Web Application Engineering
Prüfungsvorbereitung

helmi77

13. Januar 2017

Inhaltsverzeichnis

1 Grundlagen Web-Application-Engineering

Web Application Engineering ist Teilbereich des Web Service Engineerings. Er behandelt

- Internet- und Web-Services
  Konzepte zur Erleichterung des Informationsaustausches
- die systematische Entwicklung von Web-Anwendungen
  Vorgehensweisen zum Entwurf von Anwendungen aus dem Bereich Internet Services

Eine **Web-Anwendung** ist ein Softwaresystem (auf Basis von W3C Spezifikationen) das webspezifische Ressourcen bereitstellt die über eine Benutzerschnittstelle (*Browser*) abgerufen werden können. Der Softwareaspekt alleine (dh. statische Webseite) stellt keine Web- Anwendung dar, mangels Benutzerschnittstelle. Umgekehrt gilt dasselbe.

Die Entwicklung von Webanwendungen leidet unter einigen Problemen, allen voran, dass sie als einmaliges Ereignis angesehen wird, das in einer spontanen Art und Weise von statten geht. Generell gibt es verschiedene Ursachen für **Qualitätsmängel** in der Webanwendungsentwicklung:

- Dokumentzentrierte Sichtweise
  Man versteht die Entwicklung als redaktionelle Aufgabe (Erstellung von Seiten)
- Vermeintliche Einfachheit
  Durch die Ubiquität von Editoren wie Dreamweaver entsteht schnell der Eindruck Webentwicklung wäre relativ simpel
- Nichtnutzung von Know-How
  Methoden des traditionellen SWENG werden unverändert übernommen, Bereiche wie HCI nicht berücksichtigt

Die **Hauptprobleme** großer Webprojekte sind

1. Verfehlte Zielsetzung
2. Verzögerungen
3. Budgetüberschreitung
4. Mangel an Funktionalität/Qualität

Web-Engineering ist die Anwendung systematischer Ansätze um alle Bereiche der Web-Anwendungsentwicklung (Entwurf, Implementierung, Test, Betrieb, Wartung) kosteneffektiv durchführen zu können. Zudem umfasst es die wissenschaftliche Disziplin, die ebenjene Bereiche erforscht.

Die **Grundprinzipien des Web-Application-Engineering** umfassen

- Klar definierte Ziele und Anforderungen
- Klar definierte Phasen der Entwicklung
- Klar definierte Ausgestaltung der Phasen
- Kontinuierliche Überwachung des Entwicklungsprozesses

Man kann Web-Applikationen in unterschiedliche **Kategorien** einteilen. Komplexe Webanwendungen können auch in mehrere der Kategorien fallen. Ausserdem gilt es zu beachten, dass jüngere Kategorien ältere nicht unbedingt ersetzen müssen.

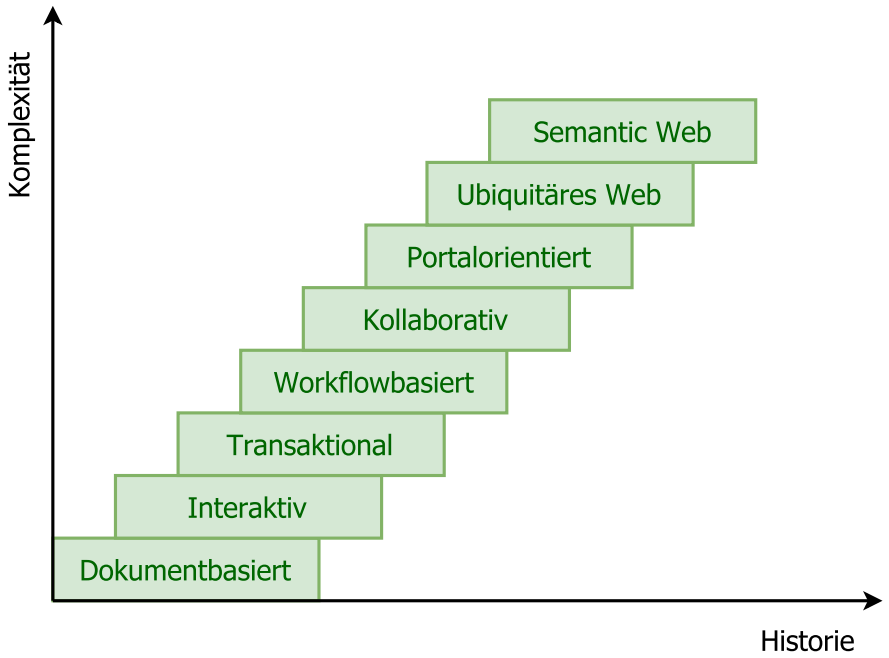

Abbildung 1: Kategorien von Webanwendungen

Interaktive Web-Anwendungen Mittels CGI und HTML-Formularen kann eine einfache Form der Interaktivität ermöglicht werden und somit erste dynamische Inhalte generiert werden, auf Basis von Benutzereingaben. Ursprünglich auf News-Seiten zu finden.

