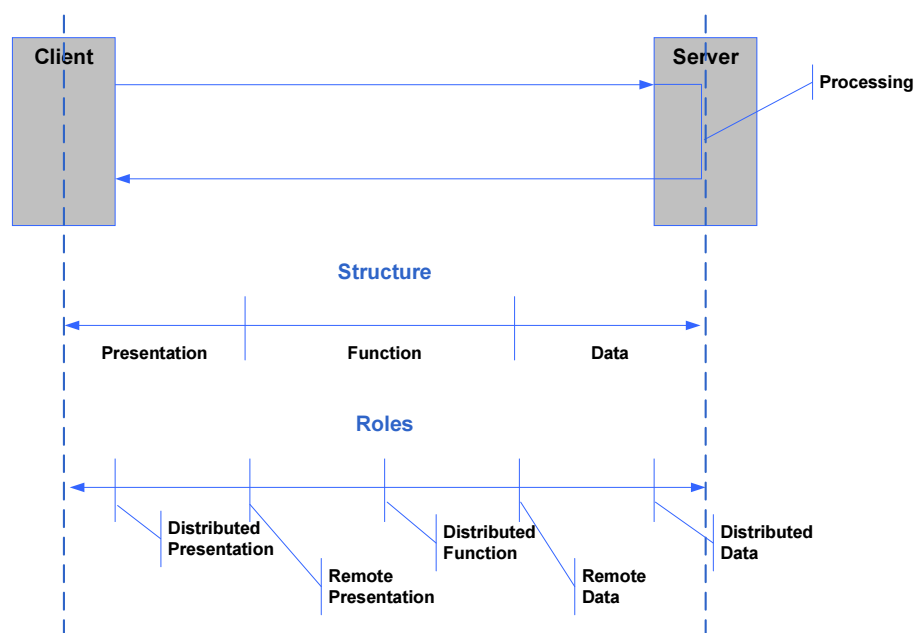


Abbildung 3: Strukturmodell

Client - Server Systeme werden in drei strukturell unterschiedliche Kategorien eingeteilt:

- Präsentation
- Funktion
- Daten

Basierend auf diesen drei Strukturkategorien werden fünf verschiedene Client - Server Rollenschematas klassifiziert:

- **Distributed Presentation:** Die verteilte Präsentation versteckt die Lokalität der einzelnen Clients, während der Server alle Clients zentral (beispielsweise auf einem Bildschirm) darstellt.
- **Remote Presentation:** Die entfernte Präsentation bedeutet, dass ein Client alle Präsentationsaufgaben übernimmt, während Anwendung und Daten vom Server verwaltet werden.
- **Distributed Function:** Die verteilte Funktionalität setzt eine Arbeitsteilung auf funktionaler Ebene zwischen Client und Server um (Cooperative Processing).
- **Remote Data:** Der Server hält alle Daten an einem Ort, es werden keine Daten auf den Clients gespeichert.
- **Distributed Data:** Die Daten werden verteilt auf verschiedenen Servern gehalten, der Client greift gleichzeitig auf verschiedene Server zu.

2.6 Die Rolle des Netzwerkes in einem Client Server Modell

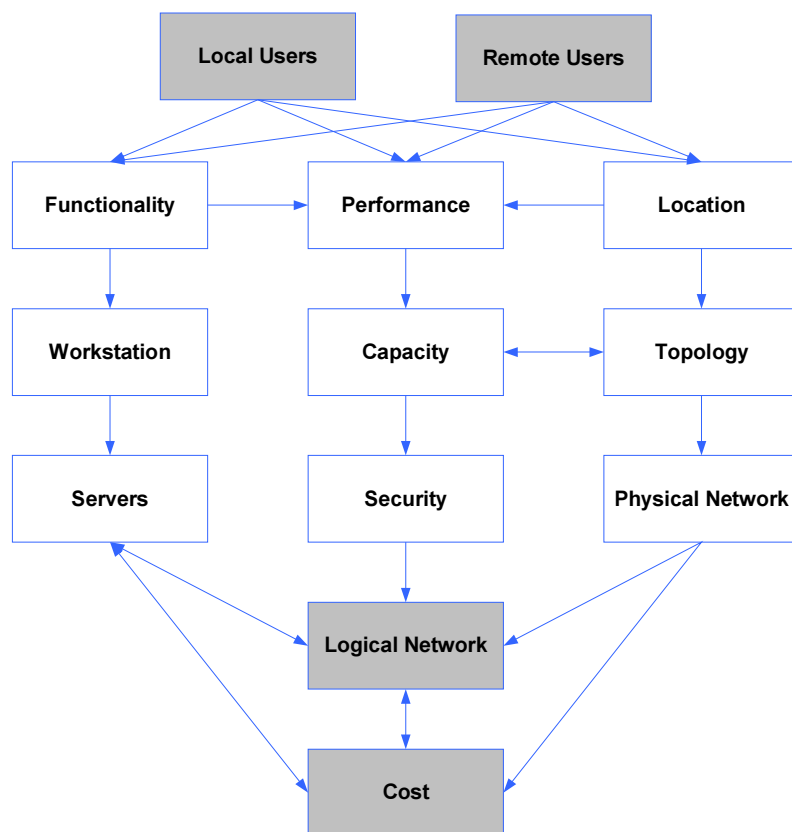


Abbildung 4: Einflussfaktoren der Netzverbindung zwischen Client und Server

In jeder Client/Server Umgebung spielt das Netzwerk eine entscheidende Rolle. Die Dimensionierung eines Systems, vor allem eines kombinierten Intranet/Extranet Systems, spielen folgende Faktoren eine Rolle:

- Die **Lokal Users** im Intranet profitieren von der heute sehr hohen Bandbreite jedes internen Netzwerkes.
- Die **Remote Users** sind je nach **Location** hinsichtlich der Bandbreite äusserst beschränkt.
- Die geforderte **Performance** eines Systems ist abhängig von der **Location** und der **Funktionalität**. Sie wiederum bestimmt die notwendige Kapazität eines Netzes.
- Die Aufteilung zwischen **Workstation** (PC oder Middleware) und den **Servern** (Datenbank oder Host) beeinflusst sowohl die **Kosten** als auch das **logische Netzwerk**.
- Die Topologie ist je nach **Location** (Local oder Remote) verschieden zu wählen (LAN / WAN / Wireless).
- Die **Sicherheit** ist einer der wichtigsten Einflussfaktoren des **logischen Netzwerkes**. Dabei ist sowohl die Segmentierung eines Netzes also auch die Zonierung des Gesamtsystems wichtige mögliche Lösungen.

Java Platform, Enterprise Edition (Java EE)

3.1 Einleitung

Die aktuelle Version 7 der Java Plattform (seit Juni 2013 auf dem Markt), Enterprise Edition (Java EE) ist der Nachfolger von J2EE Java 2 Plattform, Enterprise Edition von SUN – heute Oracle, eines Frameworks für die Entwicklung verteilter Anwendungen. Die Zielsysteme, auf die dieses Framework ausgerichtet ist, sind betriebliche Informationssysteme. Java EE basiert wie Java ME (Micro Edition für mobile Anwendungen) auf der Java Plattform, Standard Edition (Java SE) [Chinnici, Shannon 2009].

Die Programmiersprache Java ist aus einem Versuchsprojekt mit dem Namen „Green Project“ entstanden, welches die drei bei Sun Microsystems angestellten Programmierer Patrick Naughton, James Gosling und Mike Sheridan im Jahr 1990 begonnen hatten. Für die Programmierung einer Haushaltssteuerung wurde die Programmiersprache Object Application Kernel (OAK) entwickelt. Daraus ist ein paar Jahre später Java entstanden. Sun Microsystems brachte im Jahr 1996 die erste Entwicklungsumgebung, das JDK 1.0 heraus. Die aktuelle Version 7 unterstützt HTML 5.

Die Java EE Version 8 wird voraussichtlich erste im ersten Quartal 2017 veröffentlicht werden.

