Netzbasierte Informationssysteme
Die Architektur des Web II

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierte Informationssysteme
mailto: tolk@inf.fu-berlin.de
http://www.robert-tolksdorf.de

Representational State Transfer
REST

Nach

Roy T. Fielding and Richard N. Taylor. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2), May 2002, pp. 115-150.

Roy T. Fielding. Architectural styles and the design of network-based software architectures. *PhD Thesis*, University of California, Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Software Architektur

- Software Architecture: Abstraktion von den ausführenden Elementen eines Systems
  - Ansammlung von Komponenten (Components)
  - Beschreibung der Interaktion zwischen ihnen (Connectors)
    - Prozeduraufruf
    - Broadcast von Ereignissen
    - Protokolle
    - Pipes (siehe UNIX-Shell)
    - ...
- Architectural Style: Menge von Einschränkungen des Verhaltens von Komponenten eines Systems

A layered system is organized hierarchically, each layer providing service to the layer above it and serving as a client to the layer below. In some layered systems inner layers are hidden from all except the adjacent outer layer, except for certain functions carefully selected for export. Thus in these systems the components implement a virtual machine at some layer in the hierarchy. (In other layered systems the layers may be only partially opaque.) The connectors are defined by the protocols that determine how the layers will interact. Topological constraints include limiting interactions to adjacent layers. Figure 2.4 illustrates this style.

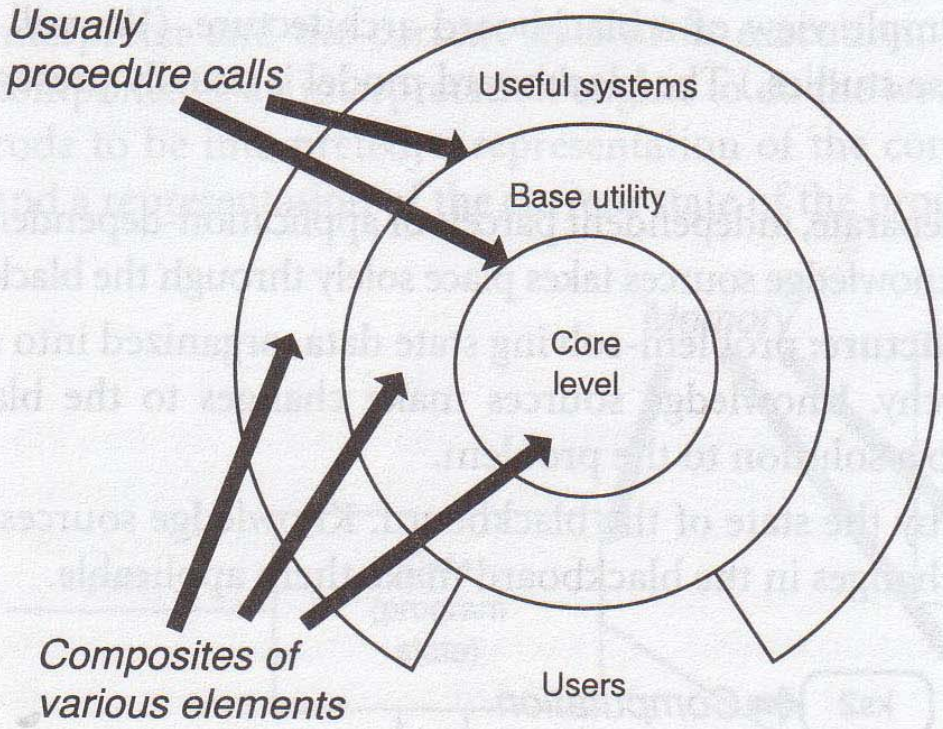


FIGURE 2.4 Layered Systems

[Mary Shaw und David Garla. Software Architecture - Perspectives on an Emerging Discipline. Prentice Hall, 1996, ISBN 0-13-182957-2]
In Teilen als [David Garlan and Mary Shaw. An Introduction to Software Architecture. Technical Report CMU-CS-94-166. Carnegie Mellon University. 1994. http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf]

- Was ist die Software-Architektur des Web?
- REST: Representational State Transfer
  - Minimierung von
    - Netzwerklatenz und
    - Netzwerkverkehr
  - Maximierung von
    - Unabhängigkeit und
    - Skalierbarkeit

    von Komponenten

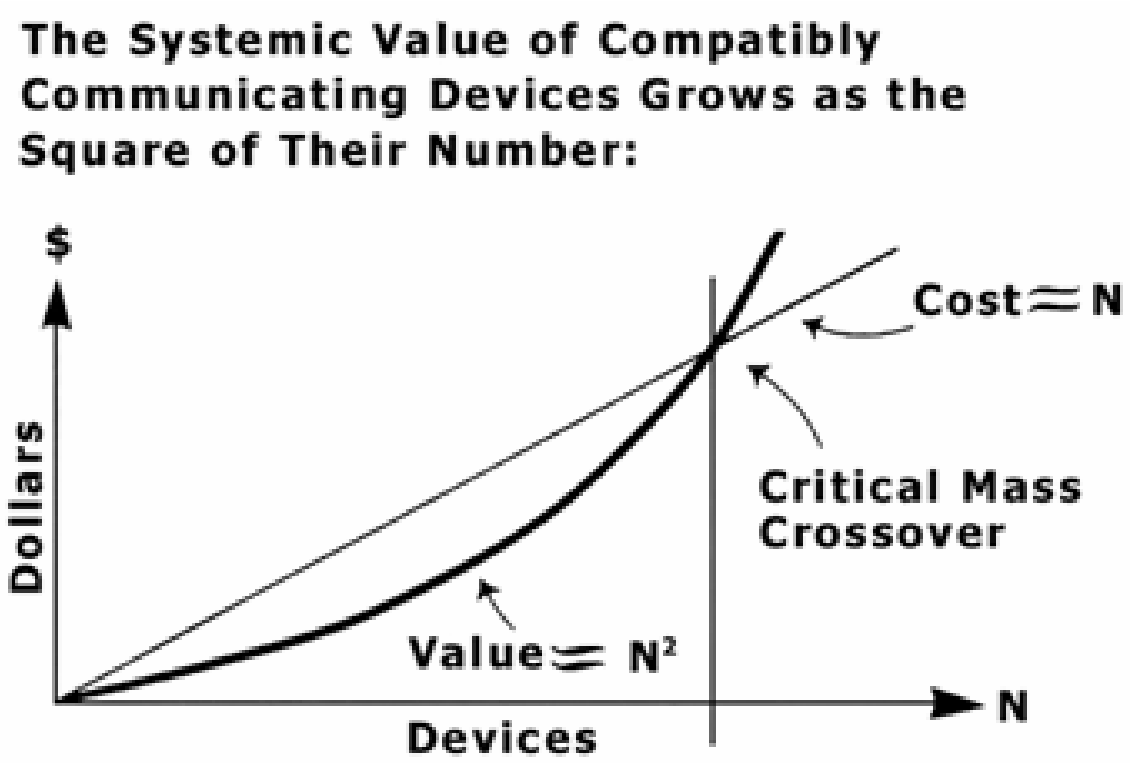
Anforderungen

- Ziel beim Entwurf des Web:

  Erstellung eines
  - einheitlichen Zugangs
  - von unterschiedlichen Plattformen
  - zu Informationen, die an unterschiedlichen Orten
  - in unterschiedlichen Formen vorliegen,
  - der inkrementell eingeführt werden kann

Anforderungen

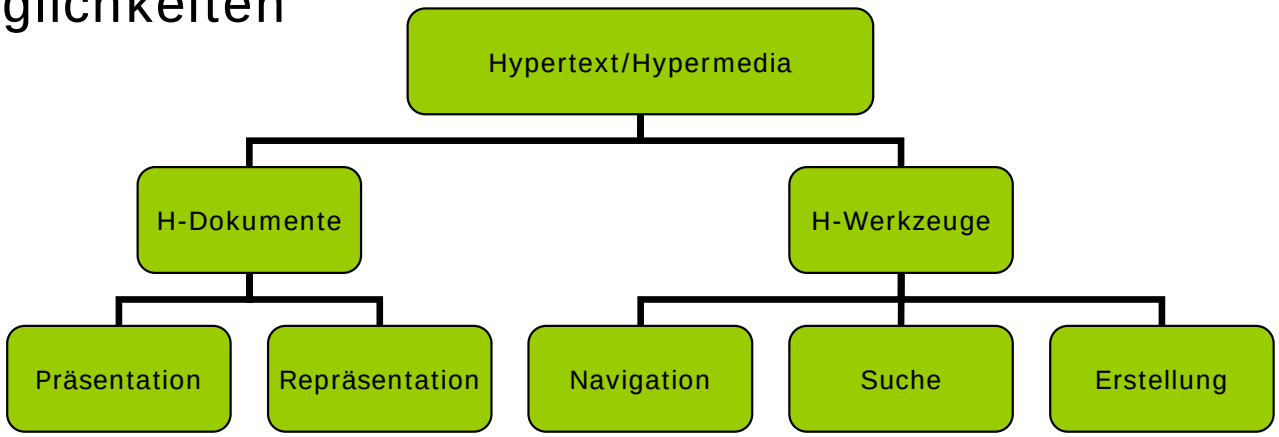
- Erstellung und Nutzung von Web-Inhalten ist *freiwillig*
- Architektur muss niedrige Eintrittsschwellen bieten um Netzwerkeffekte zu realisieren
  - Bob Metcalfe: Wert eines Netzwerks wächst quadratisch mit der Anzahl seiner Knoten



