Verteilte Web-basierte Systeme – SS 2006

Verteilte Web-basierte Systeme

Dr.-Ing. Martin Gaedke
Universität Karlsruhe (TH)

Verteilte Web-basierte Systeme – SS 2006

Part V

Planning

Part 5 – Overview

1. Conceptual Design: "Describing the problem"
   1. Content
   2. User-Interface Experience (UIX)
   3. Distributed Web System
2. Design: "Describing the solution"
3. Content
   1. Logical Design
   2. Physical Design
   3. DTD
   4. Schemas
   5. Example
4. User Interface Experience
   1. Presentation Design
   2. Navigation Design
   3. Dialogue Design
5. Web System
   1. Introduction
   2. Endpoint Design of Processes
   3. Endpoint Design of Services
   4. Components as Services
   5. Network Services
   6. XML Web Services
   7. Services in Context
   8. Web System Aspects
   9. Federated Systems
6. Further Readings

Verteilte Web-basierte Systeme – SS 2006

Chapter://1

Conceptual Design

Purpose

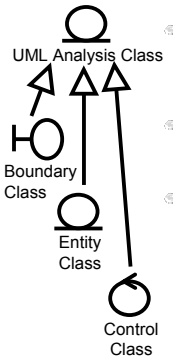
- Analysis of customer problem description
  - Describing the problem in **developer language**
  - Creating a **Conceptual Design**
- When to employ Conceptual Design?
  - Need for concurrent "Design & Implement" phases
  - Provides an excellent overview of the product
  - Easier to study than Design (~ 1:5)
  - Product uses Legacy Systems

* Analysis and Conceptual Design are used as similar terms/concepts

Conceptual Design Activities

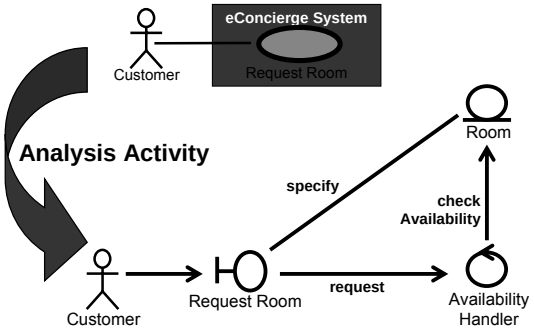
- **Activities to perform:**
  - **Transform requirements** into conceptual model model
  - Structure model elements wrt **product dimensions**
  - **Define validation tests**

Conceptual Model



- **«boundary»**
  Models Interaction between Product and Actors (e.g. User Interface or Communication Interface)
- **«entity»**
  Models information and associated behavior (e.g. real-life object, event)
- **«control»**
  Models coordination, sequencing, transactions, and control of other objects (e.g. Business Logic)
  - Do not encapsulate interactions with actors!

Transforming Example



Product Dimensions

- Product Dimensions
  - Separation of concerns of product
  - Each dimension subject for conceptual design
- **Aspects of Content**
  - Data - Entities and Relationships the product deals with
  - Semantic annotations/extensions etc. of Data that might be useful for other applications
- **Aspects of User-Interface Experience**
  - Presentation
  - Navigation
  - Dialogue
- **Aspects of Distributed Web Systems**
  - Processes
  - Communication
  - Web System Architecture

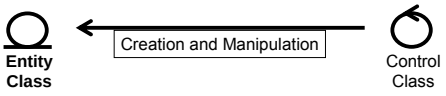
Verteilte Web-basierte Systeme – SS 2006

Section://1

Content

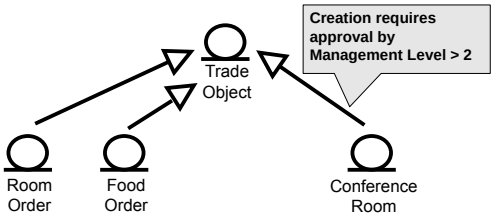
Data Design

- Define **Entity Classes**
  - Complete, semi-structured, etc.
  - Capture special requirements, e.g. persistency
- Identify relationships
  - Abstractions: refinements / generalization
  - *Roles, responsibilities, security concerns*
  - *Access rules: Creation and management*



Data Example

- Extract shared and common behavior
- Objective: Make solution easier to understand on a conceptual level



Data - Information and Media

- Some Observations and Aspects
  - static vs. dynamic
  - persistent vs. transient
  - passive vs. interactive
  - pull vs. push

These aspects must be considered in design.

Static vs. Dynamic

- Static
  - Access at different times returns the same result
- Dynamic
  - Access at different times may return different results
- Dynamic Information in the Web
  - Irregular changes
    - Extensions, updates, etc.
  - Regular Changes
    - Periodical issues (e.g. every month, week, day, hour)
    - Periodically generated information (e.g. every day, hour, min, sec)
  - Generated on demand
    - For each access
    - At certain events

Persistent vs. Transient

- Persistent
  - Data can be accessed time independent and repetitive (content is not changing)
    - E.g. books in the library, CD-ROM, DVD, ...
- Transient
  - “passing with time”
    - E.g. TV and Radio
- Transient Media on the Web
  - Web is only the transport media, e.g.
    - WebCam – snapshots sent over the web
    - Live media streams like Real Audio, Real Video
    - Results of a calculation (dynamic applications)

Passive vs. Interactive

- Passive
  - Unidirectional information from provider to consumer
  - No (very little) decision for the consumer
- Interactive
  - Information exchange in both direction from provider to consumer and vise versa
  - Symmetric, e.g. video-conferencing
  - Asymmetric, e.g. options to choose from, Wiki principle
- Interactive media on the Web
  - Hyperlinks
  - Forms
  - GUIs

Pull vs. Push

- Pull (Basic Web Mechanism)
  - Services by request
  - Communication is initiated by the user
- Push
  - Service by distribution
  - Communication is initiated by the provider
- Push-Media in the Web
  - Subscribing to information
  - Information distribution, e.g. ActiveChannels
  - Simulated – automated browser refresh
  - Server push
  - Streaming server (like radio and TV)

Verteilte Web-basierte Systeme – SS 2006

Section://2

User-Interface Experience (UIX)

Aspects of Interaction

- Human Computer Interaction (HCI)
  - User is interacting with the computer in order to accomplish something
    - (Dix et al., 1998)
  - ... It's not audio. It's not video. It's user control and dynamic experience
    - (T. R. Schussler, Mac World San Francisco conference, 1998)
  - Interactivity is about genuine human engagement.
    - (Nathan Shedroff's, 1994)
- In other words:
  - **If the experience you create is not a compelling one (whether justified by the bounds of the technology or not), you will never find a large audience.**
    - **(Shedroff, 1994)**
  - Teach interaction design like they do in film school
    - (Laurel, 1990)

What is Interaction Design?

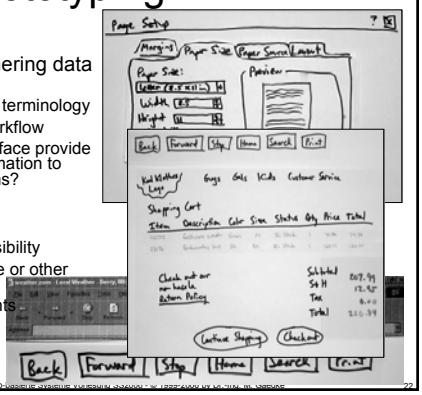
- Design of any direct or indirect communication between a user and computer?
  - **Direct interaction**: dialogue with feedback and control throughout the performance of a task
  - **Indirect interaction**: may involve background or batch processing
- It is about Creating experiences
  - The user should feel good

Model Presentation Aspects

- Presentation – Core component of interaction
  - Explanatory Interactive Assistance
  - User Agents aspects, e.g. screen-size
- Dynamics of content
  - Analyze entity and control classes
  - Identify static elements (e.g. temporary static)
  - Focus on: Access Frequency vs. Change Frequency
- Define layout and design elements
  - Analyze mock-ups and prototypes for reusable layout blocks
  - Define abstract, conceptual elements

Paper Prototyping

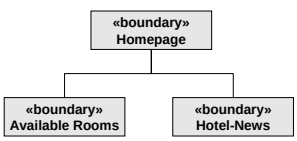
- Usability tool
- Useful for gathering data about:
  - Concepts and terminology
  - Navigation/workflow
  - Does the interface provide the right information to make decisions?
  - Page layout
- Not ideal for:
  - Technical feasibility
  - Download time or other response time
  - Colors and fonts
  - Scrolling



http://www-106.ibm.com/developerworks/library/us-paper/

Model Navigation

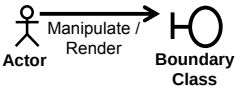
- Analyze Relations
  - Entity to entity relations based on RNA results
  - Set-based relations
- Possible Navigation Patterns to apply



- Types of Relations
  - Direct access to entity
  - Unidirectional / bidirectional
  - Access by type, e.g. to retrieve additional information about an entity
  - Transactional
  - Context-oriented relations
- A lot of research done here, e.g. OOHDM, UWE, OOH, WebML etc.

Model Dialogue

- Define roles for actors interacting with the information space
- Consider special requirements, e.g. browser or bandwidth, as additional dimensions
- Analyze major activities on entities
  - Context, e.g. events, authentication
  - Create, read, update, delete
  - View, Hide
  - Roles



Verteilte Web-basierte Systeme – SS 2006

Section://3

Distributed Web System

Introduction

- Conceptual Design of overall (distributed) application logic ~ "Business Process"
- **Business Process** – is defined by
  - Two or more autonomous Participants
  - Communicating via Operations
    - Message-based operations
    - Distributed environment
  - No central coordinating Authority
- Each **Participant**
  - Well-defined Behavior
  - Well-defined Relationships with other Participants
  - Concrete implementation is opaque (Black-Box Principle)

Model Business Process

- Design the overall (**usually distributed and large scale**) business logic of the Web product
- Analyze questions wrt remote execution
- Boundary class
  - Dedicated to system-based actors
  - May represent legacy system access
  - Prepares for wrapper/broker-pattern
  - Focus on Integration Aspects, like EAI, SOA
  - Security and protocol issues

Model Environmental Aspects

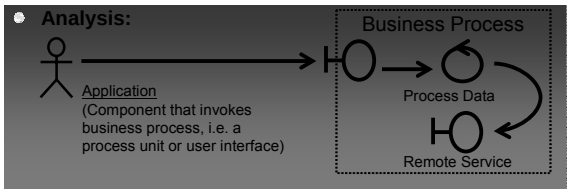
- Architecture is foundation of application
  - **Focus is on its distributed nature**
  - Evolution will happen soon
  - Product might fail due to environment restrictions
  - **Changes are very expensive!**
- Analyze
  - Dependencies between processes – Try to limit
  - Review non-functional requirements – e.g. Bandwidth
  - Optimizing
    - By reducing communication load
    - By increasing communication but reducing processing load
    - Check for caching possibilities
  - Check for security concerns