Transaktionale Web-Anwendungen weisen eine hohe Interaktivität auf, so sind oft Modifikationen durch Benutzer möglich. Durch die Einbindung von Datenbanksystemem werden strukturierte Abfragen möglich. Zu den Beispielen zählen Online-Banking-Systeme, E-Commerce und Reservierungssysteme.

Workflow-Basierte Web-Anwendungen erlauben die Abwicklung von Geschäftsprozessen ("workflows") zwischen Benutzern respektive Unternehmen. Voraussetzung ist, dass die zu automatisierenden Prozesse strukturiert sind. Als Beispiel können E-Government Anwendungen angeführt werden.

2 Web-Service-Engineering

Beschäftigt sich mit dem Problem der Umsetzung und des Management von komplexen Webseiten. Es gibt zwar unzählige Systeme für die Implementierung und das Design von Webseiten allerdings kaum Lösungen für den gesamten Lebenszyklus. Ein weiteres Problem ist die mangelnde Zusammenarbeit zwischen Designer und Programmierer, da es kaum überschneidende Aufgabenbereiche gibt.

Zur Realisierung von komplexen Webseiten gibt es mehrere Ansätze.

- Simple HTML-Erstellung
- Seiteneditoren (zB. Dreamweaver)
- Scriptsprachen (ASP, PHP, etc.)
- Objektorientierte Ansätze (JESSICA, Mason, etc.)

Die Ansätze führen alle zum selben Ergebnis: Der Benutzer sieht HTML-Dokumente auf dem Bildschirm.

Teil des Web-Service-Engineering ist auch die Analyse des **Lebenszyklus einer Webseite**.

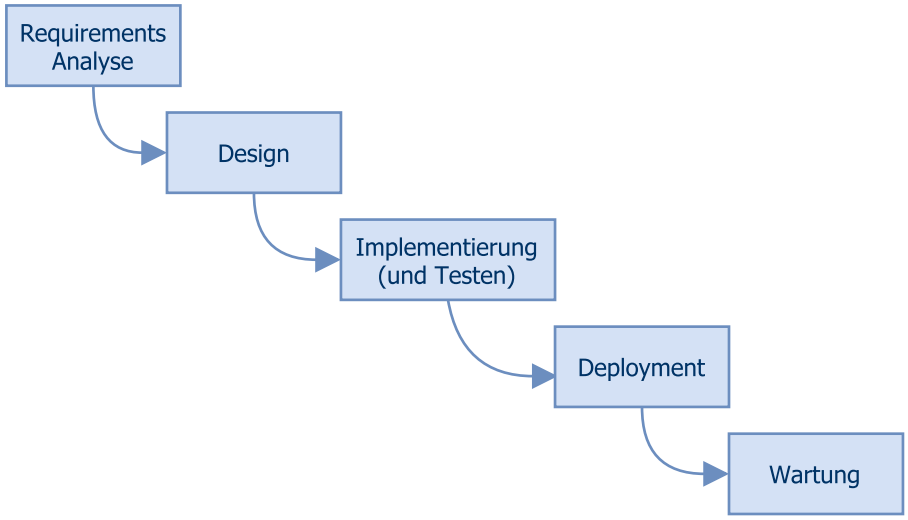

Abbildung 2: Webseite Lebenszyklus

Einer der moderneren Ansätze zur Entwicklung von Webseiten ist das **Extreme Programming** (XP). Die Kernidee dabei ist es viele kleinere Iterationen und mehrere Releases vorzunehmen, anstelle eines durchgehenden langwierigen Entwicklungsprozesses, wie im Wasserfallmodell. Die Iterationen bestehen im Wesentlichen aus den 3 Phasen **Entwurf**, **Implementierung** und **Tests**.