3.2 Aufbau des Modells

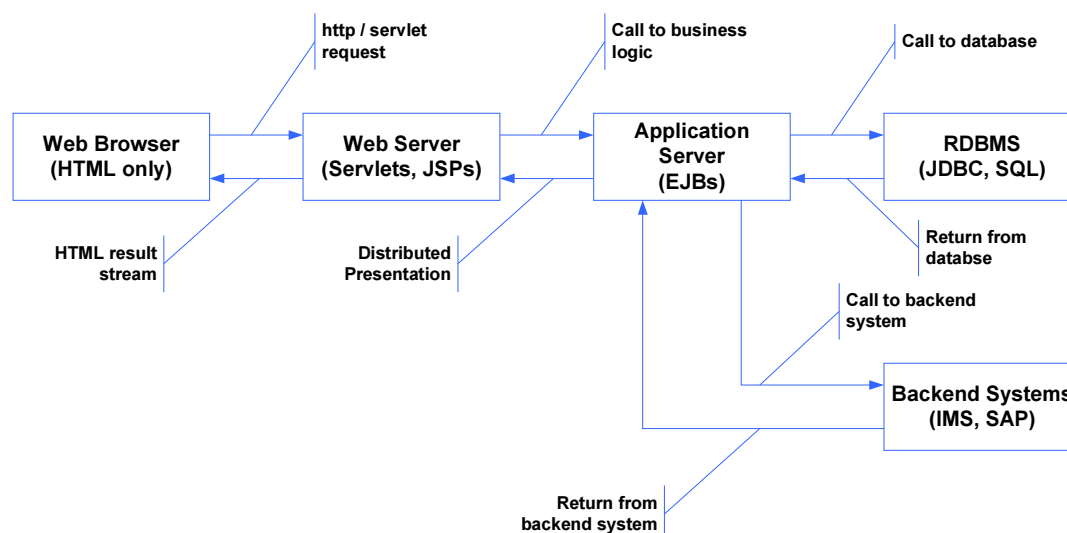


Abbildung 5: Grobablauf einer Java EE Transaktion

Die Grundbausteine von Java EE sind:

- **Applet:** Ein Applet ist eine Software mit oder ohne User Interface, welches in einem Applet-Container abläuft. Der Applet-Container wird in den meisten Fällen in einem Web-Browser ausgeführt.
- **Application:** Eine J2EE Application ist ein selbständiges Java-Programm, mit oder ohne User Interface.
- **Servlet** (sowie JSP Page oder auch Web Event Listener): Ein Servlet ist ein serverseitig ausgeführtes Programm, welches in den meisten Fällen über HTTP aufgerufen wird. Das Servlet generiert entweder ein HTML-Basiertes User Interface oder eine XML-basierte Web Service Funktionalität.
- **Enterprise JavaBean (EJB):** Ein EJB ist eine serverseitig ausgeführte Klasse mit Transaktionsunterstützung. Ein EJB erbringt einen Dienst für die Gesamtanwendung.

3.3 Architektur von Java EE 7

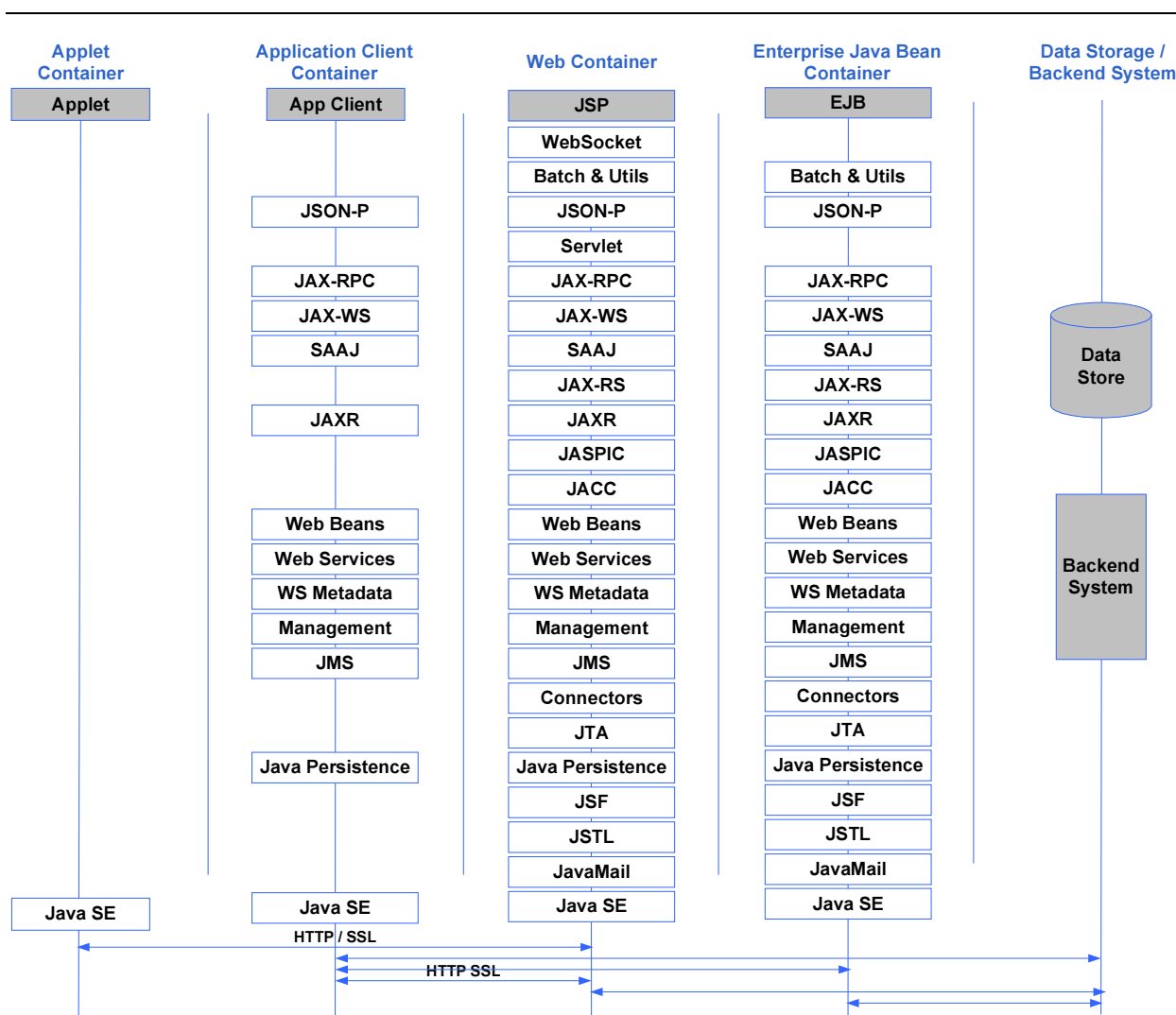


Abbildung 6: Java EE 7 Framework

3.4 Die Dienste von Java EE 7

- **J2SE:** Die Java 2 Standard Edition. Sie beschreibt den definierten Standardumfang der Programmiersprache Java samt seinen Bibliotheken. Im Weiteren werden neben der Programmiersprache, die virtuelle Maschine, der Applet-Container und sämtliche Bibliotheken spezifiziert. Die Implementierungen werden von verschiedenen Herstellern zur Verfügung gestellt.

- **JavaMail:** Das Java-Interface zum Versenden von Email.
- **JSTL:** Die Standard Tag Library für JavaServer Pages ist eine Sammlung verschiedenster Tags für JSP (Java ServerPages) Seiten.
- **JSF:** JavaServer Faces ist eine Technologie zur Realisierung von GUI's.
- **Java Persistence:** Das Java EE Service Provider Interface für die Hersteller von Frameworks für die Persistenz.
- **JTA:** Das Java Transaction API ist für die Steuerung von Transaktionen (Transaction Demarcation) zuständig.
- **Connectors:** Die Java EE Connector Architecture (JCA) ist ein Service Provider Interface, welches es erlaubt verschiedene Adaptoren für den Zugriff auf Backend Systeme zu realisieren. JCA definiert eine standardisierte Anzahl so genannter System-Level Contracts zur Verwaltung von Verbindungen, Transaktionen, Sicherheitsmechanismen, Thread Management, Meldungsanlieferung und der Transaktionsweiterleitung.
- **JMS:** Der Java Message Service ist ein Mechanismus zum asynchronen Aufruf von Services.
- **Management:** Die Java EE Management Spezifikation definiert eine Reihe von API's für die Verwaltung von Java EE Server durch ein spezielles EJB sowie durch JMX (Java Management Extensions).
- **WS Metadata:** Web Services Metadata ist ein Mechanismus zur Vereinfachung der Entwicklung von Web Services.
- **Web Services:** Fasst sämtliche Mechanismen für den Nutzer und den Anbieter von Web Services zusammen.
- **Web Beans:** Web Beans ist eine neue Technologie, die inzwischen in Java Context and Dependency Injection umbenannt worden ist, um Web und Rich Client User Interfaces zu kombinieren.
- **JACC:** Java Authorization Contract for Containers besteht aus einer Menge von Securitymechanismen für die Zugriffskontrolle.
- **JASPIC:** Der Java Authorization Service Provider Contract for Containers ist ein so genanntes Service Provider Interface (SPI) zur Realisierung der Meldungs-Authorisierung.
- **JAXR:** Das Java API for XML Registries organisiert den Zugriff auf XML-Verzeichnisdienste.
- **JAX-RS:** Das API für so genannte RESTful Web Services erlaubt die Umsetzung ressourcen-orientierter Web Service Schnittstellen.
- **SAAJ:** Das SOAP-with-Attachments API for Java ist für die Manipulation von SOAP-Nachrichten zuständig.
- **JAX-WS:** Das Java API for Web Services ist die Umsetzung von JAX-RPC für Web-Service-Aufrufe via SOAP/HTTP
- **JAX-RPC:** Das Java API for XML-Based Remote Procedure Calls.
- **JSON-P:** Java API for JSON Processing für die Verarbeitung von JSON Daten
- **WebSocket:** Kommunikationsprotokoll (full-duplex)
- **Batch & Utils:** Batch Applications und Concurrency Utilities for Java EE sind zwei API's für die Batchverarbeitung und zur Unterstützung von asynchronen Diensten

3.5 Zentrale Konzepte

Die Grundidee hinter Java EE und den Instrumenten zur Entwicklung und zur Ausführung von Anwendungen ist, die plattformunabhängige Realisierung von mehrschichtigen Geschäftsanwendungen. Die Plattformunabhängigkeit wird durch den interpretierbaren Bytecode erreicht. Aus Architektursicht stellt Java EE alle modernen Mechanismen für die Umsetzung verteilter Systeme, die im Kontext einer heterogenen Systemlandschaft ablaufen, zur Verfügung.