Idea

- Separation of Concerns
  - Specify Functionality
  - Design reusable process units
  - Package related business process logic (a.k.a. Domain Components)
  - E.g. Use Case related business objects
- Conceptual Design of overall application logic is separated into different views
  - Input for logical design: Process Design, which represents the process layer

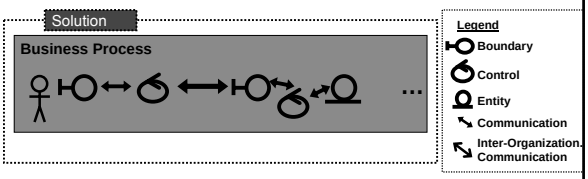
Shift To Process and Service Views

- **Analysis:**



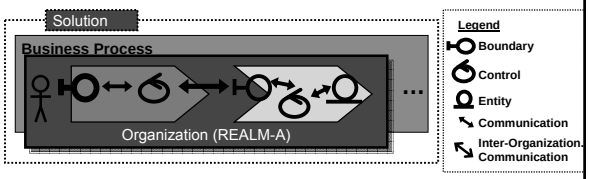
- Many questions to solve…
  - What is the application accessing? And how?
  - What is the interface becoming?
  - What is a processing unit? How is it connected with others?
  - Where is the System and what is it all about?
  - And many, many more…

Process at a Glance



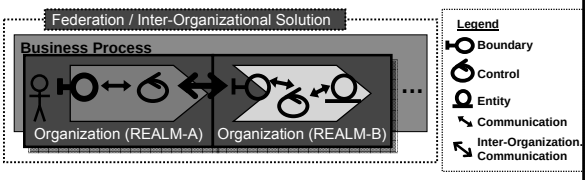
- Transform conceptual model into a more detailed business process notation
  - Helps to map conceptual model to logical models
  - Decompose Business Processes
- Complexity of Business Processes → Need for decomposition

Process Decomposition



- Business Process
  - Business process consists of multiple units
  - Units of a business process are owned by organization
  - Users are managed by organization
- Realm – Represents organizational boundaries
  - Zones of control over the owned systems, i.e. to support IAM, roles, firewall perimeter etc.

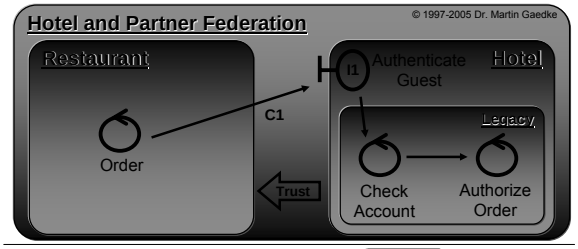
Process Federation



- Federated Business Processes
  - Business process is inter-organizational
  - Different Units of one business process are owned by different organizations
  - Requires dedicated support for identity and access management (User of realm-a acts in realm-b due to a trust-relationship between both organizations)

Example

- Example: Order – Discuss: Identity issues

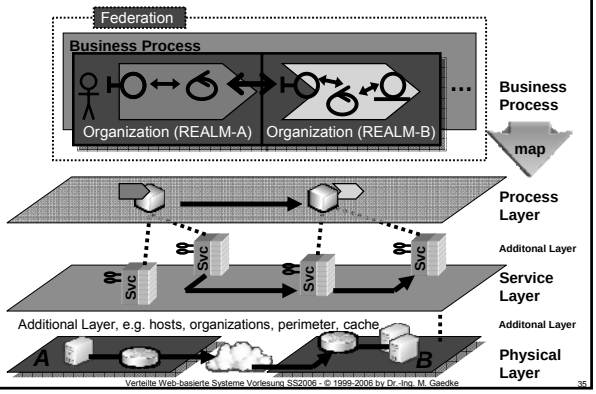


C1:
- Secured Connection
- Speed: 5-54 MBit/s

I1:
- CRUD-Interface
- Policy: Trusted Clients only

Zone Context
- Security Zone
- Trust Zone, etc.

Multi-Layer View on Business Process



Business Process Notation

- Allows to visualize communication between participants (possibly support for abstractions - business process units)
- Some approaches
  - Business Process Modeling Notation (BPMN)
  - UML Activity Diagrams
  - UML EDOC Business Processes
  - ebXML BPSS
  - Event-Process Chains (EPC)
  - State-Charts
  - Petri-Nets
  - etc.

Summary

- Business Process
  - Spawns conceptual design of the distributed system
- Distributed System is represented by different views (layers)
  - Common approach: Process Layer, Service Layer, and Physical Layer
  - Allows for layer of interest, e.g. Federation Layer for inter-organizational processes, hosting layer
  - Apply dedicated approaches (logical design and physical design) for each of these layers

Literature

- S. Ward and P. Kroll. *Building Web Solutions with the Rational Unified Process: Unifying the Creative Design Process and the Software Engineering Process*, Rational Software Whitepaper,1999.
- Chapter 3,4,8: I. Jacobson, G. Booch, J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 1999.
- S. Seely and K. Sharkey. *SOAP: Cross Platform Web Services Development Using XML*, Prentice Hall PTR, 2001.
- W. L. Oellermann Jr. *Architecting Web Services*, aPress, 2001.

==========

Further information available at **Lecture Web Site**

==========

Verteilte Web-basierte Systeme – SS 2006

Chapter://2

Design – Describing the Solution

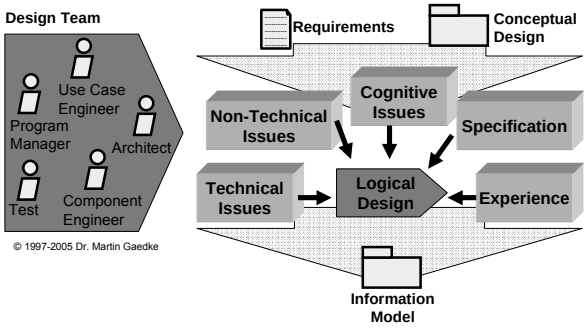
Introduction

- Shape the product and find its form – Regarding all requirements and conceptual design
  - E.g. data, presentation and hypermedia aspects
  - E.g. architecture and its distributed aspects
- Achieve Design Model of Web Application
  - Maintainable and detailed description
  - Easy to understand by common notation
- We focus on maintainable and reusable design aspects

Aspects of Design

- Design Aspects: Logical and Physical
- Separate logical units of the product
  - Complete solution independent of physical matter
- **Goal: Allow seamless mapping to physical design and implementation**
- Possibly implement parts of design in parallel

Design Team and Artifacts

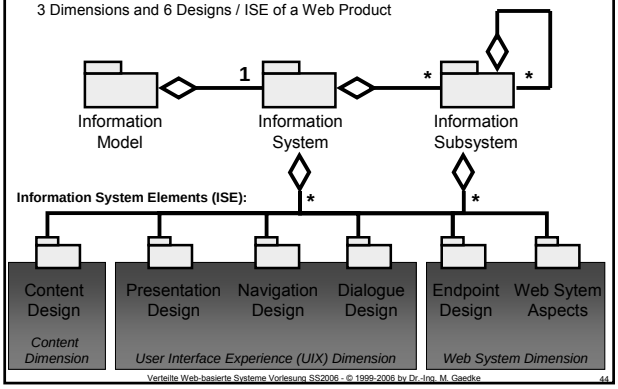


Information Model Approach

- Information Model is the first step in Design towards an overall Web Application Design
  - **Information Model** defines an Information Space
  - **Information Space** is a core concept of hypertext
  - Information Spaces can be part of larger Information Spaces
  - Information Space is part of the **Cyberspace**
- Purpose of the Information Model
  - Guiding development of logical and physical designs
  - Clarify the role, structure and experience of information to provide
  - Hypermedia idea: Associations between concepts to support location of information
- Support for managing process/progress
  - Without a model we can not manage information effectively
  - Documents transformations

Information Model

3 Dimensions and 6 Designs / ISE of a Web Product

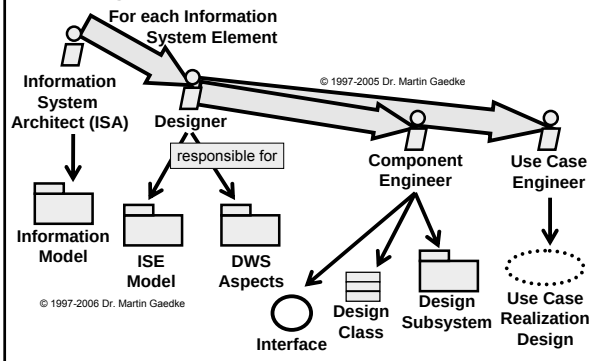


Information Model - II

- Activities regarding Information Model Design
  - Depending on major Product Dimensions
    - Example for Static Site:
      - Navigation,
      - Content,
      - Presentation Design

- Each information system element is closely related to each other
- There is no single thread / process model through the six designs

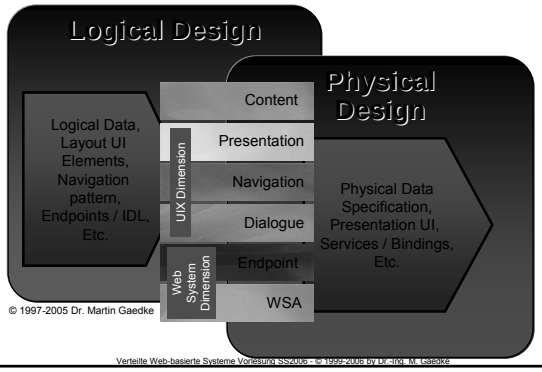
Design Team Responsibilities



Design Model Activities

- *Each* **Information System Element (ISE)** defined by the Information System Architect will be used as direct input for the Designer
- Designer responsible for mapping ISE
  - E.g. UML-Class design, applying navigation design pattern
  - Document DSA aspects, e.g. state (Statechart Diagram), information wrt scalability
- Develop Logical and Physical Design

From Logical to Physical Design



Verteilte Web-basierte Systeme – SS 2006

Chapter://3

Content

Goal

- Aspects of Content Design
- Designing the "**Information Space**"
  - Define **structure and structural links (references) of entities (data)**
  - Understand impact on **distribution**, i.e. **(physical memory) location of entities**
  - Understand and provide semantics
- Challenges
  - Develop maintainable structure
  - Focus on evolution (growth and changes)
  - Large amount of (multimedia) data