Weitere Herausforderungen komplexer Webanwendungen sind:

- Sicherheit und Zuverlässigkeit
- Performanz
- Management und Wartung

3 Web-Anwendungsentwicklung

Statische Inhalte sind in der Regel auf privaten Homepages und kleineren Informationsseiten zu finden, sie können leicht durch HTML-Editoren wie ehemals Microsofts Frontpage umgesetzt werden.

Dynamische Inhalte hingegen sind solche die zur Zugriffszeit generiert werden. Sie kommen Klassischerweise bei Wetterdiensten oder Börsenkursen zum Einsatz. Als Quelle für die *dynamische* Information dienen Datenbanken oder andere Programme. Im Falle von anderen Programmen ergibt sich oft die Schwierigkeit der Anbindung an den Webserver.

Solche Programme laufen unter dem Titel **Legacy Applications**, dazu zählen auch die erwähnten Datenbanken sowie ERP Systeme (zB. SAP). Die Anbindung erfolgt in der Regel über bereits vorhandene oder erst zu programmierende Schnittstellen.

Das Resultat sind dann Systeme mit einer 3-Tier-Architektur. Aufbauend auf die Datenquelle gibt es die Datenzugriffsschicht (DAL), darauf aufbauend die Businessebene (BL) und an der Spitze die Präsentationsebene oder das UI.

Bei **CGI** handelt es sich um eine Schnittstellendefinition die Kommunikation zwischen Browser, Server und Legacyanwendung zur dynamischen Content-Erzeugung ermöglicht. Während das HTTP-Protokoll keine Zustände unterstützt können mit CGI auf verschiedene Arten States festgehalten werden.

- In Form von URL-Parametern
- In Hidden-Fields
- Als Cookies gespeichert

Neben CGI gibt es andere Alternativen zur dynamischen Content-Erzeugung, wie

- SSI (Server Side Include)
  In HTML-Kommentaren eingebettete Befehle werden on-the-fly ausgeführt
- ASP, PHP
  Ein Interpreter führt in HTML eingebettete Befehle aus
- Servlets, FastCGI
  Ähnlich zu CGI, aber der Interpreter ist Teil des Servers, dadurch wird höhere Performanz erreicht

Das Forschungsprojekt **JESSICA** hat es sich zum Ziel gesetzt ein gleichnamiges Werkzeug zu entwickeln das die Umsetzung und das Design von statischen Informationsseiten sowie die Einbindung dynamischer Inhalte unterstützt.

Erreicht wird dieses Ziel durch Management von Layout und Inhalten, so können Corporate Identities gewahrt werden. JESSICA kennt zwei Arten von Objekten: Abstrakte und Konkrete. Die finalen HTML-Dokumente werden durch eine Kombination dieser Objekte generiert. Als Vorlagen für Dokumente werden Templates verwendet.

Der Jessica Compiler ist ein Java-basierter Prozessor, dessen Aufgabe die Auflösung der Objekte ist. Dazu wird ein iterativer Prozess mit Fixpunkt-Strategie eingesetzt.

4 Web-Application-Server

Jede Webseite läuft in irgendeiner Form auf einem **Web-Application-Server**. Zunächst soll geklärt werden worum es sich bei einem Application-Server handelt und dann der Unterschied zum Web-Application-Server erläutert werden.

Der **Application-Server** ist ein Softwareprodukt das Thin-Clients unterstützt und sich um die Verwaltung der Client-Sessions kümmert. Zudem beinhält er die Business-Logik und verbindet Clients mit Back-End-Ressourcen. Der einzige Unterschied zum Web-Application-Server ist, dass dort die Server-Komponente per Web-Server aufgerufen wird. Ein PERL-basierter Web-Application-Server wäre zum Beispiel *Mason*. Ein paar der entscheidenden Features sind

Server-Side-Caching Daten können gecached werden und über Parameter wird entschieden, ob der Inhalt vom Server oder aus dem Cache generiert werden soll. Dabei ergibt sich die Problematik der Konsiszenz. Benutzer sollen möglichst aktuelle Inhalte bekommen.

Templating Basierend auf der Verzeichnisstruktur wird der Template-Baum abgearbeitet (beginnend mit dem “Base”-File)

Staging erlaubt Trennung von Entwicklungs- und Produktionsumgebung, besonders für Services die hohe Verfügbarkeit benötigen.

Debugging und Error-Handling Detaillierte Fehlerberichte, alternativ ein Default-File in der Produktionsumgebung.