[http://vcmike.wordpress.com/2006/08/18/metcalfe-social-networks]

Anforderungen

- Anforderungen:
  - Für Nutzer: Hypermedia als Schnittstelle gewählt
    - Einheitliche Schnittstelle
    - Einfache Navigation
    - Einfache Suchmöglichkeiten



  - Für Autoren:
    - Unabhängigkeit vom Gesamtsystem
    - Informationen vor ihrer Entstehung referenzieren
  - Für Anwendungsentwickler:
    - Einfachheit

Anforderungen

- Erweiterbarkeit
  - Einfachheit ohne Erweiterbarkeit verhindert Evolution
  - Anforderungen an das System ändern sich
  - Architektur muss erweiterbar sein
- Verteiltheit
  - Repräsentationen von Informationen liegen an unterschiedlichen Orten
  - Nutzer soll keine zusätzliche Latenz spüren
  - Architektur muss Interaktionen über das Netz minimieren

Anforderungen

- Skalierbarkeit auf Internet-Größe
  - Ungestörter Betrieb bei Änderungen außerhalb der eigenen Organisation
    - Klienten sind unabhängig vom Wissen über alle Server
    - Server sind unabhängig vom Wissen über alle Klienten
    - Hypermedia Dokumente sind unabhängig von den Verweisen auf sie
  - Als Normalfall verzichtet man auf Sicherheit weil Authentifizierung Abhängigkeit von einer Vertrauensdomäne bedeutet
  - Unabhängige Evolution der Komponenten

Wahl des Architectural Style

- Das Web folgt dem Client-Server Stil

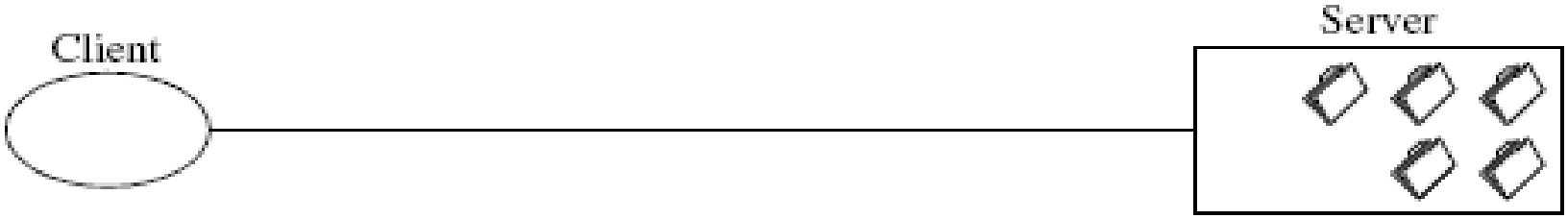


Figure 5-2. Client-Server

  - Trennung von
    - Aspekten der Nutzerschnittstelle (beim Klienten)
    - Aspekten der Datenhaltung (beim Server)
  - Höhere Portabilität der Nutzerschnittstelle
  - Bessere Skalierbarkeit der Server durch Einfachheit
    - *Skalierbarkeit*: Fähigkeit der Architektur, das aktive Zusammenspiel einer großen Menge von Komponenten oder einer großen Menge von Interaktionen zu unterstützen
  - Trennung erlaubt unabhängige Evolution der Komponenten

Wahl des Architectural Style

- Interaktion zwischen Klient und Server ist zustandslos

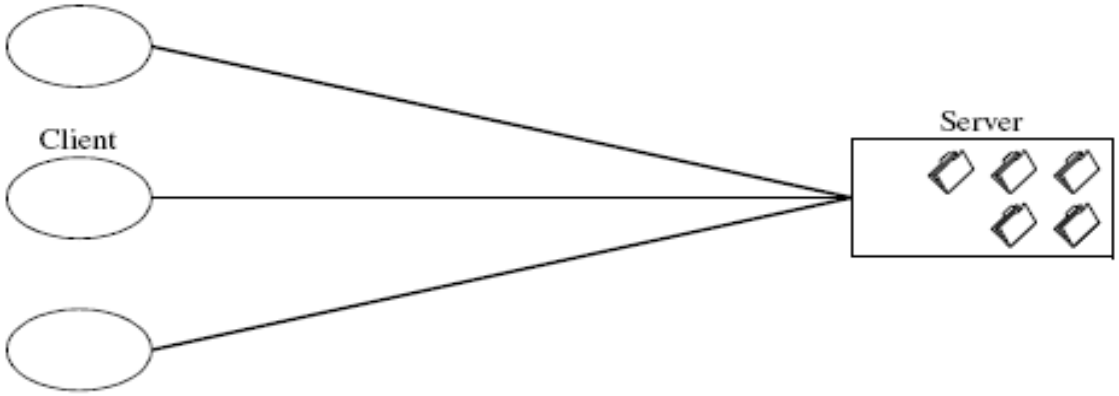


Figure 5-3. Client-Stateless-Server

  - Anfrage nicht relativ zu einem Kontext auf Server
  - Zustand muss beim Klienten gehalten werden
  - Visibility erhöht, da nur eine Mitteilung betrachtet
    - *Visibility*: Fähigkeit von Komponenten der Architektur, Interaktionen zweier anderer Komponenten zu beobachten oder zu vermitteln
  - Zuverlässigkeit erhöht, weil nach Fehlern keine Historie nachgespielt werden muss
  - Skalierbarkeit erhöht, weil der Server keine Zustandsinformation speichern und managen muss
  - Nachteil: Mehr Netzverkehr wg. Zustandsübermittlung

Wahl des Architectural Style

- Mitteilungen können in einem Cache gespeichert und später wiederverwendet werden

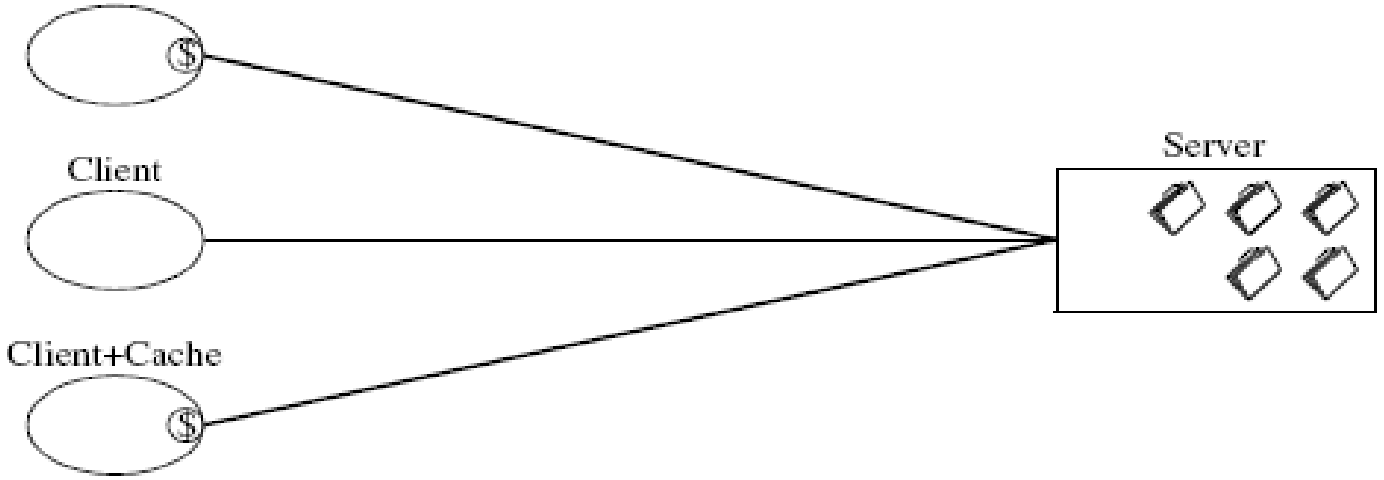


Figure 5-4. Client-Cache-Stateless-Server

  - Ausgetauschte Mitteilungen müssen als zwischenspeicherbar erkennbar sein
  - Interaktionen werden vermieden, Netzwerklatenz sinkt
  - Nachteil: Zuverlässigkeit kann durch Inkonsistenzen sinken

Wahl des Architectural Style

- Zwischen Komponenten existieren einheitliche Schnittstellen
  - Gleiche Identifikation von Ressourcen
  - Ressourcen werden über Repräsentationen modifiziert
  - Selbstbeschreibende Mitteilungen
  - Hypermedia als Ausführungssystem