Verteilte Web-basierte Systeme – SS 2006

Section://1

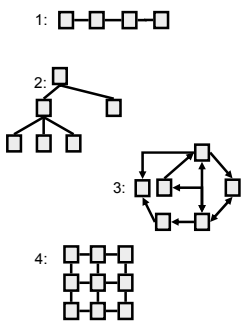
Logical Design

Logical Design Approaches

- Object-oriented Design approaches
- Hierarchical approaches
- *Web-compliant Data Definitions*
- *Diagramming Technique*
- Related approaches
  - Entity Relationship Modeling (ER-Design)
  - Relationship Management Methodology (RMM)
  - Object Role Modeling (ORM)
  - Object Z
  - Object Constraint Language (OCL)

Structure and Structural Links

- Linear Structure (1)
  - (ordered) Collections / Sets
  - Pre-caching possible
- Hierarchical Structure (2)
  - Deep or flat structured
  - Requires navigation aid
- Network Structure (3)
  - Extremely expressive
  - Complex Entities
- Grid Structure (4)
  - Collections of related Items
  - Requires uniform data



Verteilte Web-basierte Systeme – SS 2006

Section://2

Physical Design

Physical Design

- Transform logical design to physical model
  - Examples:
    - Class source code
    - Database specific table descriptions
- Dedicated approaches for representing data / information in the Web
  - **XML DTD**
  - **XML Schemas**
  - RelaxNG
  - Schematron
  - Resource Description Framework (RDF)
  - XML Information Set
  - Microformats (apply with care – if at all)
  - And many more…

Verteilte Web-basierte Systeme – SS 2006

Section://3

DTD

Web-compliant Data Definitions

- Extensible Markup Language (XML) 1.0 (Third Edition)
  - W3C Recommendation 04 February 2004
  - http://www.w3.org/TR/REC-xml/
  - "The function of the markup in an XML document is to describe its storage and logical structure and to associate attribute-value pairs with its logical structures. XML provides a mechanism, the document type declaration, to define constraints on the logical structure and to support the use of predefined storage units."

  - **Document Type Declaration** – Contains or points to markup declarations that provide a grammar for a class of documents. This grammar is known as a document type definition, or DTD.
  - **Document Type Definition** – Set of markup declarations included in or referenced by an XML Document.
- Design using DTD e.g. using Diagramming Technique

Document Type Declaration

- **Document Type Declaration Syntax**
  - **Document Type Declaration** ::= XMLDecl? Misc* (**doctypedecl** Misc*)?
  - **XMLDecl** ::= '<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'
  - **VersionInfo** ::= S 'version' Eq (""" VersionNum """ | "'" VersionNum "'")
  - **Eq** ::= S? '=' S?[26]
  - **VersionNum** ::= ([a-zA-Z0-9_.:] | '-')+
  - **Misc** ::= Comment | PI | S
  - S ::= White Space

Comments

<!-- a comment -->

- Contents are ignored by the XML processor
- Cannot come before the XML declaration
- Cannot appear inside an element tag
- May not include double hyphens

XML Declaration

<?xml ?>

- Not required, but typically used
- Attributes include:
  - version
  - encoding – the character encoding in the document
  - standalone – if yes no external DTD required

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

<?xml version="1.0" standalone="yes">

DOCTYPE

<!DOCTYPE ...>

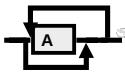
- Specify a DTD for the document
  - Refer to a DTD using a URI
  - Include a DTD inline as part of the document
- Example: Refer to a DTD

  <!DOCTYPE *order* SYSTEM "*http://a.b/order.dtd*">

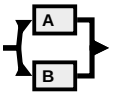
Document Type Definition

- **Document Type Definition Syntax**
  - Doctypedecl ::= '<!DOCTYPE' S Name (S ExternalID)? S? ('[' (Markupdecl | DeclSep)* ']' S?)? '>'
  - DeclSep ::= PEReference | S
  - Markupdecl ::= elementdecl | AttlistDecl | EntityDecl | NotationDecl | PI | Comment
- Types of Markup Declaration:
  - Element Type Declaration
  - Attribute-List Declaration
  - Entity Declaration
  - Notation Declaration

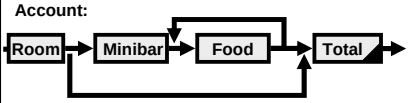
Diagramming Technique

	Description	Notation
	Component called A	(A)
	Component that can not be decomposed called A	(#PCDATA)
	A is optional or occurs once	A?
	A is repeated 1 to n times	A+
	A occurs 0 to n times	A*

Diagramming Technique 2

	Description	Notation
	Concatenation/Series B after A	(A,B)
	Selection A or B	(A|B)
	Example	***Example***

Account:



Account= (Room, (Minibar,Food+)?, Total)

Total= (#PCDATA)

Declaration, Definition, Data

Check the Lecture XML Validator Demo

```
<?xml version="1.0"?>
<!DOCTYPE account [
  <!ELEMENT account (room,(minibar,food+)?,total)>
  <!ATTLIST account AccountID ID #REQUIRED>
  <!ELEMENT room EMPTY>
  <!ATTLIST room number NMTOKEN #REQUIRED>
 <!ELEMENT minibar EMPTY>
  <!ELEMENT food EMPTY>
  <!ATTLIST food price CDATA #REQUIRED>
<!ELEMENT total (#PCDATA)>
]>
<account AccountID="a3499bxdz">
      <room number="R101"/>
      <minibar/>
      <food price="10.00"/><food price="15.00"/>
       <total>28.00</total>
</account>
```


Element Type Declaration

- Allows to define name of an element and its Content Model
  - <!ELEMENT S Name S Content-Specification>
- Name is the element type being declared
- Content-Specification:
  - **ANY** – Any use (assumed when no content model is provided)
  - **EMPTY** – No sub-elements allowed
  - **Mixed** or **Children s**pecification

Mixed Specification

- Mixed ::=
  '(' S? '#PCDATA' (S? '|' S? Name)* S? ')*'
  | '(' S? '#PCDATA' S? ')'
- Name must not appear more than once
- **Example**
  - **(#PCDATA)** – Only parsed Character Data allowed (= Text). Restricts all Child-Elements to be of Type Text.

Children Specification

- Each name is the type of an element which may appear as a child, as described in the grammar:
- Syntax:
  - Children ::= (choice | seq) ('?' | '*' | '+')?
  - cp ::= (Name | choice | seq) ('?' | '*' | '+')?
  - choice ::= '(' S? cp (S? '|' S? cp)+ S? ')'
  - seq ::= '(' S? cp (S? ',' S? cp)* S? ')'
- **Example**
  - **(room,total)** – Sequence of two elements of type room and total.

Attribute-List Declaration

- **Attributes** – Used to associate name-value pairs with elements.
- Attribute-List Declaration defines
  - Attributes bound to an Element
  - Type Constraints for these Attributes
  - Default Values for Attributes
- Syntax:
  - AttlistDecl ::= '<!ATTLIST' S Name AttDef* S? '>'
  - AttDef ::= S AttributeName S AttType S DefaultDecl

Attribute Types 1

- **String Type**
  - **CDATA** – Value is any literal string
- **Tokenized Types**
  - **ID** – Value must match name production and appear not more than once (only one ID per Element!)
  - **IDREF**, **IDREFS** – Value(s) must match ID attribute on some element in the document
  - **ENTITY**, **ENTITIES** – Value(s) must match name of unparsed entity
  - **NMTOKEN**, **NMTOKENS** – Values(s) must match NMToken production

Attribute Types 2

- **Enumerated Types**
  - **(v1|…|vn)** – Value is one of the values provided in the declaration
- Example:
  - <!ATTLIST elemname
    myenumtype (true|false|dontknow) 'true'>

Attribute Defaults

- **Attribute Default** – Defines whether an attribute's presence is required and if not how to deal with it
- Syntax:
  - DefaultDecl ::= '#REQUIRED' | '#IMPLIED' |
    (('#FIXED' S)? AttValue)
  - #Required – Attribute must be specified
  - #Implied – Attribute is optional
  - #Fixed – Required attribute; value is specified in quotes
  - AttValue – Contains the declared default value

Entity Declaration

- **Entities** – define storage units of an XML-document. They are either parsed or unparsed.
  - Allow for better maintenance
  - Parsed entity – content is text replacement
  - Unparsed entity – a resource whose content may or may not be text (text may be other than XML)
- Very powerful tool for advanced usage, cf. XML Specification for full details.

Entity Examples

- Common Entity Declarations
  - <!ENTITY lt "<">
  - <!ENTITY gt ">">
  - <!ENTITY amp "&">
  - <!ENTITY apos "'">
  - <!ENTITY quot """>
- Character and Entity Reference
  - Character: <
  - Entity (Declaration above): < >
- Parameter-Entity Reference for order.dtd:
  - Declaration: <!ENTITY % minibar.items "book | cdrom">
  - Usage: <!ELEMENT item (%shop.items;)+>
  - Means: item = (book | cdrom) +

CDATA

<![CDATA[ ]]>

- Allows to define special sections of character data, which the processor does not interpret as markup
- Anything inside is treated as plain text
- Example:
- <![CDATA[<ThisIsNoElement why="it is just data in a CDATA section"/>]]>

The Need For A Better DTD

- DTD in use for:
  - Sharing/Reuse many (!!) grammars
  - Validation by the parser
  - Defaulting of values
- Weaknesses of the concept:
  - DTD has a limited capability for specifying data types
  - DTD requires its own language
  - DTD provides incompatible set of data types with those found in databases
  - Example: DTD do not allow to specify element **day** and **month** of Type *Integer* and within a certain *Range*:
    <day>32</day><month>13</month>

Verteilte Web-basierte Systeme – SS 2006

Section://4

Schemas

XML Schemas

- XML Schema Definition Language (XSD)
  - http://www.w3.org/XML/Schema
  - XML Schemas provide a superset of the capabilities found in a DTD
- Motivation:
  - "While XML 1.0 supplies a mechanism, the Document Type Definition (DTD) for declaring constraints on the use of markup, automated processing of XML documents requires more rigorous and comprehensive facilities in this area. Requirements are for constraints on how the component parts of an application fit together, the document structure, attributes, data-typing, and so on."
- Notes:
  - W3C recommends "Schemas" as plural of schema
  - XDR (XML-Data Reduced) was an early attempt by Microsoft to define a Schema Language. XDR has been replaced by XSD

XML Schema Specification

- XML Schema Specification is partitioned into two parts
  - Part 1 specifies a language for defining composite types (called complex types) that describe the content model and attribute inventory of an XML element.
  - Part 2 specifies a set of built-in primitive types and a language for defining new primitive types (called simple types) in terms of existing types.
  - In addition to Parts 1 and 2, there is a primer to the XML Schema language known as Part 0 that provides an excellent overview of XML Schemas.
    - → Read Part 0 – This is Homework !!!