5 Software-Architekturen im Web-Engineering

Software lässt sich auf mehrere **Ebenen** unterteilen. Ganz oben findet sich die **Applikationsebene**, die unabhängig von speziellen Platformen ist und üblicherweise auf einer bestimmten **Middleware** aufbaut. Die Middleware dient der Maskierung von heterogenen Betriebssystemen und definiert ein Programmiermodell. Darunter findet sich das **Betriebssystem** selbst, welches eine Schnittstelle zu den Systemressourcen oder der **Hardware** bietet. Hardware und Betriebssystem fasst man auch als "Platform" zusammen.

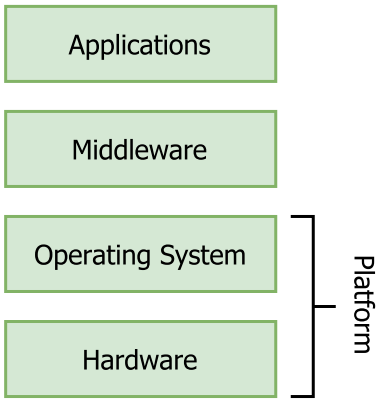

Abbildung 3: Software Ebenen

Bezüglich **Client-Server-Architekturen** gibt es verschiedene Ausprägungen, die klassische Architektur, wie etwa in *Abbildung* 4 zu sehen.

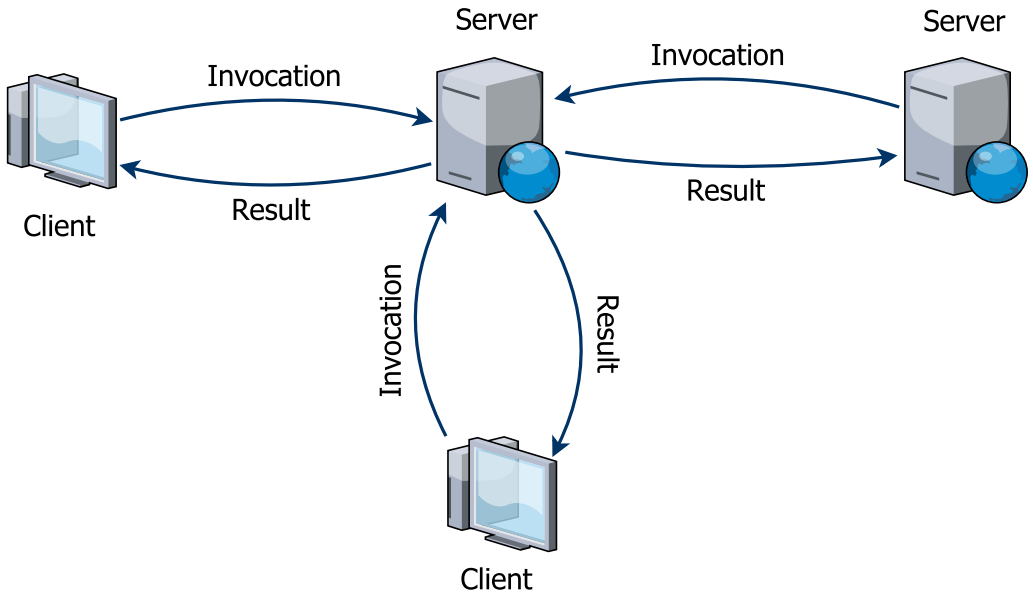

Abbildung 4: Client-Server-Architektur

Eine andere Ausprägung wäre das **Multi-Server-Modell** das in *Abbildung* 5 zu sehen ist.

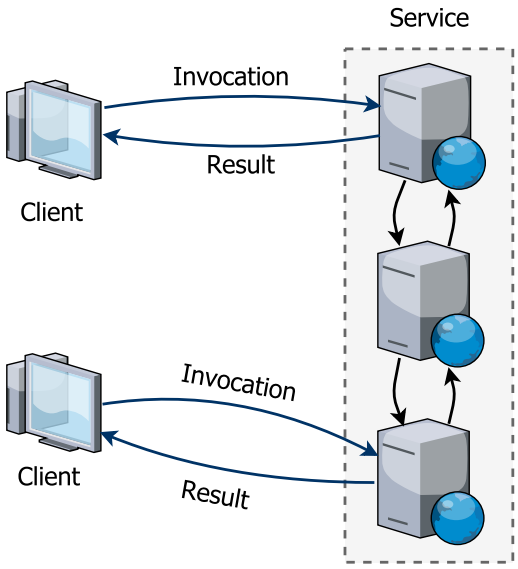

Abbildung 5: Multi-Server-Architektur

Der Grund warum man sich eigehend mit den verschiedenen Architekturen auseinandersetzen sollte liegt in den erzielten Effekten, allen voran dem Effekt auf die **Performanz**. Benutzer stellen gewisse Anforderungen an die Leistungsfähigkeit eines Systems, darunter:

Antwortverhalten Möglichst schnell, dazu benötigt man wenige Komponenten, lokale Kommunikation und Austausch kleiner Dateneinheiten

Durchsatz Die Rate mit der Berechnungen durchgeführt werden, schwächste Komponente bestimmt den Durchsatz

Load-Balancing Verlagerung von Arbeit auf weniger belastete Komponenten, dadurch kann Replikation erforderlich werden

Die Verbindung zwischen Client und Server ist im Normalfall eine **One-Time-Connection** und persistent (TCP-Verbindung).

Serverseitig gibt es wiederum verschiedene architektonische Möglichkeiten.

- Monolithische Server
- Parallele Prozesse mit Dispatcher
- Light- oder Heavy-Weight-Server
- Load-Balancing-Server
- Monitored-Server

Um ein paar konkrete Ansätze aufzugreifen sei etwa der **Apache Webserver** angeführt, der einem Cluster-Modell folgt, das bedeutet er kann sowohl auf einem einzelnen als auch auf mehreren Rechnern ausgeführt werden. Intern weist er einen modularen Aufbau auf.

Generell werden bei einem **Server-Cluster** mehrere gleichwertige Server von einem Frontend aus angesprochen. Eigenschaften eines Server-Clusters sind:

- Viele Hardware-Server werden zu einem logischen Gesamtsystem
- Requests können von mehreren Server beantwortet werden
- Ein Dispatcher entscheidet wer den Request erhält

- Zufällig
- Load-Balancing
- Round-Robin
- Membership-Management

Angewandt wird dieses Prinzip beispielsweise bei Web-Proxy-Servern, Web-Cache-Servern oder CDNs (Content-Delivery-Netzwerken).

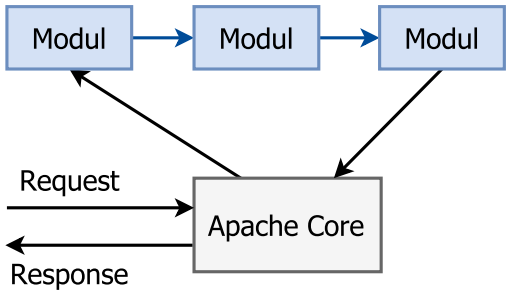

Abbildung 6: Apache modulbasierte Architektur

6 Content-Management

Content Trennung zwischen Struktur, Inhalt und Darstellung

Content-Management Erstellen und Verwalten von Inhalten

Content-Management-System (CMS) Software für Content-Management

Ein Web-Content-Management-System ist ein Redaktionssystem zur Verwaltung einer komplexen Website mit Hilfe von Techniken des Web-Engineerings und des Content-Managements.

Auf dem Markt gibt es eine Vielzahl an verschiedenen CMS, dabei handelt es sich zum einen um **CMS-Produkte**, das sind kommerzielle wie auch Open-Source Applikationen die sich weiter Verbreitung erfreuen. Zum Anderen um **CMS-Lösungen** die kundenspezifisch entwickelt werden und speziell an die Bedürfnisse des Kunden angepasst sind.

Das Projektvorgehen beim Einführen eines CMS umfasst einige Schritte:

1. Auswahl eines Projektteams
2. Umstellung der Arbeitsprozesse (Change-Management)
3. Eventuell externe Beratung

Bei der **IST-Analyse** soll in Form von Interviews mit Mitarbeitern eine Bestandsaufnahme durchgeführt werden um festzustellen wie das aktuelle Angebot aufgebaut ist und welche Wünsche die Mitarbeiter an ein neues CMS haben. Bei der **SOLL-Analyse** werden die Anforderungen an ein neues CMS evaluiert (dezentrale Inhaltspflege, Rollenkonzept, Platformunabhängigkeit, etc.)