Anwendungen werden zu Modulen zusammengefasst:

- **EJB Module:** Sie enthalten Enterprise JavaBeans und die zugehörigen Klassen
- **Web Module:** Sie bestehen aus den Komponenten des Web Layers und den Ressourcen, wie beispielsweise den Webseiten.
- **Application Client Module:** Die Klassen des Rich Clients.

- **Adapter Module:** Sie enthalten die so genannten Resource Adaptern.

In der Praxis haben sich allerdings die EJB Module nicht gut durchgesetzt, da sie ohne speziellen Container nicht lauffähig sind, was ein vorgängiges so genanntes Deployment notwendig macht und damit die Entwicklung verzögert. Die Alternative sind so genannte POJO's (Plain Old Java Objects), die ohne den EJB Overhead auskommen. Auf die Gesamtarchitektur einer Java EE Anwendung hat dies keinen Einfluss.

Die neuen Konzepte von Java EE sind Annotations, Profiles und die Etablierung von SPI (Service Provider Interfaces) neben den gängigen API's

Konzept	Erklärung	Bemerkung
Annotations	Annotations dienen dazu, Informationen über die Konfiguration von Anwendungen direkt in den Java Code einzubetten. Sie sind die Alternative zu den getrennten Deployment Deskriptoren.	Mittels Annotations können Mechanismen wie beispielsweise die aspekt-orientierte Programmierung realisiert werden.
Profiles	Profile dienen zur Konfiguration einer Java EE Plattform für einen bestimmten Anwendungszweck	Ein Beispiel ist das Java EE Web Profile, welches den Einsatz bestimmter Komponenten des Frameworks spezifiziert
Service Provider Interface	Eine Schnittstelle, die einem Anbieter die Realisierung eines bestimmten Java API's erlaubt.	Für die Hersteller von Persistenzframeworks und Adaptern zu Backendsystemen

3.6 Java EE Application Servers

Produkt Name	Hersteller	Web Site
Oracle WebLogic	Oracle	www.oracle.com
WildFly Application Server	JBoss Group	wildfly.org
IBM WebSphere Application Server	IBM	www.ibm.com
JBoss Application Server	JBoss Group	www.jboss.org
GlassFish	Oracle	glassfish.java.net
Jetty	OSS (Eclipse)	eclipse.org/jetty/
SAP NetWeaver Application Server	SAP	www.sap.com

.NET Framework

4.1 Einleitung

Definition .NET Framework von Microsoft:

NET Framework ist ein Produkt, das die Entwicklungsgrundlage für die Microsoft .NET-Plattform bildet. .NET Framework und das Geräteorientierte .NET Compact Framework stellen eine verwaltete, sichere Ausführungsumgebung für XML-Webdienste und Anwendungen sowie umfassende Unterstützung für XML zur Verfügung. Die Schlüsseltechnologien in .NET Framework sind Common Language Runtime, die Klassenbibliotheken und ASP .NET.

- Das .NET Framework ist eine Umgebung zum Erstellen, Bereitstellen und Ausführen von XML Webdiensten und anderen Anwendungen.
- Das .NET Framework besteht aus zwei Teilen: Common Language Runtime und den Klassenbibliotheken, die Microsoft ASP .NET, Unternehmensdienste, Microsoft ADO .NET und andere umfassen.
- Das .NET Framework ist das Kernstück der .NET-Entwicklerangebote von Microsoft; Microsoft Team Systems – die zentrale und integrierte Entwicklungsumgebung ist darauf ausgerichtet.

4.2 Framework

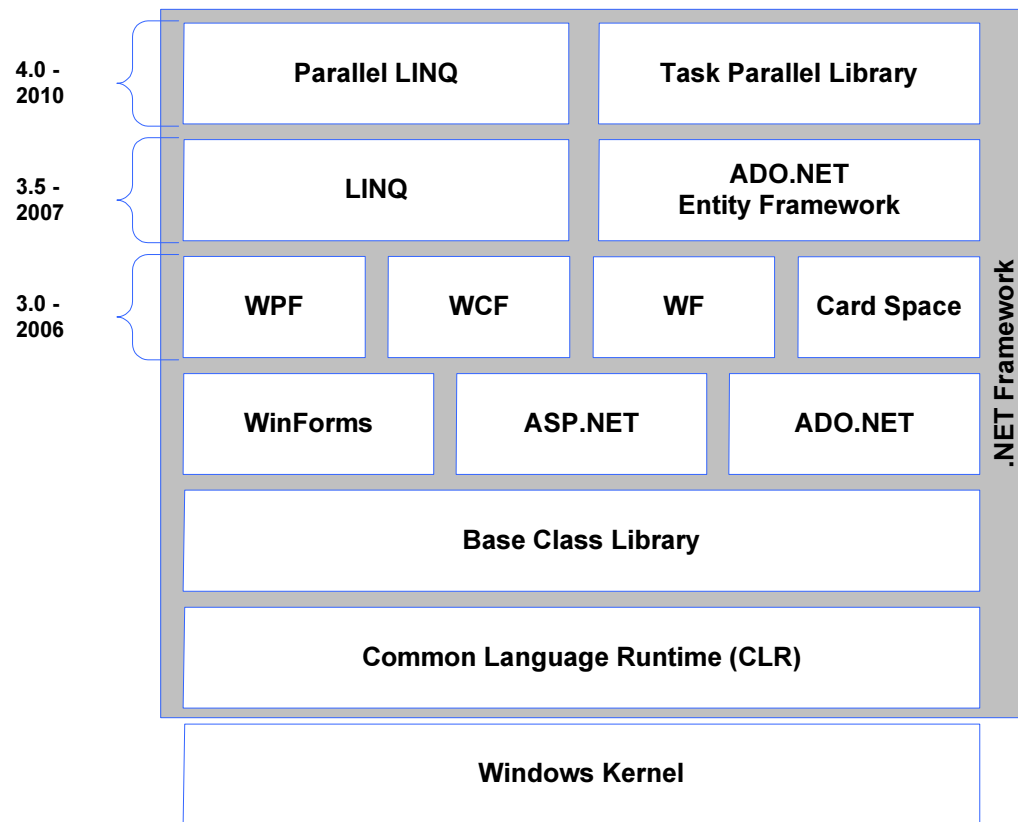


Abbildung 7: .NET Framework

Das .NET Framework bildet die Architekturbasis für die Microsoft .NET Plattform [Schwichtenberg 2008].

- **Common Language Runtime:** "Virtuelle Maschine" von .NET.
- **Base Class Library:** Grundfunktionalität bestehend aus verschiedenen Komponenten (Systemfunktionen, GUI, Web).
- **Data & XML:** Erweiterte Funktionalität für Daten- und XML-Verwaltung.
- **ASP:** Active Server Pages die für das Generieren von HTML Pages verwendet werden.
- **Web Services:** Standard Web Service Interface.
- **User Interface:** Mechanismen zur Erstellung von User Interfaces für Web- oder Standalone-Applikationen.

4.3 Common Language Runtime (CLR)

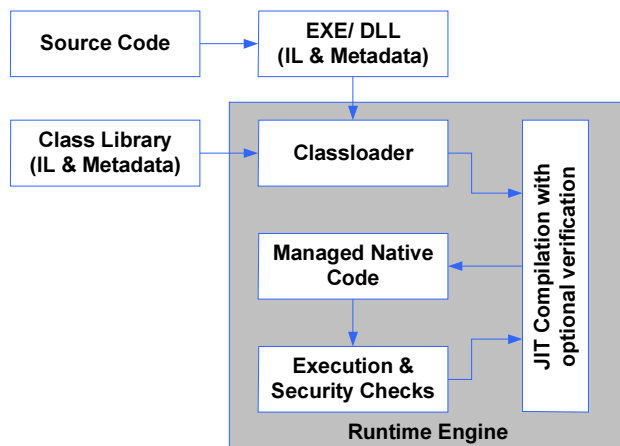


Abbildung 8: Common Language Runtime [Box 2002]

Die Common Language Runtime bildet die Umgebung, in welcher eine .NET Anwendung läuft. Sie verhält sich gegenüber der .NET Anwendung gleich, auch wenn sie auf unterschiedlichen (Windows) Betriebssystemen eingesetzt wird. Dadurch lässt sich eine Anwendung "unabhängig" vom Betriebssystem ausführen.

Ablauf:

- Schreiben des Quellcodes in einer .NET Sprache (C#, vb.NET, c++.NET, cobol.NET,...)
- Der Quellcode wird in MSIL übersetzt und als EXE / DLL abgespeichert.
- Die entstandene EXE oder DLL wird durch den Classloader in die Runtime geladen.
- Der Just In Time Compiler übersetzt die Code Bereiche, welche zur Ausführung der Anwendung notwendig sind, in Maschinencode.
- Zusätzlich benötigte Basisklassen werden vom Just In Timer eingelesen und in "Maschinencode" übersetzt.
- Der erzeugte Maschinencode wird innerhalb der Runtime ausgeführt.