XML Schema in short…

- XML - Meta-language for defining markup
- Schema - Formal specification of grammar for a language (in XML!!!!)
  - As such it inherits all the good “stuff”, we know from XML
  - Useful for validation, interchange etc.
- XML Schema - Language for writing specifications

Solution: Namespaces

XML-Element written as <*nsname*:element>

- Help avoid element collision
  - P – Paragraph in HTML
  - P – Person in Address-Book DTD
- Namespace declaration
  - Using the xmlns:*nsname*=*value* attribute
  - URI is recommended for *value*
- Can be an attribute of any element; the scope is inside the element’s tags

Namespaces: Declaration

- Two schemas in context…
- Declaration scopes in root element
  - **<elem xmlns="uri1"** *xmlns:ns2="uri2"* ... **>**
    <*ns2:elem* **/>**
    </elem>
  - elem defines all namespaces
- Declaration after usage
  - **<ns1:elem xmlns:ns1="uri1">**
    <*ns2:elem xmlns:ns2="uri2"* **/>**
    </ns1:elem>

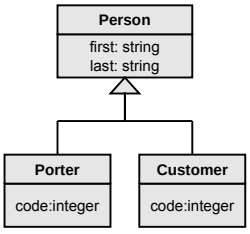
Example – Data Instances (I)

- **A porter**

```
<person>
    <first>Peter</first>
    <last>Huber</last>
    <code>4711</code>
</person>
```

- **A customer**

```
<person>
    <first>Harald</first>
    <last>Schmidt</last>
    <code>0815</code>
</person>
```



Example – Data Instances (II)

- **Porter (schemas applied)**

```
<P:person xmlns:P=“urn:person”>
    <P:first>Franz</P:first>
    <P:last>Huber</P:last>
  <E:porter xmlns:E=“urn:porter”>
      <E:code >4711</E:code>
  </E:porter>
</P:person>
```

- **Customer (schemas applied)**

```
<P:person xmlns:P=“urn:person”>
    <P:first>Harald</P:first>
    <P:last>Schmidt</P:last>
  <C:customer xmlns:C=“urn:customer”>
    <C:code >0815</C:code>
  </C:customer>
</P:person>
```

Example – Schema (I)

```
<xsd:schema id="person" targetNamespace="urn:person“
    xmlns="urn:person“
    xmlns:xsd="http://www.w3.org/2001/XMLSchema“
    attributeFormDefault="qualified“
    elementFormDefault="qualified">
  <xsd:element name="person">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:any namespace="urn:porter" />
        <xsd:any namespace="urn:customer" />
        <xsd:element name="first" type="xsd:string“ minOccurs="0" />
        <xsd:element name="last" type="xsd:string" minOccurs="0" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```


Example – Schema (II)

```
<xs:schema id="customer" targetNamespace="urn:customer“
xmlns="urn:customer“
xmlns:xs="http://www.w3.org/2001/XMLSchema“
attributeFormDefault="qualified" elementFormDefault="qualified">
    <xs:element name="customer">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="code" type="xs:int" minOccurs="0" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

- **Porter schema cf. Customer schema (Homework!)**

Schema Element

- Root Element of a Schema
  - <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- XML Schema Namespace
  - Each Element in the schema is prefixed by xsd: (xsd is only a convention)
  - Namespace declaration xmlns:xsd="http://www.w3.org/2001/XMLSchema“
  - The same prefix also appears on the names of built-in simple types, e.g. xsd:string.
- Most notably Subelements:
  - element
  - complexType
  - simpleType

Complex Type Definitions

- **Complex Types** – Allow child elements and may carry attributes
- Example: Element <USAddress> must consist of five Elements and one Attribute

```
<xsd:complexType name="USAddress" >
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="street" type="xsd:string"/>
    <xsd:element name="city" type="xsd:string"/>
    <xsd:element name="state" type="xsd:string"/>
    <xsd:element name="zip" type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:NMTOKEN"
  fixed="US"/>
</xsd:complexType>
```


Complex Type Definitions II

- Use of Complex Types - Example:

```
<xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
      <xsd:element ref="comment" minOccurs="0"/>   <xsd:element
  name="items" type="Items"/> </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
```

  - Use of shipTo and billTo Elements in XML requires that these Elements have the five Subelements as defined in USAddress
  - Ref Attribute (here its value is comment) indicates a Reference to elsewhere declared Element (global Element)
  - Comment Element here is optional due to Occurance Constraint

Complex Type Definitions III

- Element occurrence constraints
  - minOccurs
    - Required if value is 1
  - maxOccurs
    - Bound, e.g. value is 42
    - Unlimited if value is **unbound**
  - Default value for both attributes is 1
- Attributes occurrence constraints
  - Appear once or not at all
  - Constraints by use-attribute with value: required, optional or prohibited
  - Default value by default-attribute
- Example:
  - (minOccurs, maxOccurs) fixed, default = (0, 2) -, 37
    - Element may appear once, twice, or not at all
    - If the Element does not appear it is not provided; if it does appear and it is empty, its Value is 37; otherwise its value is that given

Simple Type Definitions

- **Simple Types** – Cannot have element content and cannot carry attributes. XML Schema has more than 40 built in Simple Types, e.g. string, integer, boolean, time, dateTime, date, gMonth, anyURI, language
- Defining new Simple Types is allowed
  - Derive and restrict existing simple type
  - Define by **simpleType** element
  - Use **restriction** sub-element to define **Facets** that constrain the range of values

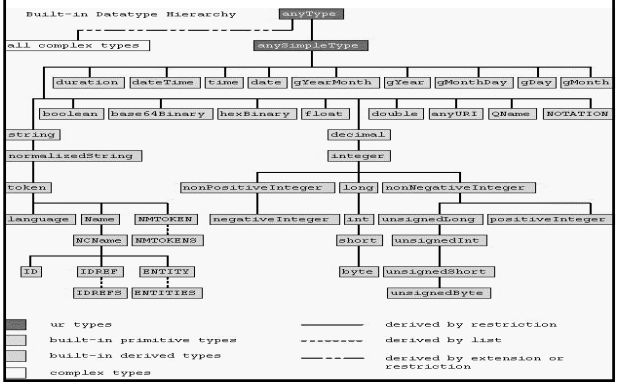
Simple Type Definitions II

- Example – Use of Facet called pattern

```
<xsd:simpleType name="SKU">
   <xsd:restriction base="xsd:string">
         <xsd:pattern value="\d{3}-[A-Z]{2}"/>
   </xsd:restriction>
</xsd:simpleType>
```

  - Derived from string value space
  - Facet: three Digits followed by a Hyphen followed by two upper-case ASCII Letters
- Other Facets available: Range, Enumeration, List, Union

Simple Type Definitions III



Element simpleContent

- Complex Types from Simple Types
  - **Example: <internationalPrice cur="EUR"> 423.46</internationalPrice>**
  -
```
<xsd:element name="internationalPrice">
 <xsd:complexType>
   <xsd:simpleContent>
     <xsd:extension base="xsd:decimal">
       <xsd:attribute name="cur" type="xsd:string"/>
     </xsd:extension>
   </xsd:simpleContent>
 </xsd:complexType>
</xsd:element>
```


Element complexContent II

- Mixed Content
  - **Example:**
    <hello>Dear <name>Martin Gaedke</name>.</hello>
  -
```
<xsd:element name="hello">
 <xsd:complexType mixed="true">
   <xsd:sequence>
     <xsd:element name="name" type="xsd:string"/>
   </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

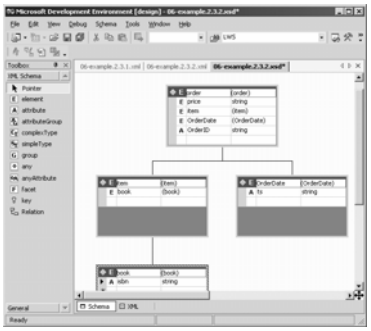

Power of XML Schema

- Defining Complex Types by group elements
  - E.g. sequence, choice, group, all
- Support for maintenance and evolution
  - Target Namespace
  - Schemas in multiple documents (include)
  - Deriving types by extension
  - Abstract elements and types (abstract="true")
  - Keys and references

Developing Schemas

- Use Tools:
E.g. XMLSpy, XMLAuthority, Visual Studio.NET

Many other exist



Verteilte Web-basierte Systeme – SS 2006

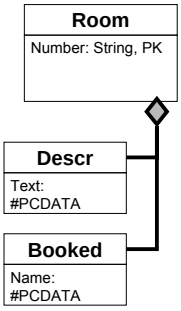
Section://5

Example

Example: Room Entity

- Pay attention to database design
- E.g. containment design does not scale…

```
<room number="R10">
    <descr>A very nice room</descr>
    <booked>Reseller Wonder Travel Corp.
    </booked>
</room>
<room number="R11">
    <descr>A very nice room</descr>
    <booked>Reseller Wonder  Travel Corp.
    </booked>
</room>
```



Example: Room Entity

- Structural Linking

```
<room number="R10">
    <descr href="Nice"/></room>
<room number="R11">
    <descr href="Nice"/></room>

<descr type="Nice">A very nice room</descr>

<booked roomref="R10" by="MrG"/>
<booked roomref="R11" by="MrG"/>
<Guest gid="MrG">
    Reseller Wonder Travel Corp. </Guest>
...
```


Example: Room Entity

- References
  - DTD or XML-Schema, e.g. **ID** and *IDREF*

```
<booked roomref="R10" by="MrG"/>
<Guest gid="MrG">...
```

- Shift to Physical Design
  - If XML many opportunities… for enhancements

Re*think*ing: The Room Entity

- Physical Design using XML
  - → Adding semantic support
  - E.g. enhance using Resource Description Framework (RDF)

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
   ns#" xmlns="http://hotel/rdf/syntax#">
   <room rdf:about="http://hotel/room/R10">
      <descr rdf:resource="http://hotel/descr/Nice"/>
   </room>
...
```

Verteilte Web-basierte Systeme – SS 2006

Chapter://4

User Interface Experience

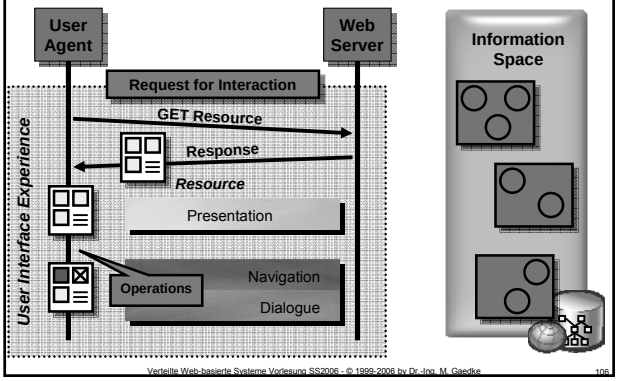
Goal

- Aspects of User Interface Experience (UIX) Design with Hypertext in mind
  - 1) Presentation
  - 2) Navigation
  - 3) Dialogue
- Focus on the interaction between human and solution