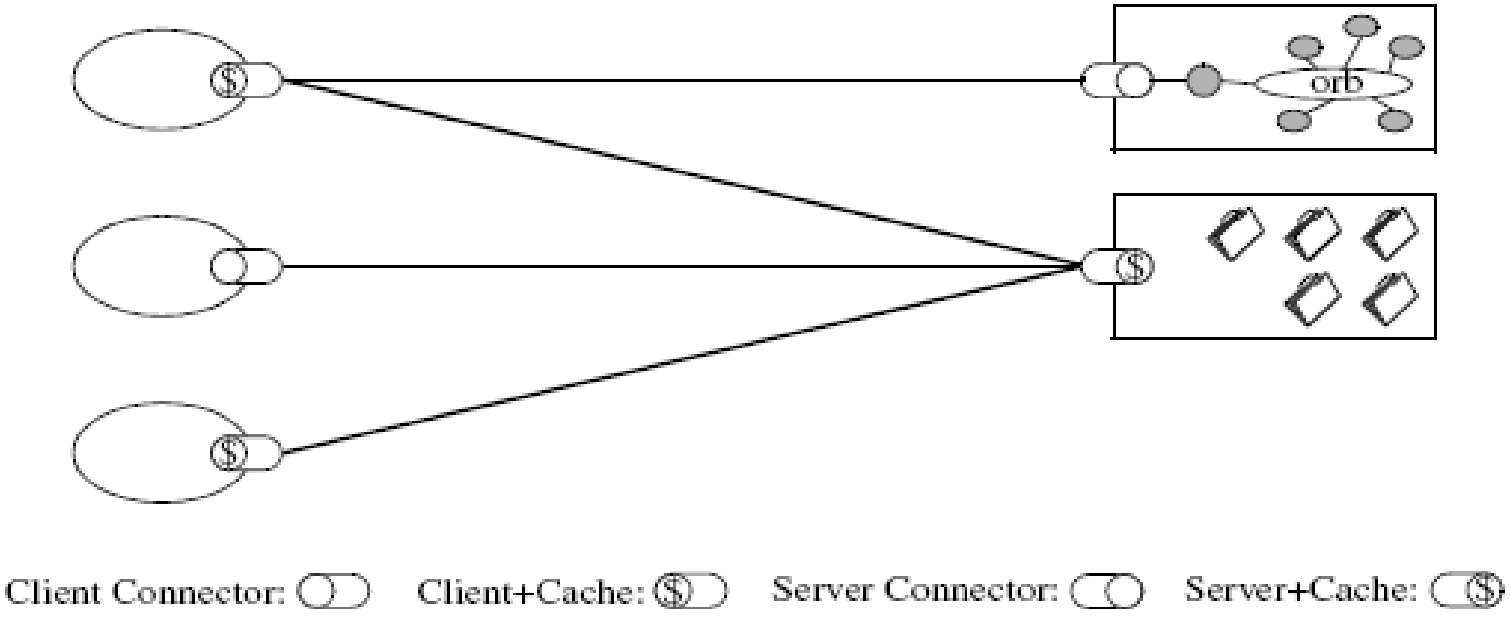


Figure 5-6. Uniform-Client-Cache-Stateless-Server

Wahl des Architectural Style

- Schichtenarchitektur
  - Reduktion von Komplexität und Abhängigkeiten
  - Nachteil: Overhead durch Verarbeitung über mehrere Schichten hinweg
  - Ausgeglichen durch Caches

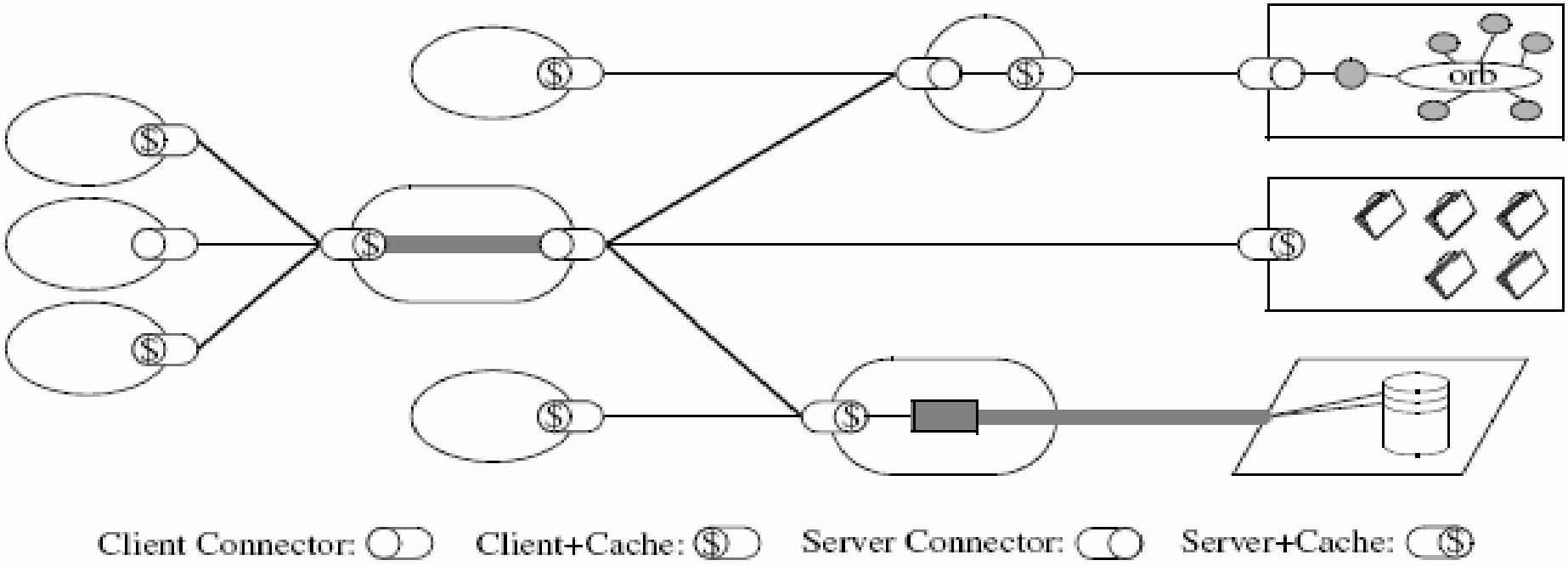


Figure 5-7. Uniform-Layered-Client-Cache-Stateless-Server

Wahl des Architectural Style

- Code on demand
  - Funktionalität des Klienten kann durch nachgeladenen Code erweitert werden
  - Nachteil: Visibility sinkt

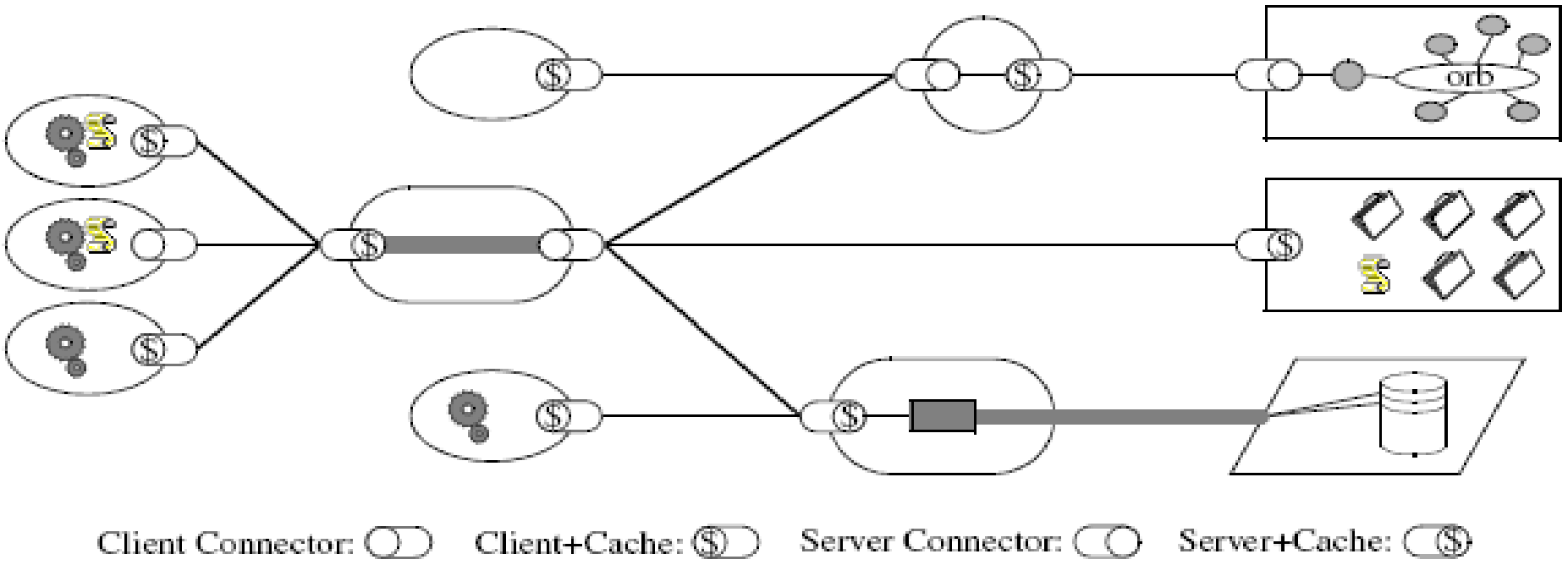


Figure 5-8. REST

REST Architektur / Daten

Freie Universität Berlin

- Ressourcen: Abstraktion von einer Information („Wetter in Berlin“)
- Ressource ist eine Abbildung zu einer Zeit auf eine Menge von
  - Repräsentationen (HTML-Seite mit Beschreibung des Wetters,Bild)
  - Identifikationenen (URL der Beschreibungsseite, des Bilds)
- Konzept von Ressourcen
  - Abstrahiert von einer möglichen Repräsentation
  - Erlaubt späte Bindung eines Werts an die Identifikation
- Identifizierungen werden *dezentral* gewählt und verwaltet
- Repräsentation
  - Darstellung des Zustands einer Ressource
  - Bytestrom + Metadaten
    - Kontrolldaten: Was soll mit der Repräsentation wie geschehen (z.B. PUT Methode oder Antwortcode)
    - Medientyp