4.4 Base Class Library

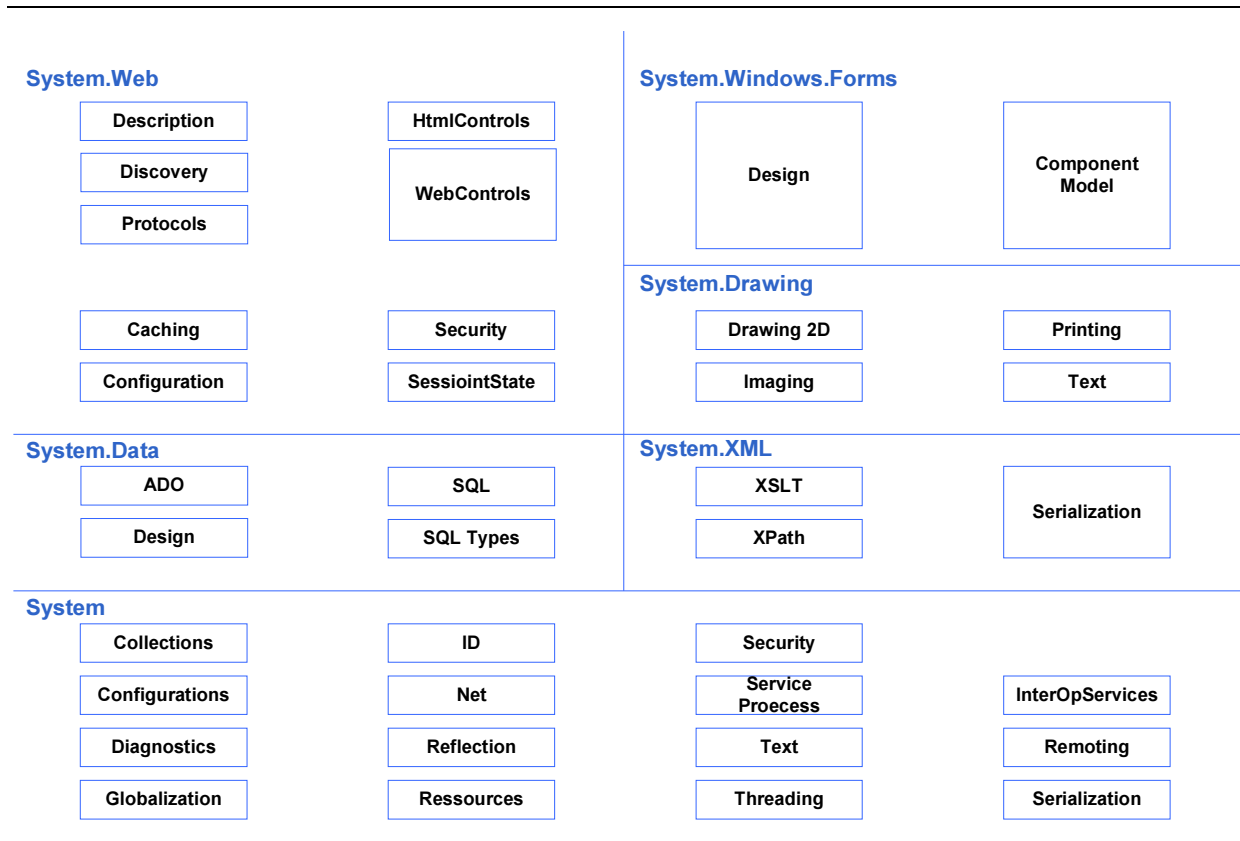


Abbildung 9: Base Class Library

- **System.Web** enthält die Basisfunktionalität für den Zugriff auf Web-Komponenten. Wichtige Bestandteile sind WebServices und WebForms.
- **System.Windows.Forms** stellt die Komponenten für die Erstellung von Windows Forms bereit.
- **System.Drawings** integriert die Basis für grafische Darstellung und Druckfunktionalität.
- **System.Data** ist für den Zugriff auf Daten verantwortlich und enthält die Komponenten für ADO.NET
- **System.XML** ist für die Verarbeitung von XML Daten verantwortlich.
- **System** ist der Basis-Namespace. Er beinhaltet alle Basisklassen, von welchen andere Klassen abgeleitet werden. Wichtige Funktionen sind IO, Net, Collections und Serialization.

4.5 Microsoft .NET Servers

Name	Funktion
.NET Web Server	Standard Web Server
.NET Standard Server	Netzwerkserver für kleinere Unternehmen
.NET Enterprise Server	Server für mittlere Unternehmen
.NET Datacenter Server	Server mit Clusterfunktionen für grosse Unternehmen

4.6 Version 4.5 – Heute .NET 2015 / .NET Framework 4.6

Die Version 4.5 von Microsoft .Net ist vor 3 Jahren erschienen (aktuell ist 4.6). Es enthält eine Reihe von Verbesserungen gegenüber der Version 4.0, die den Garbage Collector, die asynchrone Programmierung und LINQ betrifft. Zusätzlich wurde die Windows Workflow Foundation (WF) erweitert und Steuerelemente in WPF integriert sowie das Databinding verbessert.

Komponente	Erklärung
Windows Presentation Foundation	WPF ist ein Programmiermodell für die Trennung der Darstellung von der Anwendungslogik über XAML.
Windows Communication Foundation	Eine Service-Orientierte Kommunikationsplattform, die DCOM, Enterprise Services und Web-Services unter einem einheitlichen API zusammenfasst.
Windows Workflow Foundation	Ein Framework zur Realisierung von Workflow-Lösungen bestehend aus einem Workflow-Modul, einem Designer, einem API sowie einem Laufzeitservice.
Silverlight	Eine plattformunabhängige Browser-Erweiterung als Subset von WPF. XAML wird als Schnittstelle zwischen Designer und Programmierer eingesetzt

4.7 Gartners Vergleich

Die renommierte Marktforschungsfirma Gartner hat vor 10 Jahren folgenden Vergleich veröffentlicht [Wiggins 2003]:

Theme	Java Platform	.Net Platform
Commitment to independent open standards	Rejected in favor of its own benevolent-dictatorship-based system. Grade: F	Halfhearted commitment retaining complete proprietary control over higher-level APIs. Grade: D+
Commitment to industry community process	Well-supported and evolving Java Community Process (JCP) organization. Grade: B+	Immature and tightly controlled shared-source and shared development process. Grade: D+
Commitment to Internet standards	Strong commitment to existing Internet standards but lagging in support for Web services. Grade: A-	Strong support for existing Internet standards and visionary in emerging Web services. Grade A+
Mainstream Industry community support	JCP, despite shortcomings, is widely supported by the vast majority of Java technology vendors. Grade: A-	Highly likely to fall far short of building community support among third-party vendors. Grade: C-
Bottom Line Grade	B	C

Die Firma hat eine Liste von Empfehlungen aufgestellt:

- Small-to-medium-sized enterprises should focus on one platform as the strategic basis for e-business solutions.
- Large enterprises must support both platforms. Consequently, they should focus on integration technologies.
- Enterprises should expect and plan for staffing challenges and skill shortages on both platforms.
- Developers should focus on emerging Web services architectures on either platform.
- SOA is the best practice of modern systems design.
- Demand service-oriented modularity and event driven architectures from your developers and your vendors.

Common Object Request Broker Architecture

5.1 Einleitung

CORBA (Common Object Request Broker Architecture) ist ein Standard für objektorientierte Systeme. CORBA erlaubt ein plattform- und programmiersprachenunabhängiges Zusammenarbeiten von Objekten über Rechnergrenzen hinweg durch den Einsatz eines Vermittlungsdienstes (ORB = Object Request Broker) [OMG 2002].

5.2 Definition

Definition des Document Management Portals in Deutschland:

CORBA ist eine Spezifikation für eine Middleware-Architektur, welche die Kommunikation zwischen Objekten und Programmen ermöglicht. Über CORBA werden Schnittstellen definiert, die eine system- und anwendungsübergreifende Kommunikation ermöglichen. Auf diese Weise werden verschiedene Anwendungen zu einem Komponentenbasierten Gesamtsystem integriert. CORBA ist systemunabhängig und auch nicht an eine bestimmte Programmiersprache gebunden. Unter dem Gesichtspunkt der Integration verschiedener Anwendungen ist CORBA eine alternative Technologie zu Workflow-Produkten, welche die Integrationsanforderungen mit herstellereinspezifischen Schnittstellen (proprietary Interfaces) abdecken.