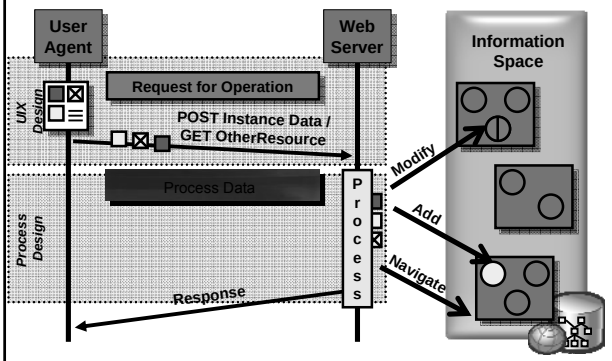
Interaction

- **Interaction** – Process involving a mode of operation in which there is a continual exchange of information between an actor and an Information Space
  - Operations (Actions) include exploring/browsing, modifying, and responding

Web Interaction Model



Web Interaction Model



Verteilte Web-basierte Systeme – SS 2006

Section://1

Presentation Design

Goal

- Design the final audio and visual presentation
  - Shape logical presentation's layout
  - Shape physical presentation's '*look and feel*'
- Challenges
  - Design maintainable, adaptable, and reusable presentation
  - Enable UI-Reuse → customization
  - Psychological problems
  - Cultural Aspects
  - Create an experience!

Sadness

Sadness

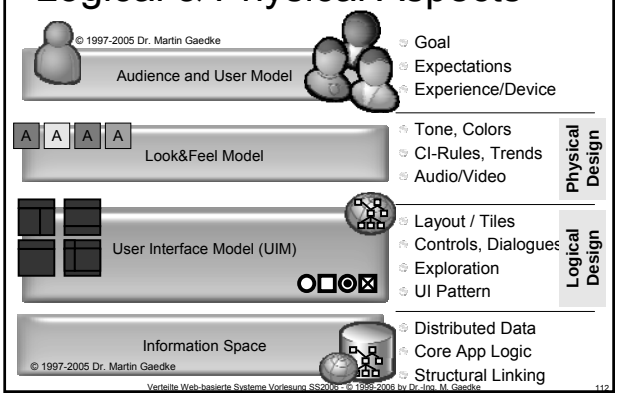
Typical Problems

- Problem – Accepting importance of design
  - Technical oriented people in the team often attack visual elements as wasteful
- Social Aspects
  - Presentation design must add to the site in a meaningful way
  - Transfer a message to the "viewer", e.g. emotions
- Web Engineers are not Graphics Designers, but
  - Bad presentation may make a perfectly engineered solution unusable
  - Badly engineered solution may not become usable by good presentation (but will look good ;-)

Some Facts to keep in Mind

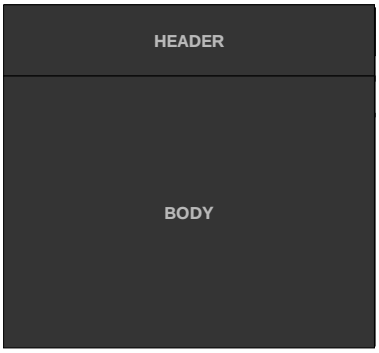
- No bullet-proof rules for Presentation Design exist
- Marketing department will provide some guidelines, e.g. Corporate Identity (CI) Styles
- Corporate Web Sites must promote and reflect the company's brand
- Intranet/Extranet Sites usually pay less Attention about Presentation Design

Logical & Physical Aspects



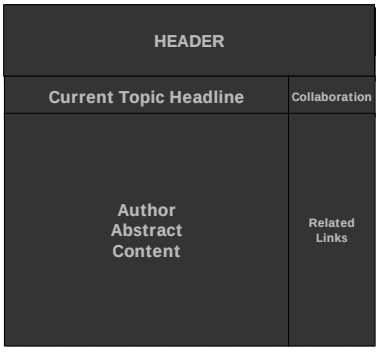
Tiles

UIM



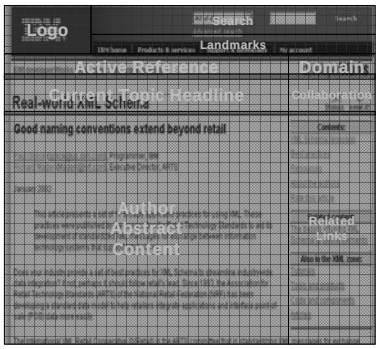
Tiles

UIM



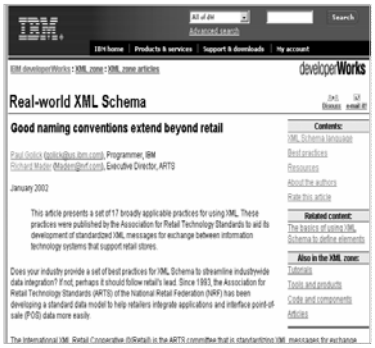
Tiles

UIM



Tiles

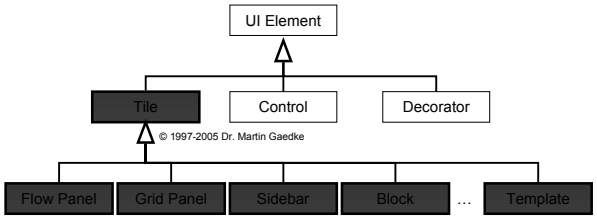
UIM



Core Tiles

UIM

- Tiles – Structure and arrange UI Elements



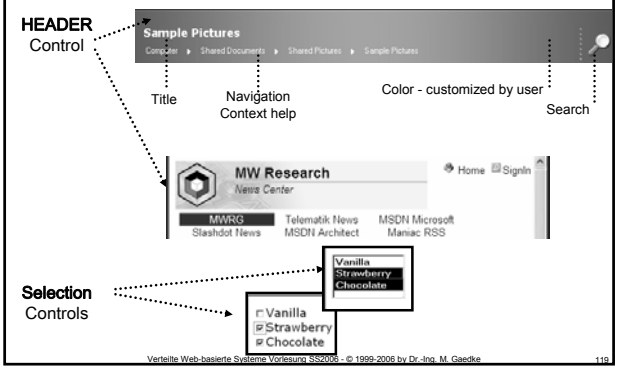
Controls

UIM

- Tiles are filled with controls
  - Tiles provide a structure for rendering controls
- **Control** – Describes a way of accessing instance data of the information space with an interaction handler
  - Abstracts - UI Element treated independent of its presentation
  - Separates presentation of content and its interaction
  - Treats Data as a first class citizen
  - Takes a uniform approach to presentation experience
    - I.e. captions, help text, tabbing and keyboard shortcuts

Control Examples

UIM

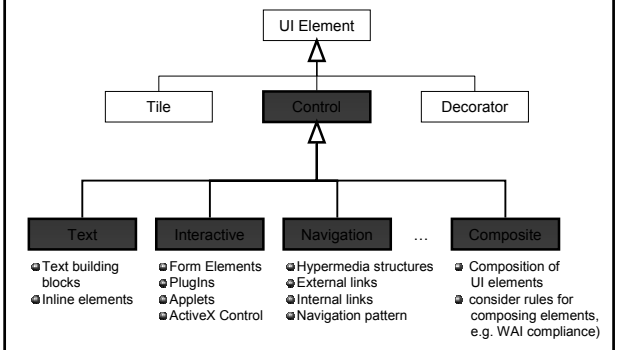


Control Aspects

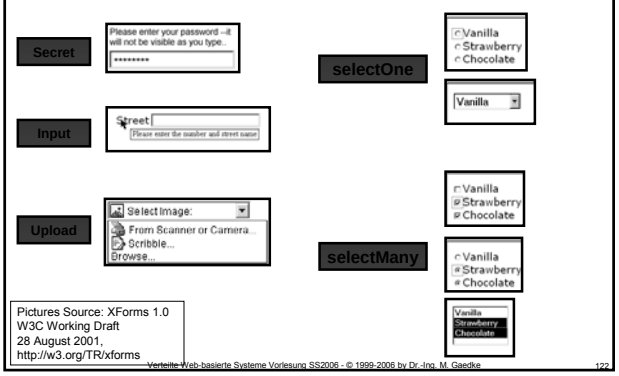
UIM

- Controls are UI Components
  - Allow for early testing, e.g. loading times
  - I.e. bandwidth or kilobytes per control
  - I.e. bandwidth per page = Sum of all Tiles and containing controls
- **Control Model** – Defines a set of abstract User Interface controls
  - Allows to manage abstract controls and their mapping to physical representatives
  - Help for maintenance and reuse
  - Improve consistency of the overall site
  - Object-oriented concepts may be applied
  - Examples
    - Define all icons and their meaning in a consistent way
    - Define text, size, colors, etc.

Simplified Control Model

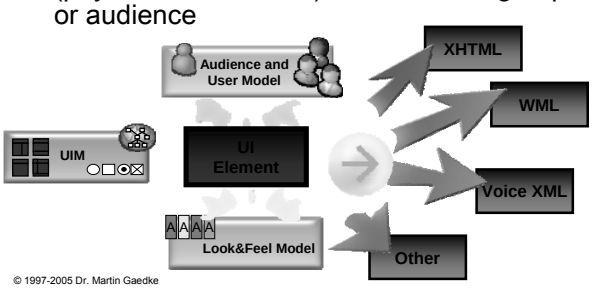


Interactive Controls



Shift to Physical Design

- Develop UI Model **representatives** (physical UI elements) for each user group or audience



© 1997-2005 Dr. Martin Gaedke

Physical Design Examples

- Map and decorate/style logical elements … based on audience / culture

Verteilte Web-basierte Systeme – SS 2006

Section://2

Navigation Design

Goal

- Improve ease of use and access to information and processes
  - Focus user's attention — unnecessary information are out of the picture
  - Design link structure for "understandable" navigation
  - Develop understanding of semantic associations
  - In other words: *Focus on Hypermedia idea in general*
- Challenges
  - Improve "flow" of the user experience
  - Creating meaningful (Navigation) structure – without cognitive overload
  - Build secure connections between features
  - Reuse of navigational pattern

Hypermedia Linking

- **Link** – Association between a Source Node and one or many Targets Node
- Taxonomy based on type of information
  - Structural links
  - Associative links
  - Referential links
  - Contextual links
- Note: WWW does not provide mechanisms for differentiating link types

Associative Links

- Define semantic relationship between information elements
  - "Standard" Hypermedia Idea
  - E.g. for more information on X refer to Y
  - Independent of the underlying content structure (This is not structural linking!)
- Example:

Referential Links

- Provides Link between item and an explanation
- Item at end of Referential Link exists because of the existence of the other item
- Example – Fragment of Room description
  - The business rooms are equipped with WLAN and On-Demand video…
  - **Definition**: Wireless network adapters connect individual …refer to this as a **Wireless Local Area Network** (**WLAN**).