Zu den Kernkomponenten eines CMS zählen

- Bereits erwähnte Trennung von Struktur, Inhalt und Darstellung
- Workflow-Komponente

  - Publikationsprozess
  - Informationsverarbeitungsprozess
  - Content-Life-Cycle
- Asset-Management zum Editieren und Verwalten von Inhalten
- Benutzer- und Zugriffsverwaltung
- Schnittstellen für externe Erweiterungen
- Import und Exportfunktion
- Datenbankanbindung

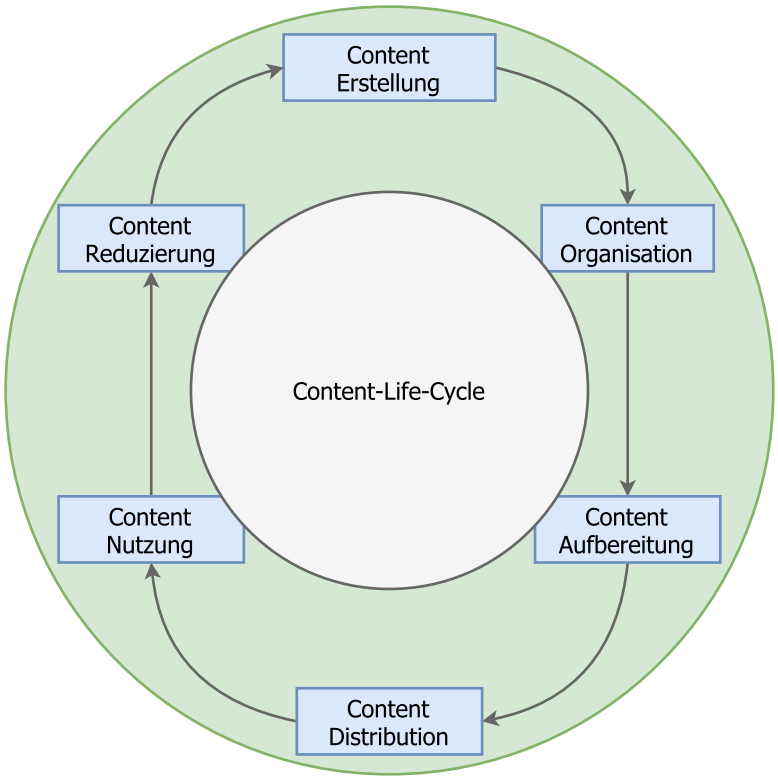

Abbildung 7: Content-Life-Cycle

Einige der **Kriterien** zur Beurteilung von Web-Content-Management-Systemen:

- Wonach unterscheiden sich CMS?
- Welche Features sollen enthalten sein?
- Preis-Leistungsverhältnis

Mandantenfähigkeit Gibt es Unterstützung für die Administration mehrerer Unternehmen bzw. können Unternehmen unabhängig voneinander verwaltet werden?

Workflow-Unterstützung Wie gut werden Arbeitsabläufe unterstützt?

Manuelle Workflows Einfachste Form, ohne Automatismen

Statische Workflows Vordefinierte Workflows nach dem 4-Augen-Prinzip

Benutzerdefinierte Workflows Modellierung komplexer Arbeitsabläufe

Versionskontrolle Änderungen können korrigiert werden, ältere Versionen können archiviert und wiederhergestellt werden.

Kollaboration Unterstützung von Benutzerinteraktion mittels Messaging oder TODO-Listen

Mehrsprachigkeit Besteht die Möglichkeit Dokumente in anderen Sprachen anzuzeigen?

Skalierbarkeit Kann Leistungsfähigkeit dem Bedarf angepasst werden?

7 Security

Security verfolgt mehrere Ziele, darunter die Bereitstellung von Authentifizierungsmechanismen, Sicherheit von Geschäftsvorgängen und den Schutz von personenbezogenen- sowie Zahlungsdaten. Dazu können zwei unterschiedliche Ansätze gewählt werden:

Mechanische Sperren Das sind Zulassungsbeschränkungen auf Benutzer-, Rechner- oder Domainebene. Dazu zählen etwa Firewalls.

Logische Sperren Hier bleibt die Zugänglichkeit der Daten erhalten, sie sind allerdings nur unter bestimmten Umständen erreichbar. Zu diesem Zwecke kommt die Verschlüsselungstechnik zum Einsatz.