5.3 CORBA Referenzmodell

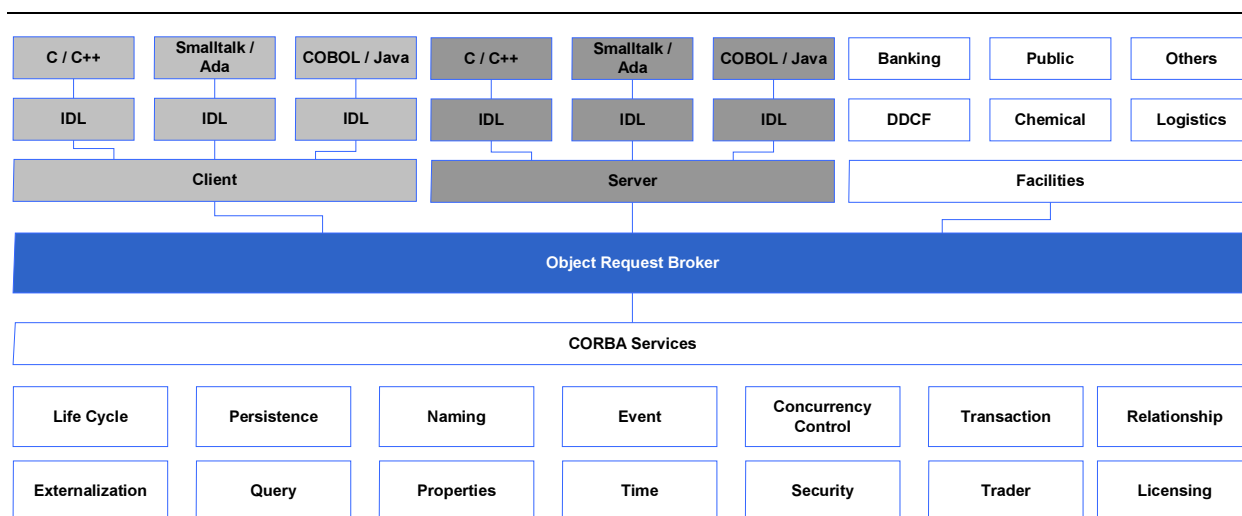


Abbildung 10: CORBA Referenzmodell

Das CORBA Referenzmodell oder auch Object Management Architecture (OMA) ist ein Framework für verteilte Systeme. Hauptziele der Architektur sind Portabilität und Interoperabilität.

5.4 Corba Services

- **Life Cycle Service:** Stellt Methoden zum Erzeugen, Kopieren, Verschieben und Löschen von Objekten zur Verfügung
- **Persistence Service:** Definiert ein Interface um Objekte persistent zu speichern. Das Speichern kann dabei in Objektorientierten Datenbanken, relationalen Datenbanken oder einfachen Dateien geschehen.
- **Naming Service:** Dieser bietet die Möglichkeit, Objekte über ihren Namen in der verteilten Umgebung zu finden.
- **Event Service:** Mit Hilfe dieses Services können Objekt dynamisch ihr Interesse an speziellen Ereignissen in der Umgebung anmelden. Dies bedeutet, dass ein Objekt diesen Dienst benutzen kann, um sich z.B. darüber informieren zu lassen, ob sich der Status eines anderen Objektes verändert hat.
- **Concurrency Control Service:** Durch diesen Dienst werden Methoden zum Sperren von Objekten angeboten.
- **Transaction Service:** Stellt ein Zwei-Phasen-Commit Protokoll zur Verfügung, um Änderung von mehreren verteilten Objekten innerhalb einer Transaktion durchführen zu können.
- **Relationship Service:** Durch diesen Dienst wird ermöglicht, dynamisch Verbindungen zwischen verschiedenen Objekten herzustellen. Auch kann man mit Hilfe dieses Dienstes referentielle Integrität zwischen Objekten implementieren.
- **Externalization Service:** Dieser bietet Standardoperationen an, mit denen der Status eines Objektes in Form eines Datenstroms aus- oder eingelagert werden kann.
- **Query Service:** Bietet Anfrageoperationen für Objekte. Die Anfragen sind dabei ähnlich, wie die Anfragen in SQL.
- **Properties Service:** Stellt Operationen bereit, mit denen man Objekten zusätzliche Eigenschaften zuordnen kann.
- **Time Service:** Wie auch andere Zeitdienste in verteilten Umgebungen, stellt dieser Dienst Methoden bereit, um die Zeit in der verteilten Umgebung zu synchronisieren.
- **Security Service:** Mit den hier zur Verfügung gestellten Diensten, kann man die in der verteilten Umgebung notwendigen Sicherheitsanforderungen implementieren.
- **Trader Service:** Diesen Dienst kann man mit den „Gelben Seiten“ vergleichen. Er ermöglicht es, dass Objekte ihre angebotenen Dienste im Netz bekannt machen. Klienten können den Trader-Service benutzen, um nach Objekten zu suchen, die die benötigten Dienste anbieten.
- **Licensing Service:** Hiermit kann der „Besitzer“ eines Objektes die Benutzung des Objektes im Netz kontrollieren und gegebenenfalls Gebühren für die Benutzung erheben.

5.5 Corba Facilities

CORBA Facilities unterscheiden zwischen horizontalen und vertikalen Facilities:

- **Horizontal Facilities:** Facilities unabhängig von deren Einsatzgebieten. Neben der bereits durch die OMG spezifizierten Facility für „Compound Documents“ sind in diesem Bereich Standardisierung unter anderem für Ton oder Graphiken denkbar.
- **Vertical Facilities:** Sie bieten Dienste an, die nur in einem bestimmten Branche benötigt werden, wie z.B. in der Petrochemie oder bei Versicherungen. In einigen Wirtschaftsbereichen haben sich bereits Konsortien gegründet, die solche Dienste definieren wollen.

5.6 Object Request Broker

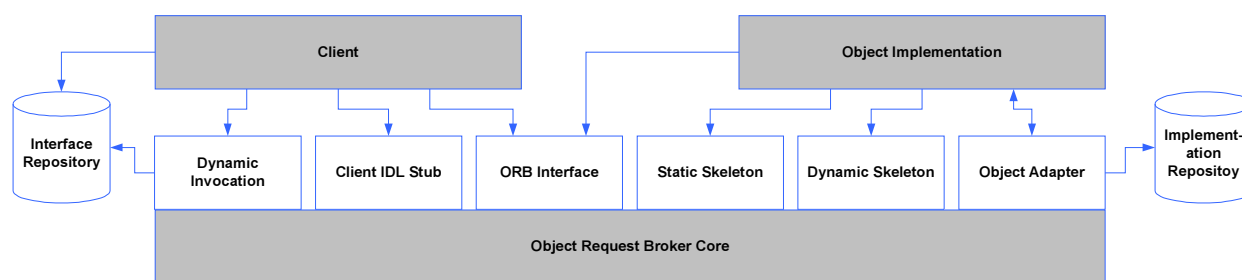


Abbildung 11: Object Request Broker

Der ORB ist das Herzstück einer CORBA Implementation. Er bietet einen transparenten Zugriff auf die Funktionalität eines Servers über eine Objekt-Schnittstelle. CORBA Services und CORBA Facilities werden als server-übergreifende Funktionalitäten realisiert.

- Der **ORB Core** umfasst die Logik des ORB. Das Auffinden von Implementationen und die Vermittlung von Daten sind die wichtigsten Aufgaben des ORB Core.

- Das **ORB Interface** enthält Hilfsfunktionen, die über den ORB systemweit zur Verfügung stehen.

Der Object Adapter ist die Schnittstelle zwischen der Implementation und dem ORB Core.

- Das **Interface Repository** ist eine spezielle Datenbank, die die Signatur aller Operationen jeglicher dem ORB bekannter Schnittstellen enthält.
- Der **Client-Stub** ist eine Abbildung der Schnittstellenbeschreibung in die entsprechende Programmiersprache, welche vom Client eingebunden werden muss. Sie dient dem Aufruf des Objekts in einer der Programmiersprache angepassten Weise.
- Eine Abbildung der Schnittstellenbeschreibung in einem **Skeleton** aus Funktionen der entsprechenden Programmiersprache. Der Implementation-Skeleton muss nur noch mit Programmcode für die Semantik der Operationen gefüllt werden.
- Das **Implementation Repository** ist ein logischer Datenbereich, der alle konkreten Realisierungen der CORBA Objekte enthält. Die Objekte selbst sind verteilt.

Information Systems Architecture

6.1 Einleitung

1987 hat der IBM Forscher John A. Zachmann in seinem Artikel "A Framework for Information Systems Architecture" versucht, eine generalisierte Architekturbetrachtung auf Unternehmensebene zu realisieren [Zachmann 1987]. Seine Überlegungen basierend auf einer Gesamtbetrachtung eines Unternehmens und dessen Informationssysteme, als verteilte Ressourcen. Diese Ressourcen werden, sofern sie nicht zentral verwaltet, geplant, aufgebaut, erweitert und betrieben werden, zu einem unkontrollierbaren Chaos. Er nannte sein Konzept Information System Architecture. Die Tatsache, dass diese System mehr und mehr zur Basis jeder Unternehmenstätigkeit werden, machen für Zachmann einen disziplinierten Ansatz zur Verwaltung dieser Systeme notwendig: Die Information Systems Architecture.