Contextual Links

- **Context** - "the set of facts or circumstances that surround a situation or event"
  - Synonyms: situation, phase, position, place, point, standing, status, occasion, environment, location, dependence ...
- Provides relation between items based on context
  - Often between items in Information Space and real world (cf. Ubiquitous Computing)

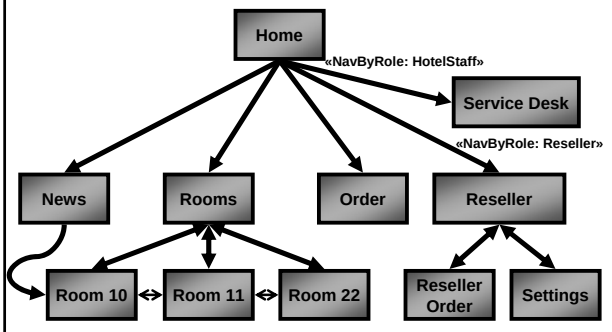
Contextual Links

- Web Applications for Mobile Devices
  - Becomes more important as technology for mobile devices evolves
  - Referring to information *surrounding* the system
  - E.g. Where a computer is used, by whom, who else is around, what are they doing, ...
- Example
  - Mobile Devices with Location Information
    - Project at the University of Lancaster, UK
    - Project NUKATH, University of Karlsruhe, Germany

Navigation Design Process

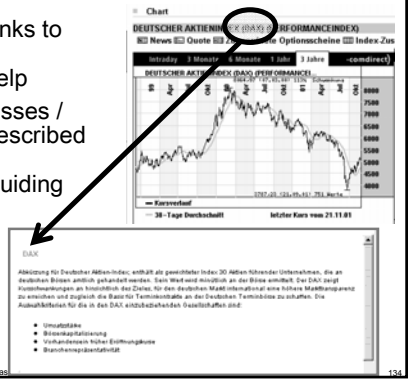
- Navigation Design involves many different types of activities
  - Logical Design connect Business Process (Site Map approach)
  - Domain specific navigation design – enhancing information access and task oriented use of processes
  - Design Navigational context with external resources
  - Enhancing user experience with Navigational Design Pattern
- Monitor and support:
  - Access by Navigation
  - Access by Direct Addressing
  - Access by Search Request
  - Access by Browsing

Site Map



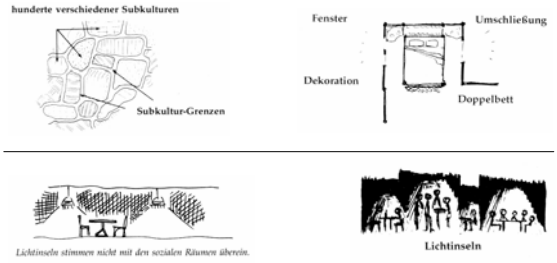
Domain-specific Navigation

- Referential Links to access information/help
- Access processes / features as described in Use Case scenarios – guiding user principle



Patterns: Reusing Experience

- Source: C. Alexander et al.: *A Pattern Language*, Oxford University Press, New York, USA, 1977



Pattern

- Architect C. Alexander
  - Collect structures of urban development and other architectural artifacts of any granularity
- Book *A Pattern Language* 253 patterns
  - Could be used to build rooms, buildings, places, cities…
  - Over ten years work experience
  - Each pattern is presented in the same format

About a Pattern

- Alexander says:
  - Each pattern should be expressed as a solution to a conflict between forces
  - Empirical evidence is needed to show that the pattern works as claimed
  - Patterns should make us feel good
- A Pattern tells us how to go from a bad solution to a good solution

Patterns Applied

- A **Design Pattern** – "describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution."
- Further exists
  - Analysis patterns
  - Architectural patterns
  - etc.

Describing a Pattern

- **A Description Approach**
  - **Name** – Short and indicates what to be solved
  - **Problem** – Description of conflicts within a given environment
  - **Discussion** – Explains the problem in more detail and describes empirical evidence of existence
  - **Solution** – Instructions for solving the problem
- Other Descriptions are in use
- Tip: It is useful to apply this kind of approach for remembering personal experience

Examples

- Examples discussed here are only a portion of patterns available today
  - Note: Pattern description here is not complete
- Valuable sources to visit:
  - **Hypermedia Design Patterns Repository**
    http://www.designpattern.lu.unisi.ch/
  - **Patterns Home Page**
    http://hillside.net/patterns/
  - **An HTML 2.0 Pattern Language**
    http://www.anamorph.com/docs/patterns/

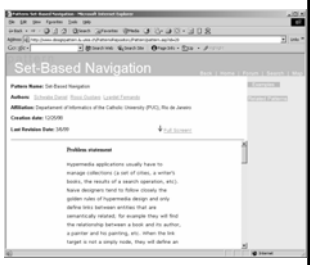
Hypermedia DP-Repository

- **Pattern Overview**
  - **Active Reference**
  - Advising
  - Analyze Organize Synthesize
  - Behavioral Grouping
  - Behavior Anticipation
  - Collection Center
  - Complex Entity
  - Guided Tour
  - Here I am
  - Hybrid Collection
  - Index Navigation
  - Info-Interaction Decoupling
  - Information Factoring
  - Information on Demand
  - Information-Interaction Coupling
  - **Landmark***
  - Navigation Strategy
  - Navigational Context
  - **News**
  - Node as a Navigational View
  - Opportunistic Linking
  - Process Feed-Back
  - Selectable Keywords
  - Selectable Search Engine
  - Selectable Search Space
  - **Set-Based Navigation**
  - **Shopping Process***
  - Simple Search Interface
  - Structured Answer

*** - not part of HDPR**

Pattern: Set-Based Navigation

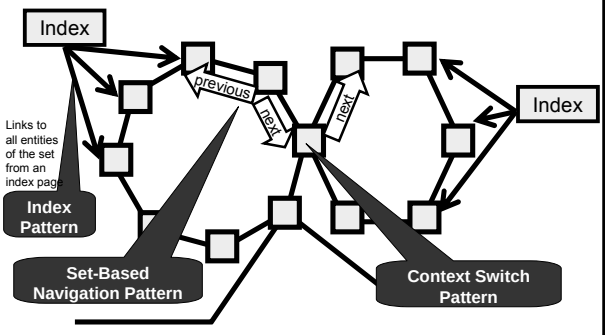
- **Name:**
  - Set-Based Navigation
- **Problem:**
  - Naive Designers tend to follow closely the golden Rules of Hypermedia Design and only define Links between Entities that are semantically related…
  - **The user will have to move from the index to a target**



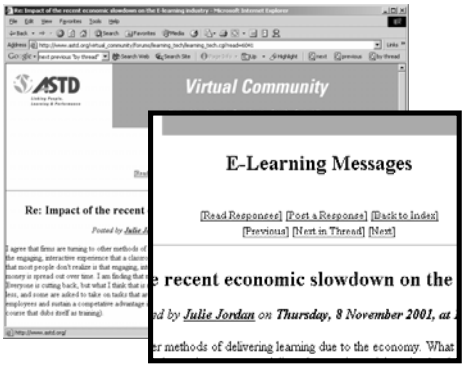
Pattern: Set-Based Navigation

- Solution:
  - **Consider sets as first class entities**
  - Provide intra-set navigation controls e.g. "next" and "previous“
  - Combine Set-based Navigation with proper indices to make exploration easier
  - Node may appear in two different sets: use the Nodes in Context pattern

Pattern: Set-Based Navigation



Pattern: Set-Based Navigation

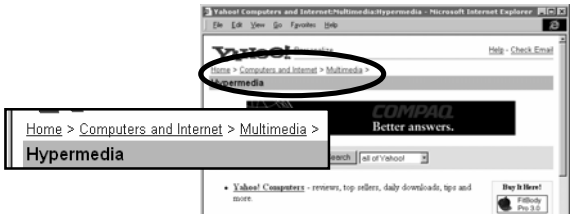


Pattern: Active Reference

- **Name:**
  - Active Reference
- **Problem:**
  - We need a way to help the user understand where she/he is and where to go next. Indexes or other access structures provide only partial solutions

Pattern: Active Reference

- **Solution:**
  - Maintain an active and perceivable navigational object acting as an index to other nodes



Pattern: Landmark

- **Name:**
  - Landmark
- **Problem:**
  - How to give easy access to different unrelated sub-systems? Web Applications usually contain many interesting entry-points; links to those points do not reflect conceptual relationships, and those links may yield a spaghetti-like structure

Pattern: Landmark

- **Solution:**
  - Define a set of Landmarks and make them accessible from every node in the application

Pattern: Shopping Process

- **Name:**
  - Shopping Process
  - Also known as: Shopping Basket / Cart
- **Problem:**
  - The Shopping Process must have well defined steps. This is necessary because we need to show the customer where she/he is in the process. The problem is now: How to describe the Shopping Process in a precise way? And how to present the customer a summary of her/his **navigational decisions**?

Pattern: Shopping Process

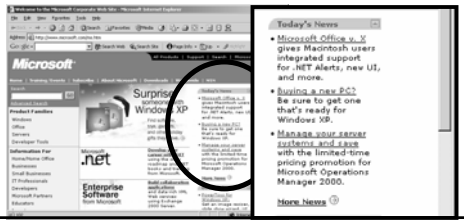
- **Solution:**
  - Provide a well-known metaphor to improve navigation
  - Provide a way for selecting items, managing items and check out (complete process)

Pattern: News

- **Name:**
  - News
- **Problem:**
  - How to tell users of dynamic Web-Sites that there is new information? Most large Web-Sites are tree-structured. These Information Spaces tend to be large, and are hardly ever completely navigated. New information may not be found.

Pattern: News

- **Solution:**
  - Structure the Home Page in such a way that space is devoted to *newest a*dditions. Make those headlines anchors to the new pages
  - Additionally provide data as RSS



Monitor and Support Access

What if users still do not find ...