REST Architektur / Connectors

- Conntector: Schnittstelle über die mit anderen Komponenten kommuniziert wird
- Vorteile der Zustandslosigkeit:
  - Connector muss keinen Zustand halten
  - Keine Abhängigkeiten zwischen Interaktionen ⇨ Parallelität
  - Zwischengeschaltete Komponenten „verstehen“ komplette Interaktion
  - Zwischenspeichern komplett möglich
- Typen in REST:
  - Client (HTTP-Bibliothek beim Klienten)
  - Server (HTTP-Bibliothek beim Klienten)
  - Cache (im Browser)
  - Resolver (DNS)
  - Tunnel (SSL über HTTP)

REST Architektur / Komponenten

- Component: Ausgeführter Prozesse, nach seiner Aufgabe getypt
- Typen in REST
  - User agent (Browser)
  - Origin server (Apache)
  - Proxy – vom Klienten gewählt (CERN Proxy etc.)
  - Gateway – vom Server festgelegt (CGI)

- REST ganz kurz:
  - Alles ist komplett durch eine URI beschrieben
  - Ein Zugriff auf eine URI ist komplette Interaktion

Datenformate
HTML

Hypertext Markup Language

Freie Universität Berlin

- Dominierende Sprache zur Auszeichnung von Dokumenten im Internet
- Definiert vom World Wide Web Consortium, W3C:
  - MIT (Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory (CSAIL))
  - ERCIM (European Research Consortium in Informatics and Mathematics)
  - Keio University of Japan
- Jedes Informationssystem im Netz muss:
  - HTML Informationen integrieren können
  - HTML Ausgaben erzeugen
  - Mit HTML-Mitteln mit Nutzern interagieren

HTML

- Sprache umfasst
  - Elemente

```
<h1>Neue Vorlesungen</h1>
<br>
<hr>
```

  - Attribute

```
<hr height="3">
```

  - Entitäten

```
&amp;
&auml;
```

  - Grammatikalische Regeln über Elemente
    - <html> ist Startsymbol,
    - <html> kann die Elemente <head> und <body> enthalten

HTML Beispiel/1

```
<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.01 Transitional//EN>
<html>
  <head>
    <title>FU-Berlin: Institut für Informatik</title>
    <base href="http://www.inf.fu-berlin.de">
  </head>
  <body>
      <p><a href="http://www.fu-berlin.de/">Freie Universit&auml;t
   Berlin</a><br>
      <a href="http://www.math.fu-berlin.de/">Fachbereich Mathematik
  und Informatik</a></p>
  <h1>Institut f&uuml;r Informatik</h1>

      <p><a href="http://www.inf.fu-
   berlin.de/index_en.html">Homepage
  in English</a>.</p>
```

HTML Beispiel/2

```
<form method="get" action="http://www.google.com/search">
<a href="http://www.google.de">Google</a>-Sitesearch:
<nobr><font size=2>
<input type=text name=q size=15 maxlength=255 value="">
</font></nobr>
<input type=hidden name=sitesearch value="inf.fu-berlin.de">
<input type=hidden name=domains value="inf.fu-berlin.de">

</form>
<h2>Aktuelle Meldungen</h2>
<p>Das Ferienprojekt <a href=
"http://www.inf.fu-berlin.de/~block/schachprojekt.h
Schachprogrammierung</a>
wird wegen der umfassenden Bauarbeiten im Institut auf die
nächsten Semesterferien verschoben!</p>
</body>
</html>
```

HTML – Elemente für Struktur

- Struktur
  - !DOCTYPE gibt Art des Dokuments an:
    <!DOCTYPE HTML PUBLIC
    -//W3C//DTD HTML 4.01 Transitional//EN>
  - <html>…</html> umfasst Dokument
  - <head>…</head> enthält Informationen zur Seite
  - <body>…</body> umfasst Inhalt der Seite

- Festes Seitenschema:

```
<!DOCTYPE…>
<html>
  <head>…</head>
  <body>…</body>
</html>
```

HTML – Elemente im Kopfteil

- <base> enthält Basis-Adresse der Seite
- <title> enthält Titel der Seite
  ```
  <title>FU-Berlin: Institut für Informatik</title>
  ```
- <meta> enthält
  - Inhaltsklassifikation der Seite
    ```
    <meta scheme="ISBN" name="identifier"
     content="0-8230-2355-9">
    ```
  - oder Protokollinformation
    ```
    <meta http-equiv="Expires"
     content="Tue 24 Sep 2002 00:00:00 GMT">
    ```
- <link> gibt Beziehung zu anderer Seite an
  ```
  <link rel="Glossary" href="URL">
  ```

HTML – Elemente für Gestaltung

Freie Universität Berlin

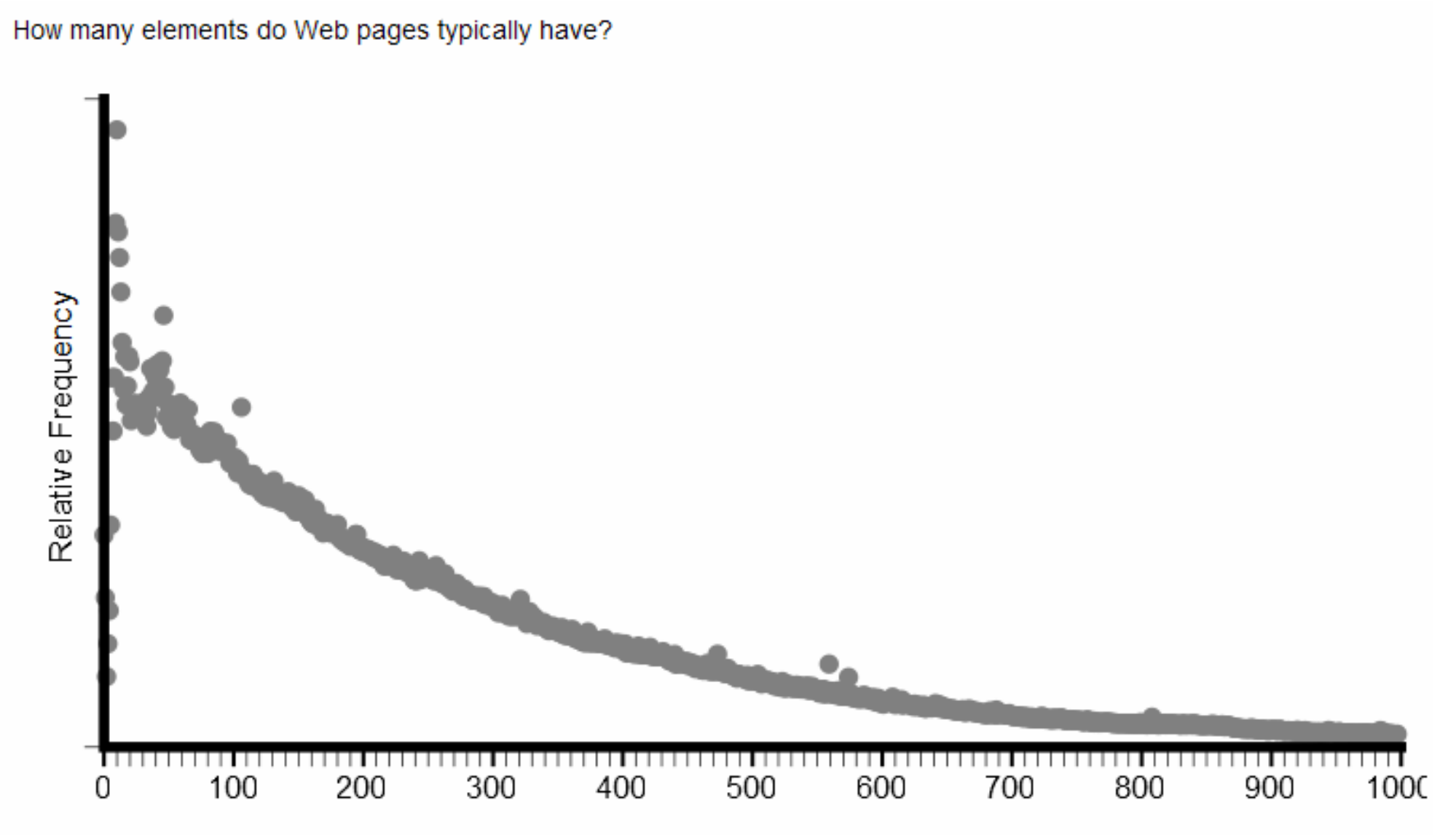
- Umbruch, Trennungen
  `wbr br nobr p spacer`
  ```
  Neue<br>Zeile
  ```
- Schriftarten
  `b i s strike tt u blink bdo marquee`
  ```
  Das ist <b>wirklich wichtig</b>
  ```
- Schriftauszeichnung
  `abbr acronym cite code del dfn em ins kbd samp strong var ruby rt rb`
- Formeln
  `sub sup`
- Schriftgröße
  `basefont font big small`
- usw.