6.2 Das ISA Framework

Model Layer	Data (what)	Function (how)	Network (where)	People	Time	Motivation
Scope (Contextual)	List of Things	List of Processes	List of Locations	List of Organisations	List of Cycles	List of Goals / Strategies
Enterprise Model (Conceptual)	Conceptual Enterprise Model	Business Process Model	Business Logistics System	Work Flow Model	Master Schedule	Business Plan
System Model (Logical)	Logical Data Model	Application Architecture	Distributed System Architecture	Human Interface Architecture	Processing Structure	Business Rule Model
Technology Model (Physical)	Physical Data Model	System Design	System Architecture	Presentation Architecture	Control Structure	Rule Design
System (Logical)	Logical Data Model	Application Architecture	Distributed System Architecture	Human Interface Architecture	Processing Structure	Business Rule Model
Functioning System	Data	Function	Network	Organisation	Schedule	Strategy

Das ISA Framework ist die Grundlage einer Reihe von Standard-Architekturen für bestimmte Branchen. Zu diesem Zweck wurden die einzelnen Komponenten der Frameworks ausformuliert und den spezifischen Bedürfnissen der Branche angepasst:

- **TAFIM** (Technical Architecture Framework for Information Management): Erweiterung für den Verteidigungsbereich. Ein wichtiger Bestandteil von TAFIM ist DGSA (DoD Goal Security Architecture).
- **FEAF** (Federal Enterprise Architecture Framework): Erweiterung für den Public Services Bereich.
- **Visible Universal Model**: Ein Versuch, für alltäglich verwendete Objekte wie etwa ein Check, ein Projekt, eine Qualifikation etc., das entsprechende universell gültige Objektmodell bereitzustellen.

6.3 Perspektiven und Fokus

Das Modell von Zachmann arbeitet mit zwei Betrachtungsweisen auf den Gegenstand des Gesamtsystems. Die Perspektive beschreibt die Sichtweise aus dem Blickwinkel der am Systembau beteiligten Personengruppen. Der Fokus versucht verschiedene Aspekte eines Systems isoliert zu betrachten [Fliss 2005].

6.3.1 Perspektive

Perspektive	Beschreibung
Scope / Planner's View	Beschrieben wird die Businessstrategie. Die Sicht dient der Unterteilung und dem Management der anderen Sichten.
Business Model / Owner's View	Beschreibung des Systems aus Sicht des Unternehmens, in das es eingebracht werden soll.
System Model / Designer's View	Bildet die Sicht des Designers. Ein grobes Modell, frei von Details, wird erstellt.
Technology Model / Builder's View	Die Sicht des Builders beschreibt den Feinentwurf und die spezielle Nutzung von Technologien.
Detailed Representations / Detailed View	Diese Perspektive betrachtet das System aus dem Blickwinkel des Programmierers.
Functioning Enterprise / Operational View	Eine Betrachtung des realisierten Systems im Unternehmensumfeld erfolgt.

6.3.2 Fokus

Fokus	Beschreibung
Data / What	Betrachtet werden Daten, Businessobjekte und Prozesse.
Function / How	Es erfolgt die Betrachtung der Handhabung und der Funktionalität des Systems.
Network / Where	Ziel ist die Darstellung der Systemverteilung und die Präsentation der bestehenden Abhängigkeiten der Elemente.
People / Who	Die Interaktion mit dem System durch Personen oder Organisationseinheiten wird untersucht.
Time / When	Stellt den zeitlichen Ablauf für die Prozesse und Workflows des How-Fokus her.
Motivation / Why	Motivationen für die Systemerstellung wie Zielsetzungen und Businesspläne werden beschrieben.

6.4 ZIFA

Das Zachmann Institute for Framework Advancement (<http://www.zifa.com/>) versucht das Zachmann Framework mittels Seminaren, Kursen und Consulting auszubereiten. Die Web Site bietet Informationen zu Zachmann selbst sowie eine Reihe von Publikationen zur Vertiefung des Themas.

Open Distributed Processing

7.1 Five Viewpoints

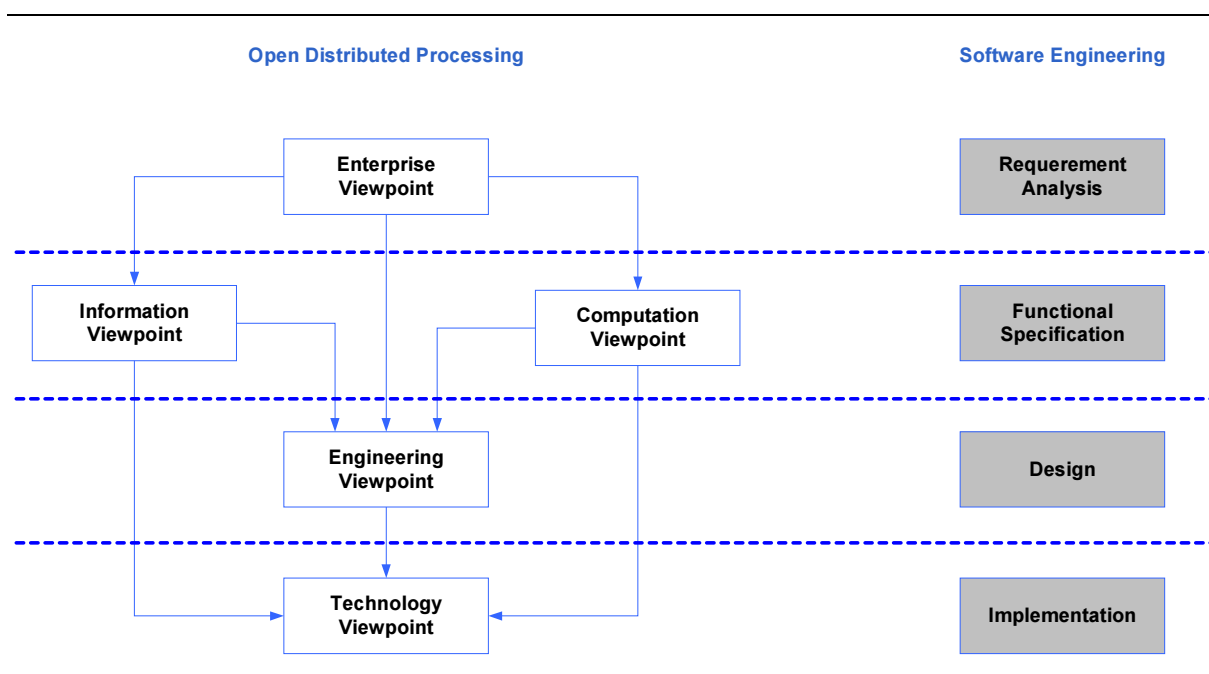


Abbildung 12: ODP und Software Engineering

Open Distributed Processing [ODP 2006] definiert folgende fünf Viewpoints:

- **Enterprise Viewpoint:** Hier wird die Gesamtumgebung für das System sowie sein Zweck beschrieben. Ausserdem werden die Anforderungen (Requirements) an das System, zu erfüllende Bedingungen (Constraints), ausführbare Aktionen (Actions) und DV-Zielvorgaben (Policies) aus Unternehmenssicht definiert.
- **Information Viewpoint:** Dieser Viewpoint legt die Struktur und Semantik der Informationen des Systems fest. Weitere Punkte sind die Definition von Informations-Source und Informations-Targets von sowie die Verarbeitung und Transformation von Information durch das System. Hierzu gibt es Integritätsregeln und Invarianten. Beispiele für Information Languages sind die Object Modeling Technique (OMT).
- **Computational Viewpoint:** Hier wird ein System in logische, funktionale Komponenten zerlegt, die für die Verteilung geeignet sind. Das Ergebnis sind Objekte, die Schnittstellen besitzen, an denen sie Dienste anbieten bzw. nutzen. Die meisten Objektorientierten Sprachen, wie z.B. C++, Java, Smalltalk und auch IDL eignen sich als Computational Language.
- **Engineering Viewpoint:** Dieser Viewpoint beschreibt die erforderliche Systemunterstützung, um eine Verteilung der Objekte aus dem Computational Viewpoint zu erlauben. Hierzu werden generische Infrastrukturobjekte einge-

führt, die die oben genannten Verteilungstransparenzen realisieren, um eine Kommunikation der verteilten Objekte zu ermöglichen. So entsteht das Modell einer generischen, Objektorientierten, verteilten Plattform.

- **Technology Viewpoint:** Dieser Punkt beschreibt die Wahl konkreter Technologien zur Implementierung und Realisierung des Systems. Hierin enthalten ist sowohl die Spezifikation der Hardware (Hersteller und Modell der Rechner, Netzkomponenten, etc.) als auch der Software (Betriebssystem, Kommunikationsprotokolle und Programmiersprachen). Da die Implementierung eines ODP-Systems auf unterschiedlichen technischen Plattformen möglich sein soll, sollten zwischen diesem und den anderen Viewpoints keine Abhängigkeiten bestehen.