- (1) Access by Navigation
  - Use common concepts
  - Solution
    - Navigational Design Pattern
- (2) Access by Direct Addressing
  - Requires unique addresses
  - Solution
    - Use guessable (common) names / URL
    - Provide URL via marketing channels, e.g. TV

Monitor and Support Access

- (3) Access by Search Request
  - Provide search facilities
  - Solution
    - Reuse external search engine or implement internal
- (4) Access by Browsing
  - Provide support if class of information is unknown
  - Solution
    - Like “browsing” in a library: Register with catalog systems
    - Examples: lycos.com, web.de, ebay.de
    - Requires good classification, keywords, ontology expert

Verteilte Web-basierte Systeme – SS 2006

Section://3

Dialogue Design

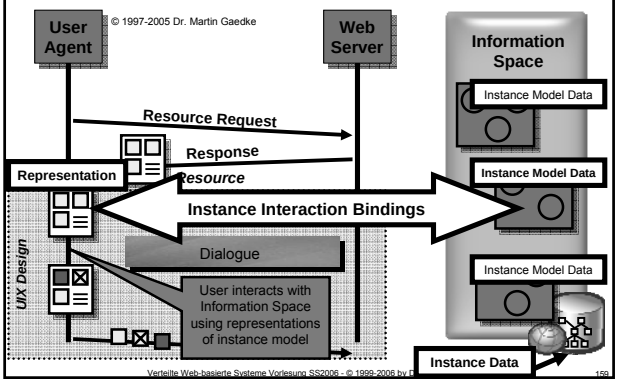
Introduction

- Purpose of Dialogue Design
  - Logical Design of how a user or system may interact with the Information Space of a Web Application
  - Physical Design representing the interaction logic
- Challenges
  - Collect relevant information about interaction from Use Cases, e.g. access based on roles or devices
  - Search for specific interaction items not covered by conceptual design
  - Separate interaction and layout concerns
  - Provide a good experience when interacting with the information space, e.g. similar to desktop experience

Instance Model

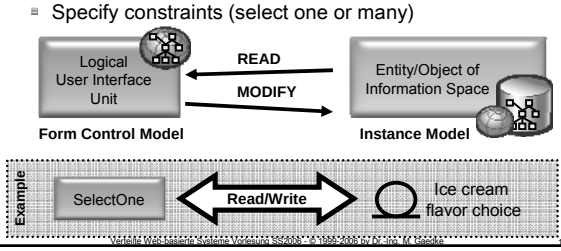
- **Instance Model** – Internal representation of values and state of all the instance data associated with a particular form
  - Usually same representation as used in Information Space – but may be different!
- Internal representation
  - Defines data type and form-specific constraints on a single piece of collected data

Interaction Model in the Web

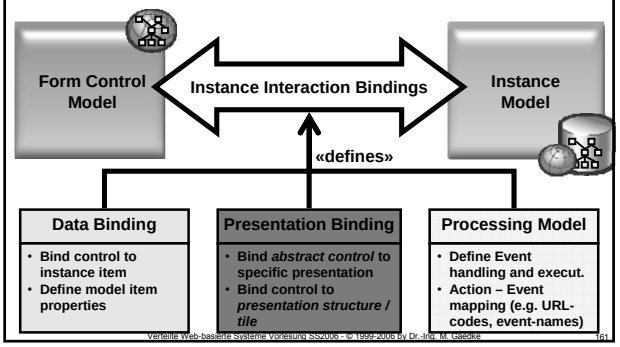


Logical Design

- Interactive user interface item
  - Many Representations – one Semantic
  - Specify security
  - Specify access patterns (Read vs. Write etc.)
  - Specify constraints (select one or many)



Shift to Physical Design



Data Binding Model

- **Data Binding Model** – Set of data bindings. Describes the glue that connects the separate items of the Form Control Model with items of the Instance Model.
- **Data Binding** – Triple defining the connection between a *Form Control* and an *Instance Data Value* and *Item Properties*.

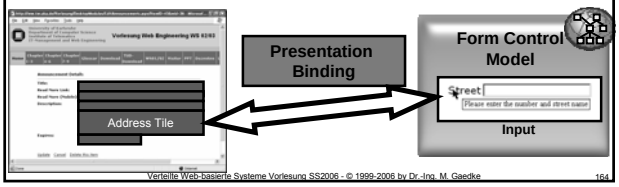
Item Properties

- Describing the logical relationship between Form Control and Instance, e.g.
  - **readOnly** – value is restricted from changing
  - **required** – value is required before the instance data is submitted
  - **relevant** – value is *currently* relevant to the rest of the Model
  - **maxOccurs** – applies only to repeat elements
  - **minOccurs** – applies only to repeat elements

  etc.

Presentation Binding Model

- Defines **Control Presentation**
  - E.g. SelectOne → Radio Buttons
  - Usually done in Presentation Design
- Defines "**Where**" in the layout structure, e.g. binding to tiles



Processing Model

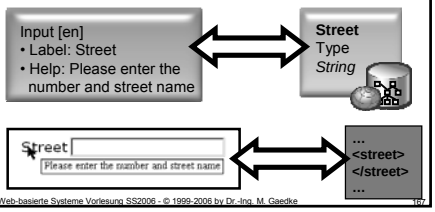
- **Processing Model** – Defines Event Processing at appropriate points within the User Interface
  - All form controls have a set of common *behaviors* that encourage consistent authoring and look and feel
  - Provides flexible means in conjunction with the Binding Mechanisms
  - E.g. send form data to server after pressing submit button *or* after every change

Processing Model: Examples

- HTML 2.0 Forms Model
  - Instance transfer: Submit
  - Events on instance data: Reset
- Scripting Approach (Advanced technologies)
  - Common approach today (but difficult to maintain, often hand-coded)
  - Use of scripting and standard form-controls
  - Instance transfer: Post initiated by script-code
  - Supporting technologies, like AJAX (Asynchronous JavaScript and XML), Backbase, Rico, DWR
- XForms Action Model
  - Classifies four behavior groups
  - Instance Transfer
  - Actions on Instance Data
  - Actions on Form Controls
  - Actions on Dependencies
- Trend: browser-based support of interface markup languages
  - Mozilla XUL (XML User Interface Language), Microsoft XAML (Extensible Application Markup Language), etc.

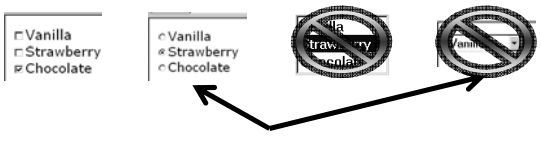
Supporting the User

- Goal: Optimize use of application
- Design Tasks
  - Minimize time user has to interact
  - Reduce elements prone to wrong inputs
  - Provide information to support use of input element
- Example:



Selecting One Flavor

- Applying - Instance Interaction Binding
  - Example: Selecting one or only one?
  - Example: What if the User Agent does not support select boxes?



Implies Select Only One

Verteilte Web-basierte Systeme – SS 2006

Chapter://5

Web System

Goal

- Aspects of a Distributed Web System
  - Using and understanding the "Environment" to host business processes and applications
  - Define/Reuse business processes and their elements
  - Define *global* (including multiple organizations) wiring of the overall elements
  - Define communication and connectivity aspects, like transport protocols, security, topologies, behavior, constraints and external relationships between all elements
- Challenges
  - Complexity of ultra large-scale distributed systems
  - Maintenance and evolution of all "elements" of the application, its business processes and environment

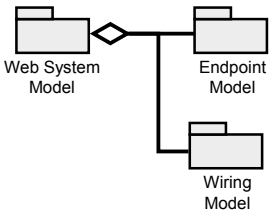
Verteilte Web-basierte Systeme – SS 2006

Section://1

Introduction

Web System

- **Web System Model** – A model that defines the overall product logic as a set of **Endpoints** and their **Wiring** (connections).
  - **Endpoint Model**
    - Endpoint interface and message definitions
    - Possible Service Level Access Requirements
  - **Wiring Model**
    - Defines wiring between endpoints
    - Depends on communication aspects



Introduction to Endpoints

- **Endpoint** (or service access point) – A specific location for accessing a functional unit operating on messages
  - Accessing an Endpoint is defined by specific protocols and data formats
  - Endpoints are provided by a Service, e.g. a component, XML Web Service, Web/Email server
- **Endpoint Definition** – A definition describing the necessary information for accessing or providing a dedicated type of an Endpoint
  - Abstract Endpoint Definition (logical aspect)
  - Concrete Endpoint Definition (physical aspect)

Wiring Model

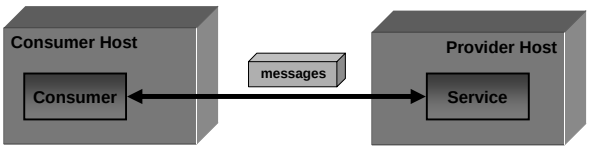
- **Wiring Model (Glue Model, Endpoint Instance Configuration)** – A Model that describes *which* Endpoints are connected and *how* they work together by using **Wire Protocols**.
  - This is more than just define bindings
  - Gluing functionality, requires solid observation and definition of factors influencing behavior
  - Allows for highly dynamic wiring using brokering approaches, e.g. Peer-to-Peer (P2P) networks or broker-proxies

Wire Protocol

- **Wire Protocol** – Defines the format of messages and how specific messages are exchanged between endpoints (respectively service consumer and endpoint).
- This Definition supports distributed systems, i.e. Wire Protocol allows for
  - Message exchange between physical nodes
  - Message patterns using multiple communication protocols
- Examples
  - Wire Protocol put into concrete form by Component Run-Time System DCOM/COM+: Object-RPC, Java: RMI, Internet: Application Protocols / Web Service Bindings etc.

Wiring Distributed Endpoints

- **Distributed Endpoints** – Endpoints are deployed to remote Hosts
- **Deployment** – describes distribution of processes and endpoints to processing unit (Hosting node, Host)



Wiring Models Examples

- Wiring Models (Orchestration Models)
  - Business Process Languages
  - XLANG – scheduling language for orchestration of components
  - Web Service Modeling Ontology (WSMO) and related approaches e.g. WSMX (Web Service Modeling eXecution environment)
- Other approaches that might be applied / useful
  - **State-Chart, Petri-Nets, etc.**
    - Petri-Nets are well understood and allow for programmatic optimization
  - **Business-driven Models**
    - ebXML – eBusiness XML by UN/CEFACT and OASIS
    - Cf. http://www.ebxml.org
  - **Architecture Description Models**
    - Support for transactional processes – supports any action, which allows to reverse the step (Undo-Semantic)
  - **Highly Dynamic Approaches**
    - Require modeling Contract-Negotiation
    - E.g. using Broker or Peer to Peer Networks (P2P network)

Factors Influencing Wiring

- Web System Aspects
  - Contracts
  - Dynamic Processes
  - Time-Behavior
  - Payload
  - Security
  - Scalability
  - Optimization
  - Specific Factors (rules) may exist, e.g. law related aspects defined in business rules
  - etc.