HTML – Elemente für inhaltliche Strukturen

Freie Universität Berlin

- Überschriften
  h1 h2 h3 h4 h5 h6
  ```
  <h2>Aktuelle Meldungen</h2>
  ```
- Blöcke
  comment hr div span address pre xmp plaintext
  listing blockquote q banner multicol center
  ```
  <center>Blah blah bla <hr> blah
  blah</center>
  ```
- Listen
- Tabellen
- usw.

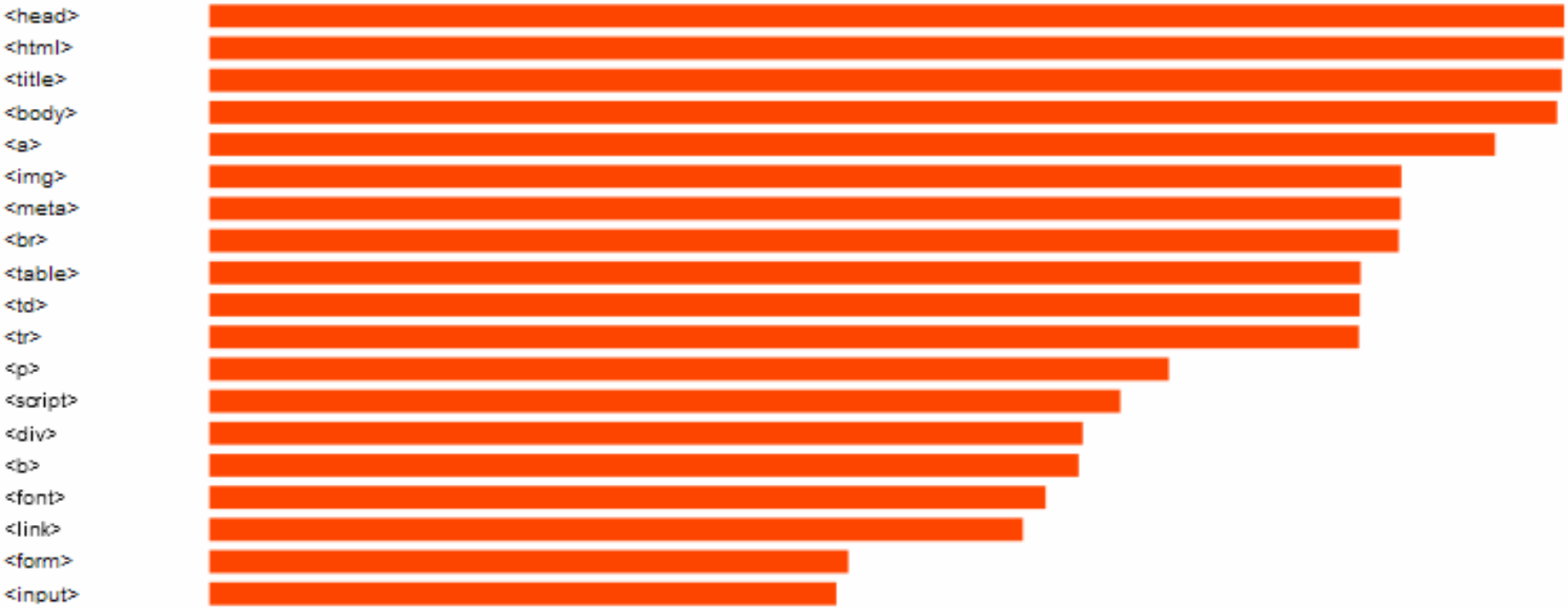
Verwendung von HTML Elementen nach Google Erhebung [http://code.google.com/webstats]



Verwendung von HTML Elementen nach Google Erhebung [http://code.google.com/webstats]

Freie Universität Berlin

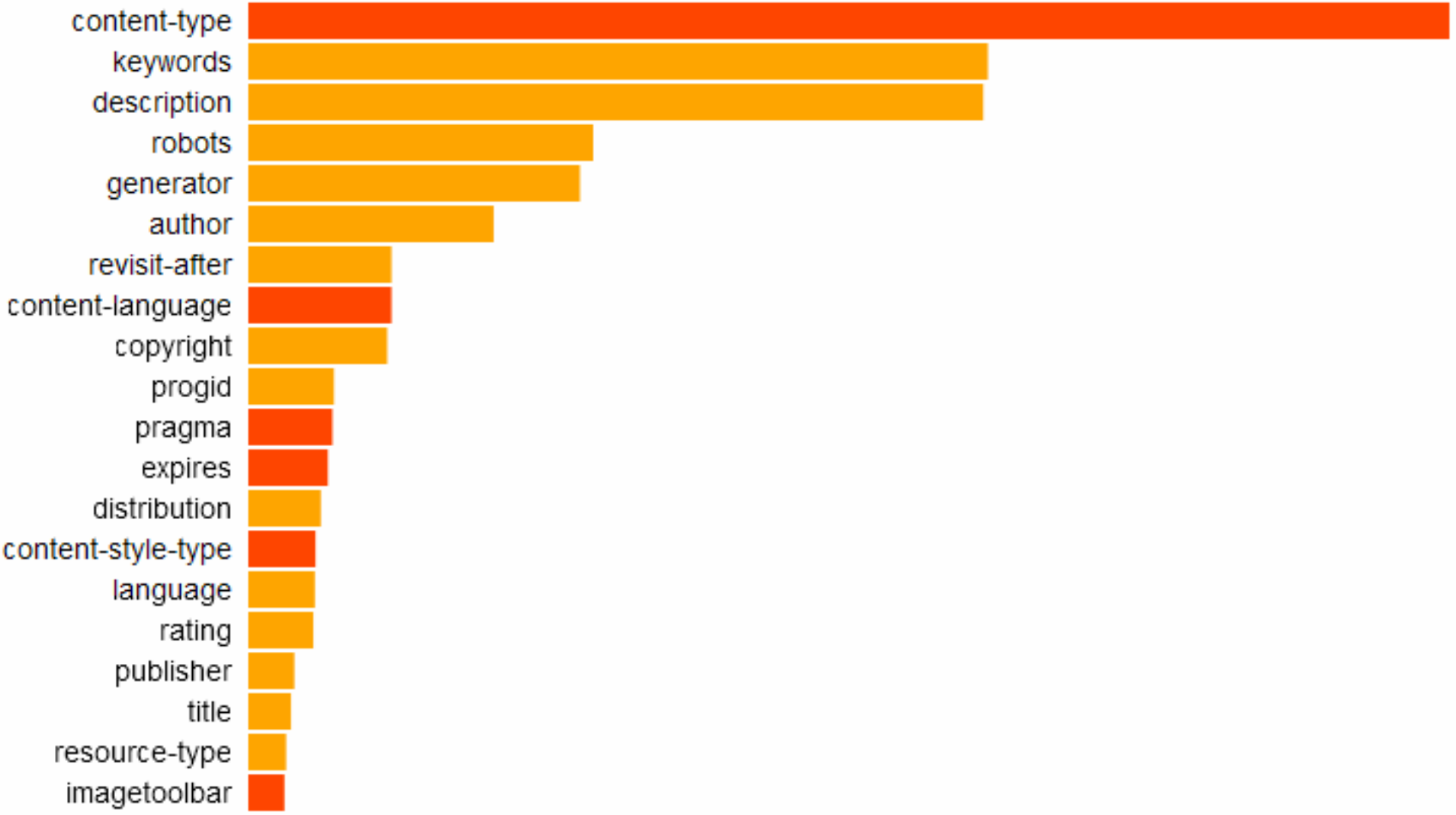
What are those elements? Well, the nineteen elements used on the most pages are:



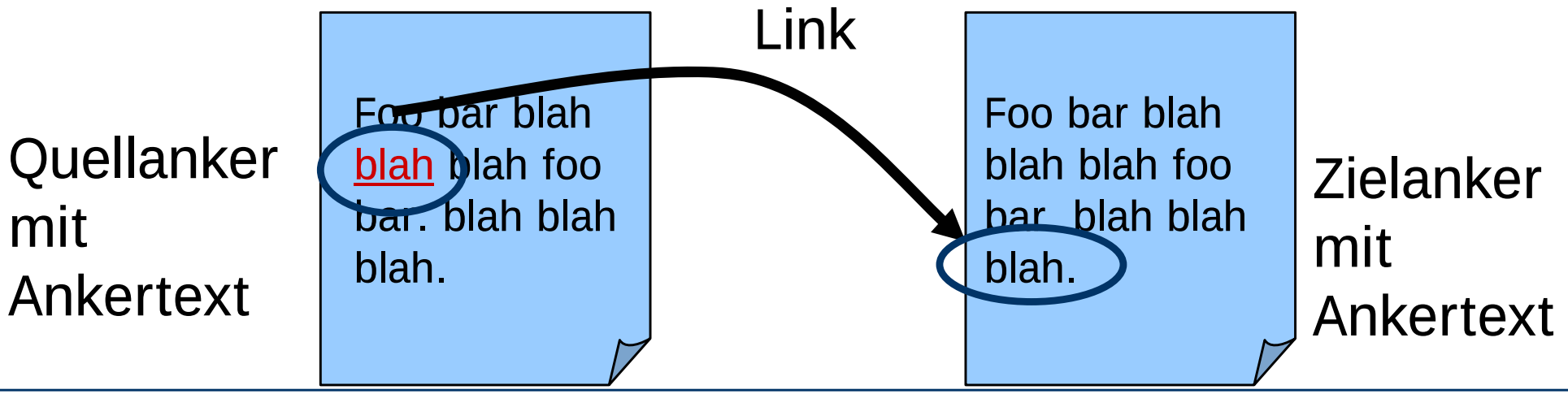
Verwendung von HTML Elementen nach Google Erhebung [http://code.google.com/webstats]