7.2 ODP Vorgehen

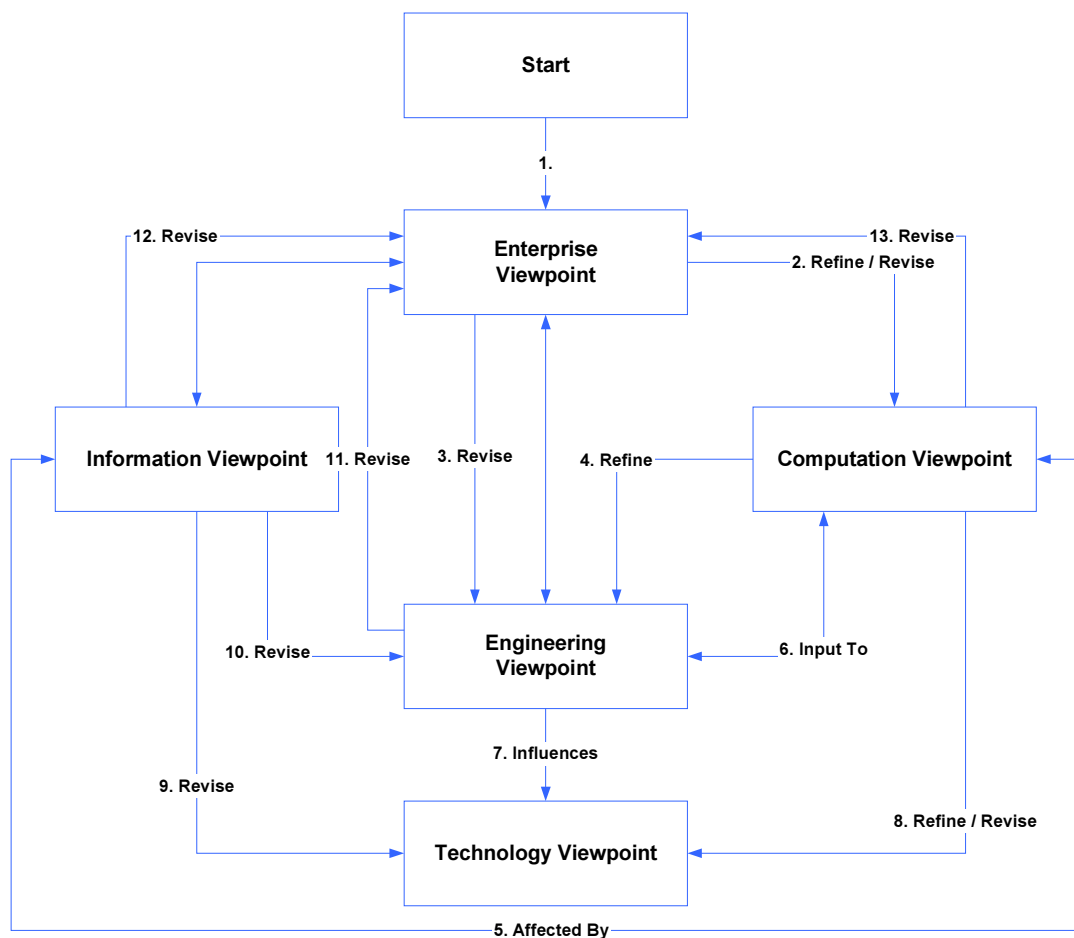


Abbildung 13: ODP Vorgehen

Das ODP Vorgehen eignet sich vor allem für die Neuentwicklung von sehr grossen Systemen. Sehr viele Abbildungen, die durch ODP vorgesehen worden sind, werden heute als UML-Diagramme realisiert.

TMN: Telecommunications Management Network

8.1 TMN Definitionen

TMN ist von ITU (International Telecommunication Union) als Architektur Standard definiert worden [Sellin 1995]. Die Definition lautet:

Telecommunication Management Network ist die geschichtete Architektur für das Management von Telekommunikationsnetzwerken.

8.1.1 Synonyme für TMN

Verschiedene Hersteller und Standardisierungsgremien benutzen verschiedene Bezeichnungen und Begriffe für TMN.

Gremium	Begriff	Informationen
ITU	Telecommunications Management Network	http://www.itu.int
IEEE	Network Management	http://www.ieee.org
OSI	Systems Management	http://www.iso.org

8.1.2 Funktionale Bereiche

TMN Funktionen werden in folgende „Systems Management Functional Areas“ aufgeteilt:

- **Fault Management:** Das Erkennen, die Isolation und die Korrektur von Fehlern in einem Netzwerk.
- **Performance Management:** Die Analyse von Performance Daten, um die Effektivität der eingesetzten Ressourcen zu optimieren.
- **Accounting Management:** Die Zuordnung der Nutzung und Verrechnung der Ressourcen.
- **Security Management:** Umfasst Dienste wie zum Beispiel Encryption, Key Management und Zugriffskontrolle.

8.1.3 Managed Object

Der ITU Standard M.3100 definiert eine Managed Object, dessen Attribute, Schnittstellen und Verhalten:

Ein Managed Objekt ist eine Ressource die gemanaged werden kann.

Ein Managed Object hat immer 4 Charakteristika:

- **Attribute:** Die Eigenschaften eines Managed Objects.
- **Operations:** Die Aktivitäten eines Managed Objects (Schnittstelle).
- **Notification:** Die definierten Nachrichten eines Managed Objects (Events).

- **Behaviour:** Das Verhalten eines Managed Objects.

8.2 Funktionale Architektur

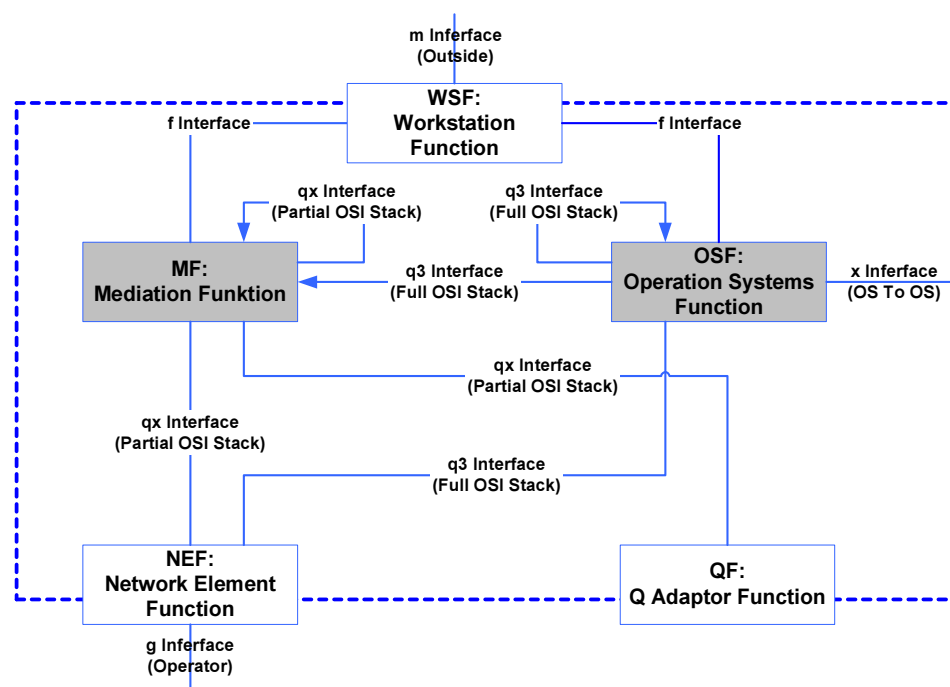


Abbildung 14: Funktionale Architektur

Die funktionale Architektur von TMN sind die Grundbausteine einer TMN Architektur:

- **Operation Systems Function (OSF):** Management- und Planungsfunktionen für das Telekommunikationsnetzwerk und die TMN Komponenten selbst. Vier Arten von OSF existieren: Element OSF, Network OSF, Service OSF und Business OSF
- **Network Element Function (NEF):** Sie repräsentieren Netzwerkkomponenten.
- **WorkStation Function (WSF):** Die Darstellung der Managementinformationen.
- **Mediation Function (MF):** Gateway für den Austausch von Management Informationen über Interfacegrenzen hinweg.
- **Q Adaptor Function (QAF):** Übersetzung von Management Informationen zwischen TMN Systemen und anderen Systemen.

8.3 TMN Layers

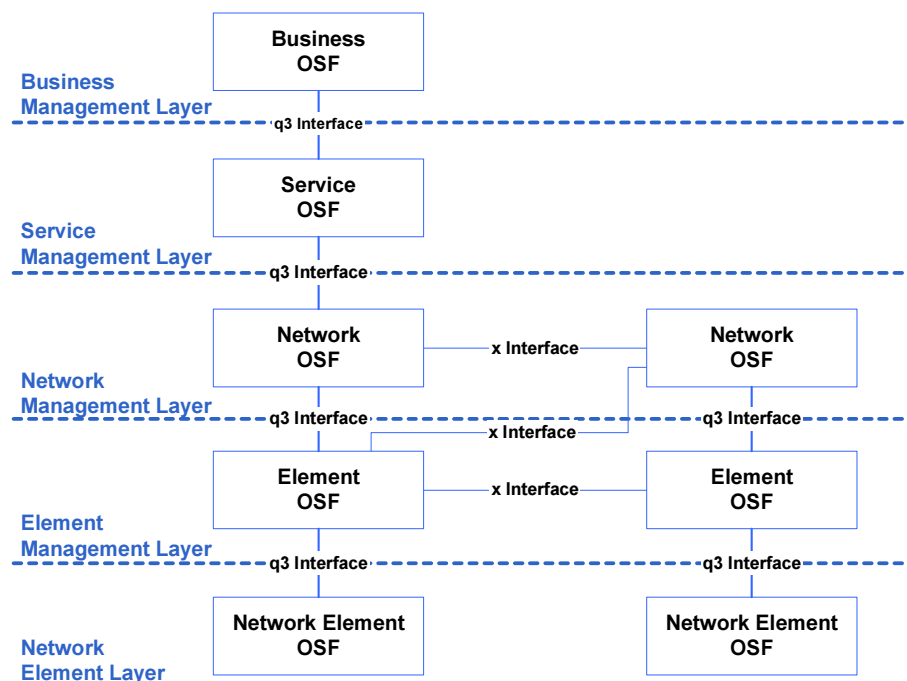


Abbildung 15: TMN Schichtung

- **Business Management Layer (BML):** Instrumente für die Netzwerk Planung
- **Service Management Layer (SML):** die Schnittstelle zum Kunden, die Bereitstellung von Diensten
- **Network Management Layer (NML):** End-To-End Managements des gesamten Netzwerkes
- **Network Element Management Layer (NEML):** Instrumente zur Verwaltung einer Gruppe von Netzwerk Elementen (Netzbereiche, Trunks, etc.)
- **Network Element Layer (NEL):** Instrumente zur Verwaltung eines Netzwerk Elementes