Bei der **symmetrischen Verschlüsselung** verwenden Sender und Empfänger denselben Schlüssel. Beide müssen zur Ver- und Entschlüsselung dieselben Schritte durchführen (in umgekehrter Reihenfolge). Die Länge des Schlüssels gilt als Maßeinheit für die Sicherheit.

Asymmetrische Verfahren beruhen auf Public- und Private-Keys. Der Public-Key ist für jeden einsehbar während der Private-Key nur dem Absender bekannt sein sollte. Eine Kombination dieser beiden Schlüssel erlaubt verschiedene Varianten der Datenverschlüsselung.

Möchte man eine Nachricht an alle verschicken so benutzt man zur Verschlüsselung den eigenen privaten Schlüssel sodass Empfänger sie mit dem eigenen öffentlichen Schlüssel entschlüsseln können.

Möchte man eine Nachricht an eine bestimmte Person verschicken so empfiehlt sich zur Verschlüsselung dessen öffentlicher Schlüssel. Dadurch kann die Nachricht nur mit dem privaten Schlüssel des Empfängers wieder entschlüsselt werden.

Bei einer 1-zu-1 Kommunikation muss die Nachricht zunächst mit dem privaten Schlüssel des Absenders und anschließend mit dem öffentlichen Schlüssel des Empfängers verschlüsselt werden.

Neben der Verschlüsselung bieten asymmetrische Verfahren auch die Möglichkeit der **Authentifizierung**, dazu muss lediglich ein Teil oder die gesamte Nachricht mit dem privaten Schlüssel des Absenders chiffriert werden. Somit ist gewährleistet, dass nur der Besitzer des privaten Schlüssels - ergo der Absender - Urheber der Nachricht sein kann.

Die **digitale Unterschrift** ist eine Spezialform der Authentifizierung mit dem Unterschied, dass die Nachricht im Klartext übertragen wird und zusätzlich ein Message-Digest mitgeschickt wird, der mit dem privaten Schlüssel des Absenders verschlüsselt ist. So kann der Empfänger einerseits überprüfen ob der Message-Digest korrekt ist, indem er ihn mit dem mitgelieferten vergleicht und andererseits die Authentizität des Absenders überprüfen.

Mittels **digitaler Zertifikate** kann ein vertrauenswürdiger Dritter, auch bekannt als Certification-Authority (CA), die Authentizität eines Informationsanbieters bestätigen. Webserverbetreiber

können bei solchen Institutionen um ein Zertifikat ansuchen und sich damit eindeutig ausweisen. Das Zertifikat enthält

- Public-Key des Servers
- Serverbezeichnung (URL)
- Ablaufdatum

Das Zertifikat ist zudem mit dem privaten Schlüssel der CA verschlüsselt.

8 Web-Services

Web-Services dienen in erster Linie dem Informationsaustausch zwischen Applikationen.

Begriff	**Bedeutung**
HTTP	Transportprotokoll
SOAP	Simple Object Access Protokoll
UDDI	Universal Description Discovery and Integration
WSDL	Web Service Description Language
XML	Präsentationsschicht

SOAP erlaubt Funktionsaufrufe auf entfernt gelegenen Servern, es baut auf HTTP als Transportprotokoll auf und verwendet XML als Notation für Argumente und Ergebnisse. Aufrufe und Ergebnisse sind in „SOAP Envelopes“ verpackt.

WSDL wird zur Beschreibung von Web-Services genutzt. Die Beschreibung umfasst Zugriffsdetails sowie die unterstützten Operationen des Service. Sprachlich basiert WSDL ebenfalls auf XML.

9 Social-Web und AJAX

Charakteristika von modernen Web-Anwendungen

- Web wird als Platform verstanden
- Daten-getriebene Anwendungen
- Vernetzung durch “Architektur des Mitwirkens” (Wikipedia uä.)
- Einfache Geschäftsmodelle
- Projekte befinden sich immerwährend im Beta-Stadium

AJAX ist keine neue Programmiersprache sondern eine Technik die es ermöglicht interaktive Web-Applikationen zu erstellen. Basiert auf Javascript um Daten zwischen Browser und Server auszutauschen. Das Grundprinzip ist, dass Daten hinter den Kulissen “ausgetauscht” werden anstelle eines kompletten Neuladens der gesamten Seite.

AJAX ist zur Gänze auf der Client-Seite angesiedelt und verwendet HTTP für den asynchronen Datentransfer. Dadurch werden Web-Applikationen kleiner, schneller und benutzerfreundlicher.