Freie Universität Berlin

- Verwendung von http-equiv und name bei <meta>:



- Konzepte:
  - Informationen werden als Dokumente aufgefasst
  - Dokumenteninhalte werden als Klartext dargestellt
  - Dokumententeile werden durch *Markierungen/Elemente/Tags* ausgezeichnet
    - Inhaltlich (<h1>Einleitung</h1>, wichtig)
    - Gestalterisch (wichtig)
  - Dokumente werden durch Links zu einem Hypertext verbunden (dadurch entsteht ein Netz, das Web)



Lebensdauer von Web-Referenzen

- Wie lange sind URLs in der Regel nutzbar?
- Studie
  Diomidis Spinellis. The decay and failures of web references. Communications of the ACM, Volume 46, Number 1 (2003), Pages 71-77
- Untersucht
  - URL Referenzen in Artikeln der Zeitschriften
    - Communications of the ACM
    - IEEE Computer

- Befürchtung:
  Zunehmende Verwendung von URLs in der wissenschaftlichen Literatur ist bedrohlich für die wissenschaftliche Qualität wenn URLs nicht mehr auflösbar sind.

Antwort Codes

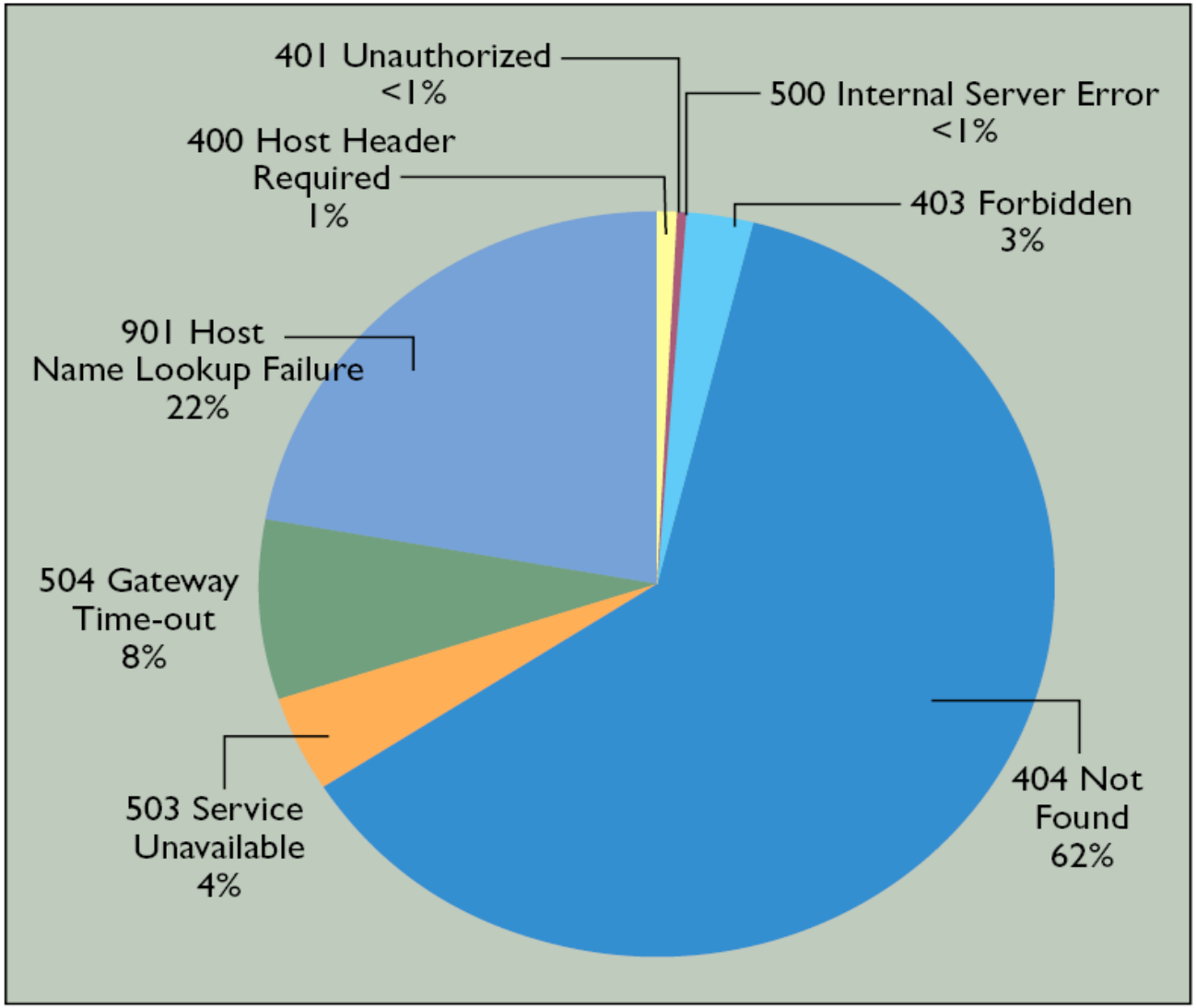
- 400-er Codes: Nicht erfolgreich, Fehler bei Client
  - 400 - Bad Request
    Falsche Anfragesyntax
  - 401 - Unauthorized
    Passwort notwendig
  - 403 – Forbidden
    Ohne Angabe von Gründen verweigert
  - 404 - Not Found
    Nicht auffindbar
  - 405 - Method Not Allowed
    Methode für die Ressource nicht zugelassen
  - 406 - Not Acceptable
    Information vorhanden aber nicht passend zu Accept-Kopfzeilen
  - 407 Proxy Authentication Required
    Zuerst Authentifizierung bei Proxy nötig, der Proxy-Authenticate Kopfzeilen mit schicken muss
  - 408 - Request Timeout
    Timeout bei Übermittlung der Anfrage
  - ...

Antwort Codes

- 500-er Codes: Nicht erfolgreich, Fehler bei Server
  - 500 - Internal Server Error
  - 501 - Not Implemented
    Angeforderte Methode nicht unterstützt
  - 502 - Bad Gateway
    Weiterer benutzter Server nicht erreichbar
  - 503 - Service Unavailable
    Server kann Dienst gerade nicht erbringen (Retry-After Kopfzeile)
  - 504 - Gateway Timeout
    Weiterer benutzter Server antwortet nicht rechtzeitig
  - 505 HTTP Version Not Supported
    Unbekannte HTTP Version