8.4 Innerer Aufbau eines OSF

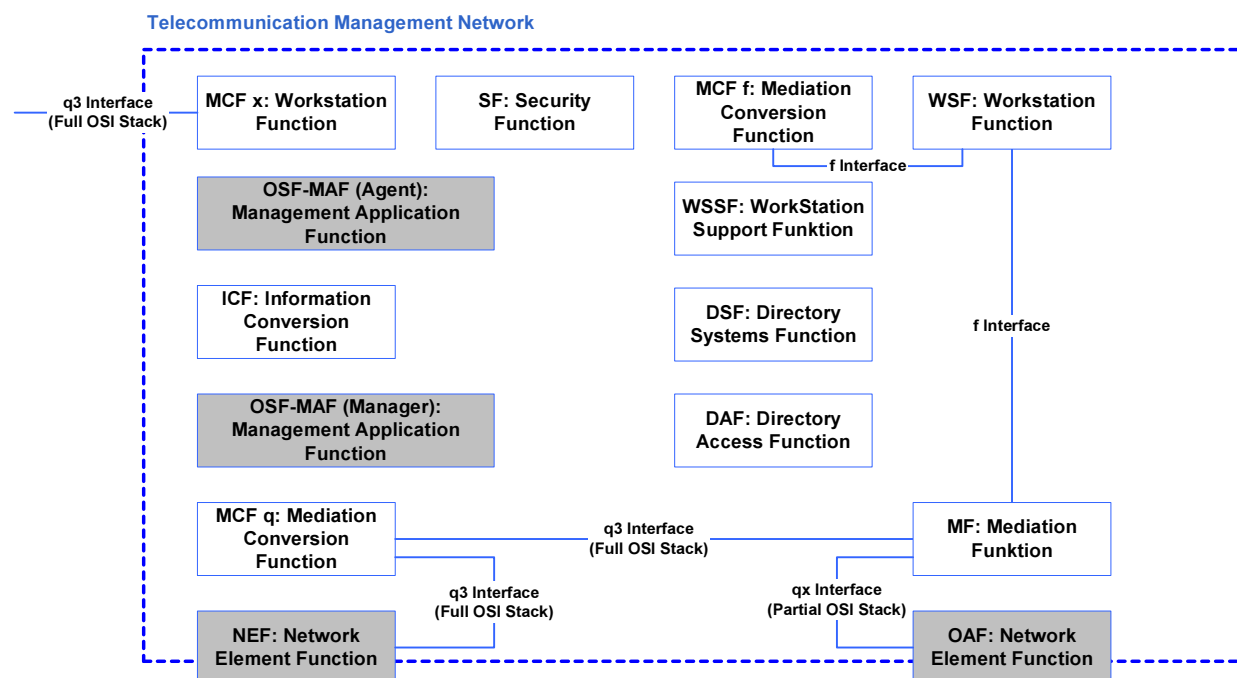


Abbildung 16: Komponenten einer Operation Systems Funktion

- **Management Application Function (MAF):** Funktionalität für einen oder mehrere Management Services und die Verwaltung der Management Informationen
- **Mediation Function - Management Application Function (MF-MAF):** Agent oder Manager einer Management Funktion
- **Operation Systems Function - Management Application Function (OSF-MAF):** Ausführung von Managementfunktionen (z.b. Alarmkorrelation, Trouble Tracking, Performance Analyse)
- **Network Element Function - Management Application Function (NEF-MAF):** Agent eines Network Elements.
- **Q Adaptor Function - Management Application Function (QAF-MAF):** Agent oder Manager eines Q Adaptors (Interface Adaptor)
- **Information Conversion Function (ICF):** Transformation von Informationen von einem Modell in ein anderes
- **WorkStation Support Function (WSSF):** Utilities für die Organisation verschiedener Management Funktionen
- **User Interface Support Function:** GUI eines TMN Systems
- **Message Communication Function (MCF):** Kommunikation zwischen verschiedenen Protokollstacks
- **Data Communication Function (DCF):** Routing und Interworking (OSI Layer 1-3)
- **Directory Systems Function (DSF):** Realisiert um Directory Support. Directories basieren auf X.500 und entsprechen einer MIB in SNMP
- **Directory Access Function (DAS):** Zugriff auf Directories
- **Security Functions (SF):** Sicherheitsfunktionen (Authentifikation, Zugriffskontrolle gemäss X.800)

8.5 Physische Architektur

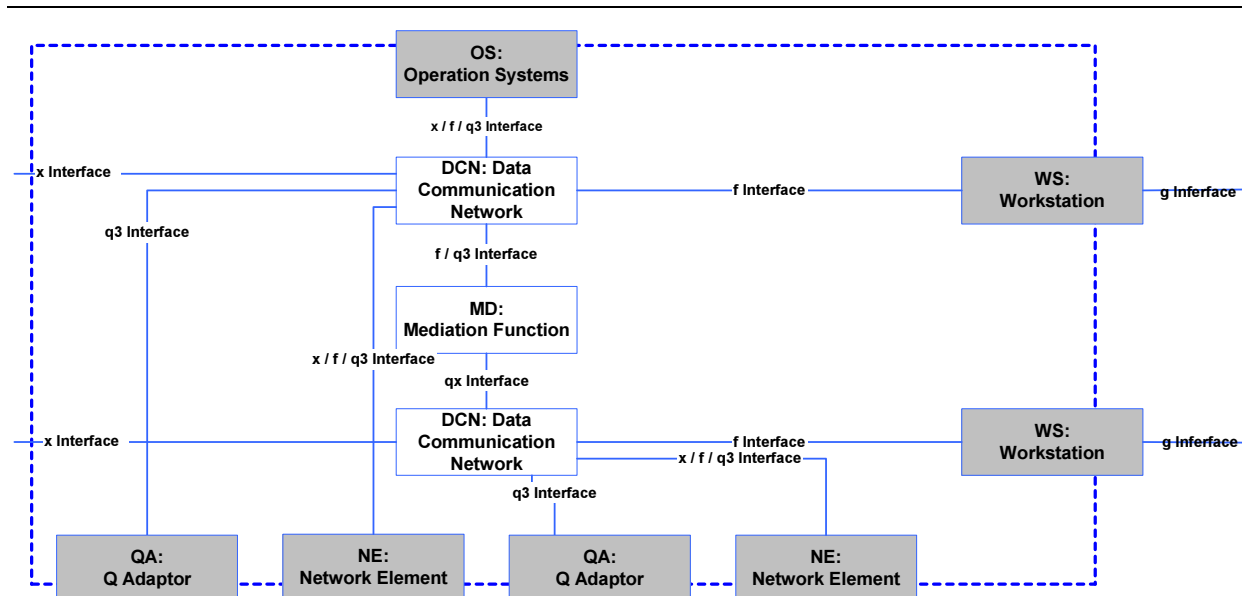


Abbildung 17: Physische Architektur eines TMN Systems

- **Operation Systems (OS)**: Informationsverarbeitung der Funktionalitäten Operations, Administration, Unterhalt (Maintenance) und der Provisionierung von Telekommunikationsnetzwerken. Operation System können als Agenten realisiert werden. Mehrere Operation Systeme können ein Management Network zusammengeschlossen sein.
- **Data Communication Network (DCN)**: Das logische Netzwerk mit seinen Routing- und Transportmöglichkeiten um Daten zwischen den einzelnen Elementen wie OS zu OS, OS zu NE, WS und OS und WS zu NE auszutauschen. Das Data Communication Network unterstützt die funktionelle Komponente DCF und bietet Unterstützung für die ersten drei OSI Schichten. Mediation Device (MD): Entität mit Funktionen wie die Konvertierung von Protokollen, Nachrichten, und Signalen, die Übersetzung von Adressen sowie das Routing.
- **Workstation (WS)**: Arbeitsplatz eines (Network-)Operators.
- **Network Element (NE)**: Ein Element, eine Gruppe von Elementen oder ein Teil des Telekommunikationsequipments welches Teil des Telekommunikationsnetzwerkes ist.
- **Q Adaptor (QA)**: Konvertiert nicht-TMN Daten ins TMN Datenformat und umgekehrt.

Anhang A: Architekturentwicklung mit TOGAF