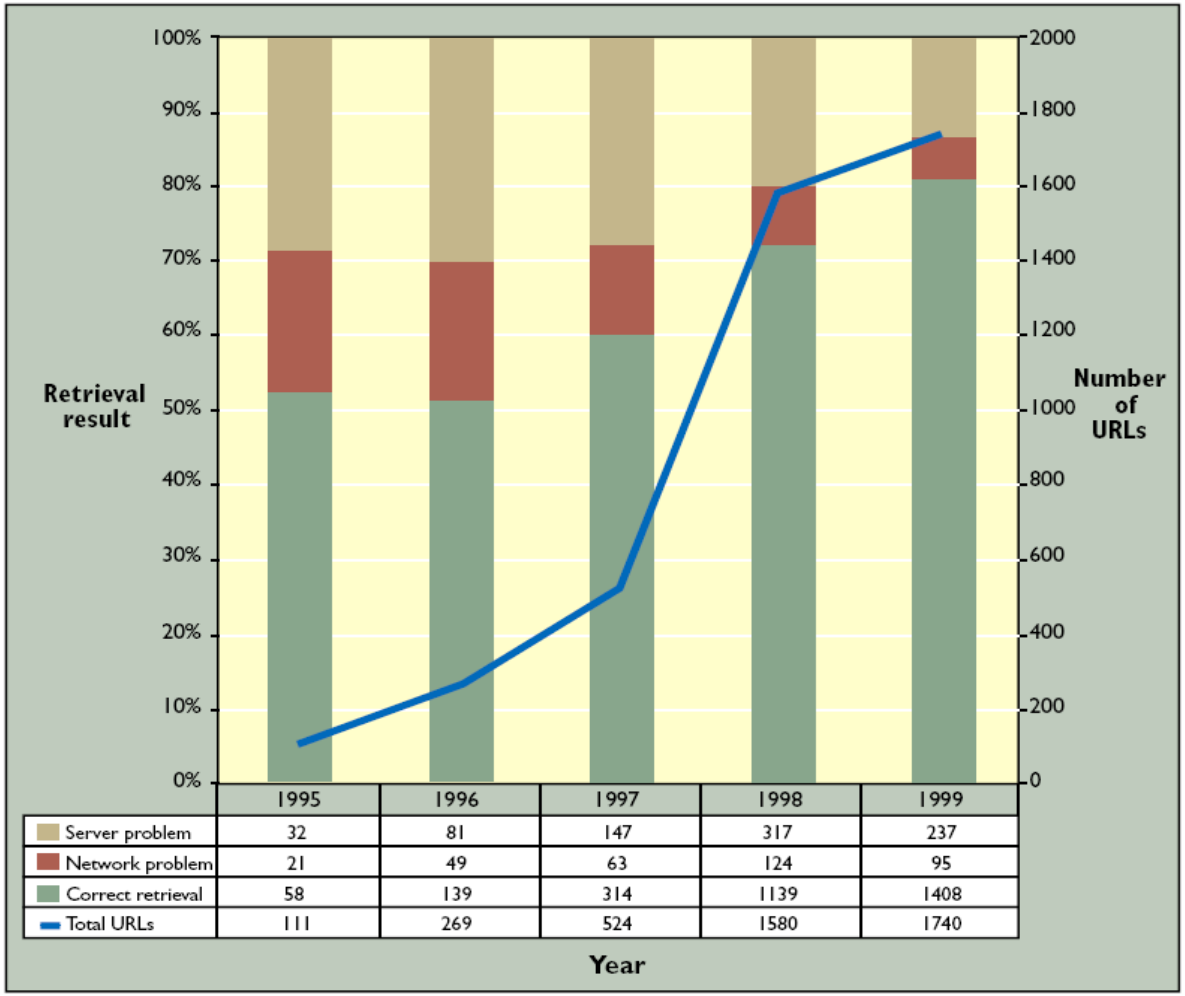
Lebensdauer von Web-Referenzen

- Aus ACM und IEEE DL insgesamt 2471 Artikel geladen
- Daraus 4224 URLs extrahiert
  - 1,7 URLs/Artikel
  - 1,49 Artikel/URL
- 72% erreichbar
- Von den verfallenen URLs
  - 83% mit falschem Host oder Pfad („901“ und 404)
  - nur 8% Netzwerk-probleme (504)



Lebensdauer von Web-Referenzen

- Halbwertszeit einer URL: 5 Jahre
- Ca. 10% URL-Verfall pro Jahr
  - Vorlaufverzögerung bis zum Erscheinen des Artikels
  - Stabilisierung auf gut gepflegten Servern
- Wiederholung der Untersuchung zeigt
  - Referenzen in neueren Artikeln sind etwas stabiler
    - Bessere Auswahl der Quellen
    - Bessere Wartung der Server

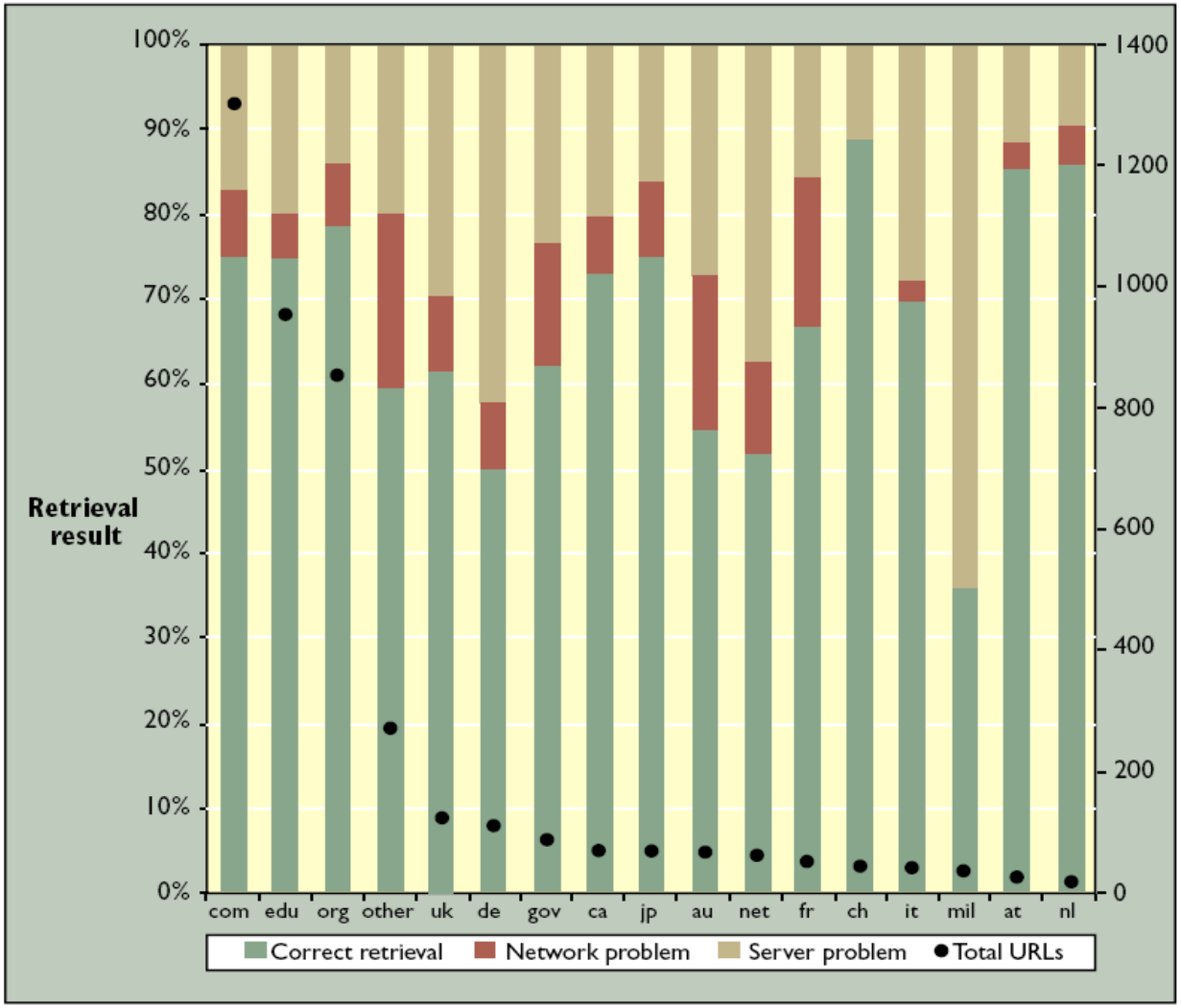


	1995	1996	1997	1998	1999
Server problem	32	81	147	317	237
Network problem	21	49	63	124	95
Correct retrieval	58	139	314	1139	1408
Total URLs	111	269	524	1580	1740

Lebensdauer von Web-Referenzen
Weitere Erkenntnisse

Freie Universität Berlin

- Tiefe Referenzen (lange Pfade) haben kürzere Erreichbarkeit als kurze Referenzen
- Referenzen auf Dateien haben schlechtere Erreichbarkeit als Referenzen auf Verzeichnisse
- Inhalte auf anderen Top-Level-Domains als .edu haben ähnliche Erreichbarkeit



Datenformate
XML

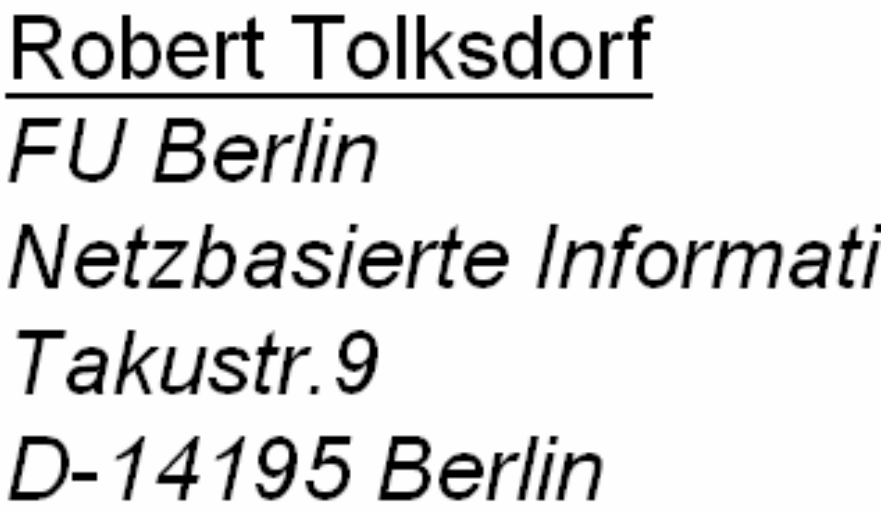
XML Q&A

- *Was ist XML?*
  Die Extensible Markup Language ist die Definition einer Untermenge von SGML, mit der man einfach Auszeichnungssprachen definieren kann
- *Woher kommt XML?*
  XML ist ein Standard des World Wide Web Konsortiums W3C
- *Was macht man mit XML?*
  Anwendungsspezifische Auszeichnungssprachen definieren und standardisieren
- *Was ist der Vorteil von XML-basierten Auszeichnungssprachen?*
  Standardisierung ermöglicht Datenaustausch

Auszeichnungssprachen

- Auszeichnungssprachen fügen *Markierungen* zu einem Text hinzu
- Beispiel HTML:

```
<u>Robert Tolksdorf</u>
<address>
FU Berlin<br>
Netzbasierte
Informationssysteme<br>
Takustr.9<br>
D-14195 Berlin<br>
</address>
```

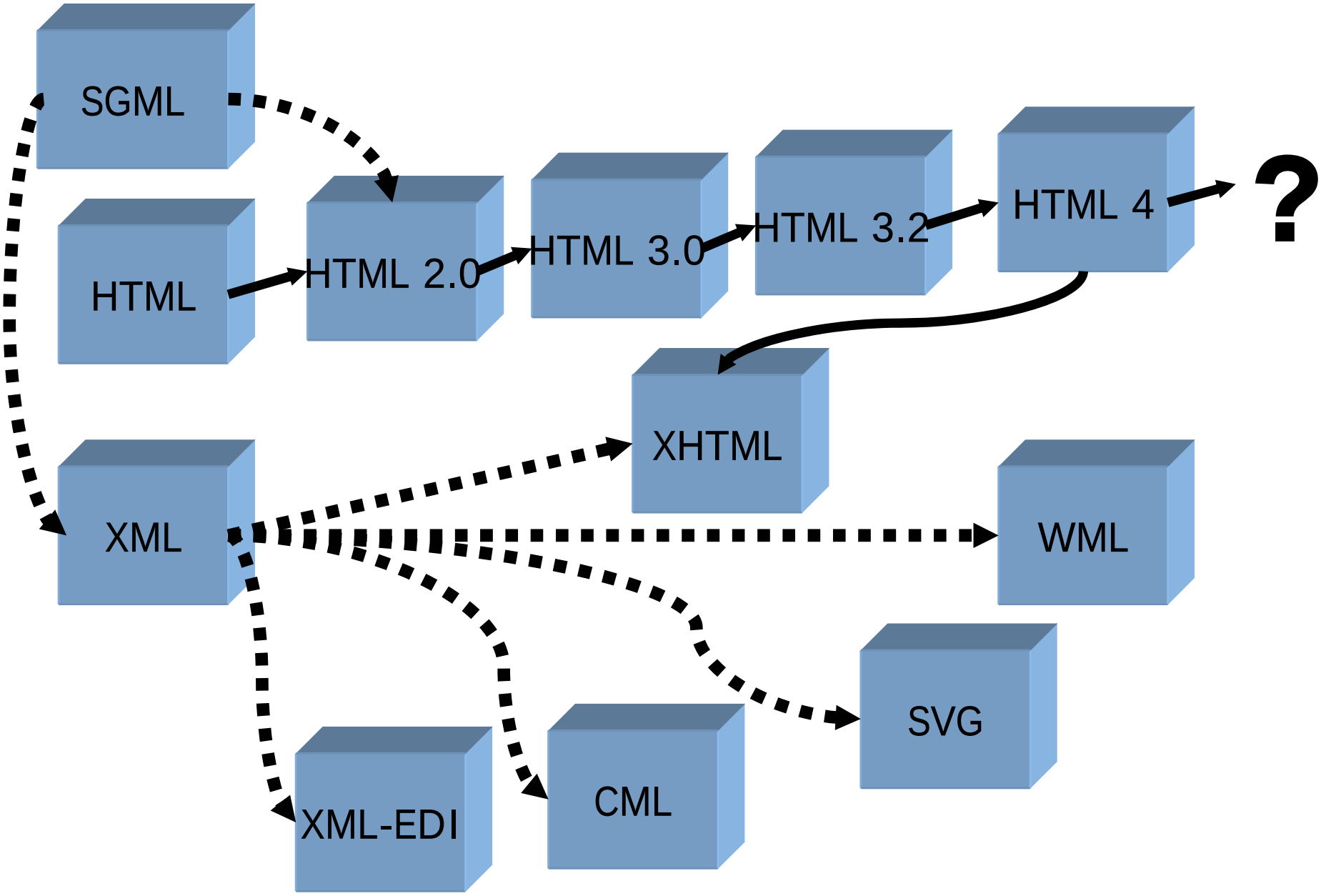


- *Tags* haben *logische* oder *visuelle* Bedeutung

Auszeichnungssprachen

- Kann es eine universelle Auszeichnungssprache geben?
  - Alle visuellen und sonstigen Möglichkeiten aller Ausgabegeräte müssten durch Tags steuerbar sein
  - Alle semantischen Konzepte aller Domänen müssten durch Tags repräsentierbar sein
  - Alle notwendigen Granularitäten der Auszeichnung müssten unterstützt werden:
    - `<ADRESSE>...</ADRESSE>`
    - `<ADRESSE><STRASSE>...</STRASSE><ORT>...</ORT></ADRESSE>`
    -
```
<ADRESSE>
  <STRASSE>...</STRASSE>
 <ORT><PLZ>...</PLZ><ORTSNAME>...</ORTSNAME></ORT>
</ADRESSE>
```
- Nein: Anwendungsspezifische Auszeichnung nötig

XML als Ergebnis der HTML Entwicklung



Anforderungen

- HTML hat Defizite
  - Nutzer können keine eigenen Tags oder Attribute definieren
    - Nutzer können so keine semantisch präzise Beschreibung ihrer Daten notieren
  - HTML verfügt nicht über alle notwendigen Strukturen zur Repräsentation von Datenbank- oder Objektschemata
    - Nutzer können vorhandene Daten nicht adäquat repräsentieren
  - HTML hat keine präzise Spezifikation (nur „best effort“)
    - Keine Überprüfbarkeit von markierten Daten
- Damit skaliert HTML und damit das „alte“ Web nicht mit sich wandelnden Anwendungsanforderungen

Anforderungen

- SGML hat Defizit:
  - Komplexität
- XML versucht die Defizite von HTML und SGML auszugleichen und so die Anforderungen Skalierbarkeit und Einfachheit zu erfüllen
  - Definition eigener Elemente und Attribute
  - Beliebig verschachtelbare Dokumentenstruktur
  - Durch Offenlegung der Grammatik Typisierung möglich
  - Erweiterte Link-Strukturen möglich
  - Style-Sheets Anbindung
- Siehe auch
  - Jon Bosak. XML, Java, and the Future of the Web. World Wide Web Journal. Volume 2, number 4, pages 219-227, 1997. http://citeseer.ist.psu.edu/cache/papers/cs/1263/http:zSzzSzwww.cis.udel.eduzSz~deckerzSzcourseszSz889czSzxmlapps.pdf/bosak97xml.pdf
  - Jon Bosak. The Birth of XML. Sun Developer Network. http://java.sun.com/xml/birth_of_xml.html

Document Type Definition

Freie Universität Berlin

- Eine XML-basierte Sprache wird durch eine XML-DTD (*Document Type Definition*) definiert

- Eine DTD enthält
  - Definitionen gültiger Elemente (Tags) der Sprache
  - Definitionen von Attributen der Elemente und deren Typen
  - Definitionen von Kürzeln (Entitäten)
  - Grammatikregeln

Beispiel: AdressenML

- Elementdefinitionen und Grammatikregeln:

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML Direktive *Elementname* *Grammatikregel*

```
<!ELEMENT ADRESSBUCH ANSCHRIFT*>
<!ELEMENT ANSCHRIFT (NAME,
          (STRASSE | POSTFACH)?, ORT)>
<!ELEMENT NAME ANY>
<!ELEMENT STRASSE ANY>
<!ELEMENT POSTFACH ANY>
<!ELEMENT ORT EMPTY>
```

- Beispiel:

```
<ADRESSBUCH><ANSCHRIFT><NAME>Robert Tolksdorf</NAME>
<STRASSE>Takustr. 9</STRASSE>
<ORT PLZ="14195" NAME="Berlin"/>
</ANSCHRIFT></ADRESSBUCH>
```

Beispiel: AdressenML

- Attributdefinitionen:
  <!ATTLIST ORT

Attributname *Typ* *Optional/Mandatorisch*

```
PLZ   CDATA #REQUIRED
NAME CDATA #REQUIRED
LAND CDATA #IMPLIED>
```

- Weiter möglich: Defaultwerte
- Mögliche Datentypen:
  - **CDATA**: Zeichenkette
  - **ID**: Eindeutiger Bezeichner
  - **IDREF**: Referenz auf **ID**
  - Selbstdefinierte Token
    <!ATTLIST PERSON GESCHLECHT (mann|frau) "frau">

Beispiel: AdressenML

- Kürzeldefinitionen:

Entitätsname *Expansionstext*

```
<!ENTITY InfStrasse "Takustr. 9" >
```

- Im Dokument:
  <STRASSE>&InfStrasse;</STRASSE>
- Zusätzlich auch DTD-weite Kürzel

Wohlgeformtheit und Validität

- *Wohlgeformtheit*:
  - Einhaltung der XML-Regeln, z.B.
    - Attributwerte in " eingeschlossen
    - Groß- und Kleinschreibung relevant
  - Alle Elemente sind geschlossen
    (Als Abkürzung: **<STRASSE></STRASSE>** = **<STRASSE/>**)
  - Korrekte Schachtelung von Elementen

- *Validität*:
  - Es gibt eine DTD zu einem XML-Dokument
  - Das XML-Dokument folgt den Regeln seiner DTD