Verteilte Web-basierte Systeme – SS 2006

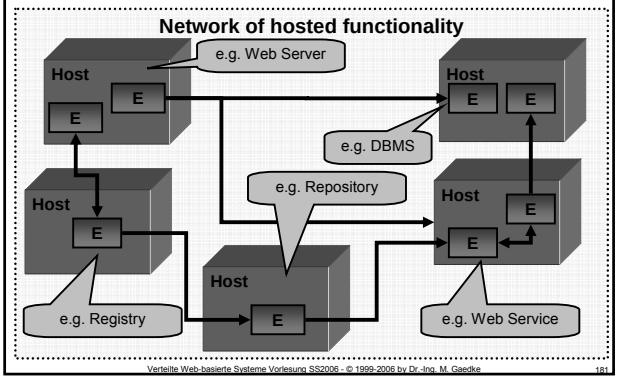
Section://2

Endpoint Design of Processes

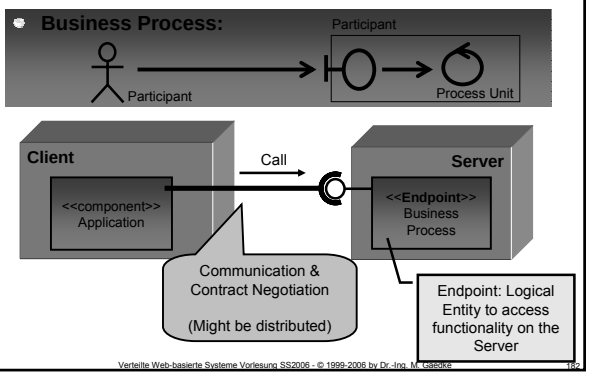
Introduction

- Design of the Process Layer
  - Logical Design derived from Business Process Model
  - Supports translating into physical model, which might be executable
- Challenges
  - Separation of concerns (functional units and flow between them)
  - Distributed systems hosting the business process
- Advantages
  - Eases considering complex **business rules**
  - E.g. customer related aspects, assurances, such as transactions, reliability, durability, by focusing on **wiring aspects** and **participant behavior**

Distributed Business Process



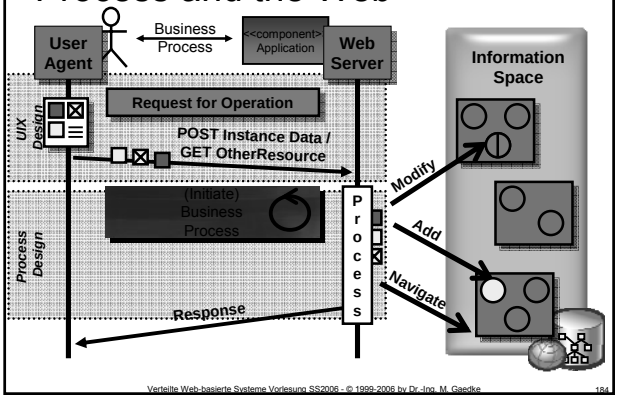
Business Process Deployed



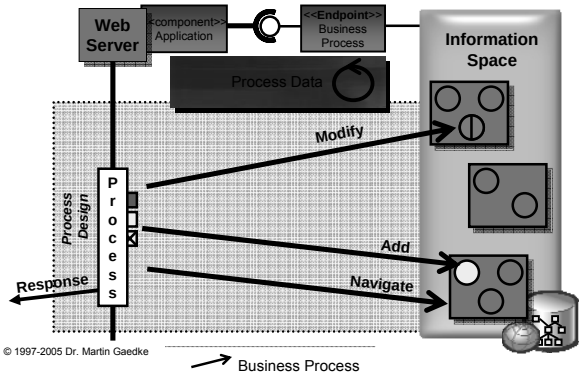
Process Unit

- **Process Unit (Process Component)**
  - Functional representation of a participant of the business process
  - **A unit accessible as Endpoint within the process layer** (and usually provided by several endpoints of the service layer)
- Modeling focuses on **wiring of process units (respectively their endpoints)**
  - Relationships between endpoints are expressed in terms of agreed upon communication patterns
  - Behavior of endpoints is described in abstract terms (Input-/Output-Contracts)

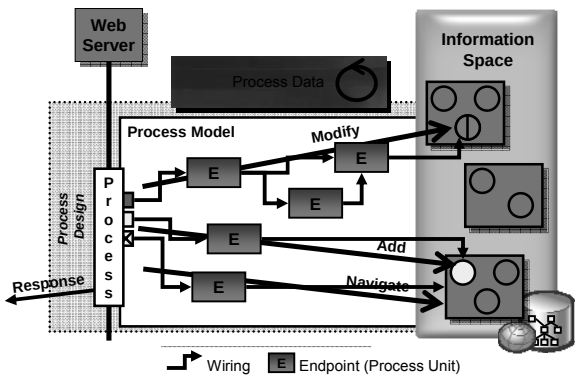
Process and the Web



Process Model and the Web



Process Model in Detail



Verteilte Web-basierte Systeme – SS 2006

Section://3

Endpoint Design of Services

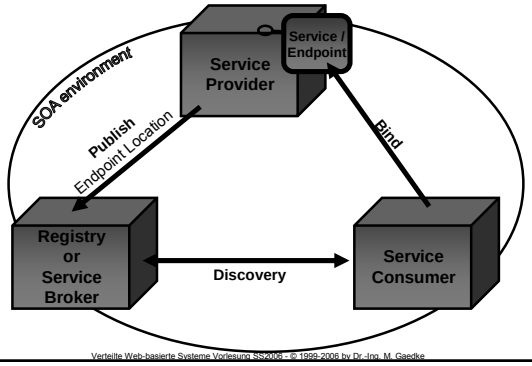
Introduction

- **Service Oriented Architecture (SOA)** – A (usually business-driven) architectural concept that defines the use and provision of services between participants of the SOA environment in a standardized way
  - A concept that is independent of specific technologies (network not required!) by focusing on the relationships between service provider, service broker, and service consumer
  - SOA environment defines the underlying technology, i.e. participants make or access resources within the environment
  - SOA allows for loosely coupling functionality (an ideal concept for implementing dynamic business processes)
  - Different SOA implementations exists, e.g. SOA with components, with web services, or with network nodes
- **Service** – An autonomous, self-contained, reusable software system (or black-box) designed to support a business function by message exchange in a standardized way – i.e. providing a set of Endpoints
  - Service implementation conform to a dedicated SOA implementation
  - Service usually accessible via network
  - Service *might* be described in a registry

Providing and Consuming

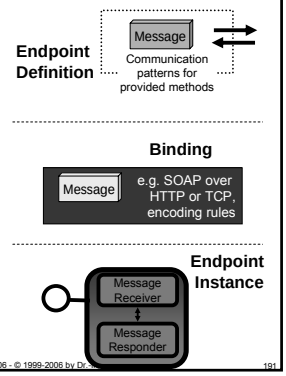
- Service-Provider
  - Develop functionality and provide it as a Service
  - Describe the service using Endpoint definitions
  - Publish the Endpoint (where to access the functionality)
- Service-Consumer
  - Service discovery - find functionality, i.e. endpoint location
  - Consume (i.e. access) functionality using endpoint definition
- Service Broker / Registry
  - Service broker stores publishing information from service provider
  - Service broker supports the discovery of dedicated service functionality
  - Service broker stores information about services in a registry – e.g. endpoint locations and other meta data like endpoint definitions

Service-Oriented Architecture



ABC of an Endpoint

- **C**ontract (Logical Design)
  - Definition of messages and communication patterns for consuming functionality
  - Describing the signature of provided functionality
- **B**inding (Physical Design)
  - Communication-related aspects
  - Impact of specific transport protocols wrt contract, e.g. message encoding
- **A**ddress (Deployment)
  - Location of service functionality's implementation
  - Service might have several addresses for endpoints at different locations



Service Model

- **Service Model** (Similar to Process Model)
  - Defines set of services / endpoints and relationships to applications/peers
  - Defines relationship aspects, i.e. required behaviors and assurances for connecting endpoints
  - Optional: Federation aspects, e.g. using WAM (WebComposition Architecture Model)
- **SOA Environments / Implementations**
  - Components as Services
  - Network Services
  - XML Web Services

Verteilte Web-basierte Systeme – SS 2006

Section://4

Components as Services

Introduction

- What is a Component
  - Different approaches for components exist
  - Term in use for many different aspects
- We will learn what approaches exist for designing and developing Web Applications and how they work together
  - Principles of modern Software Engineering
  - Component-Based Software Development (CBSD)

Understanding Components

- "Software Components are binary Units of independent Production, Acquisition, and Deployment that interact to form a functioning System"
  [Szyperski, "Component software: beyond object-oriented programming", 1997]
- Component
  - A unit of independent Deployment
  - A unit of third-party Composition
  - Has no persistent State

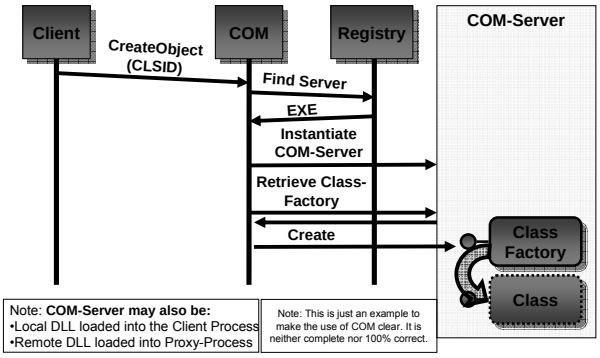
Component Environment

- A **Component** in this context is a part of the application logic – providing a required cohesive/atomic set of functionality.
  - Functionality may be calculations, providing data, storing documents etc.
- **Component Model** – defines how the components provide their application logic and how the functionality may be accessed.
  - Responsible for component compatibility

Components – Example

- **Component Object Model (COM)**
  - Microsoft's Component Technology until 2001
- Component is called COM Server
  - Providing Functionality as COM Class
- **Registry** - Component-Database stores location and metadata about Components
  - Components have to be registered first
  - Components can be and discovered by Registry
- COM Class
  - Source Code that implements COM Interfaces
  - Has a Unique Identifier (CLSID)
- COM Servers
  - In-Process Server: DLL loaded into client process
  - Out-of-Process Server: Executable File, either on same Machine or on remote Machine (DCOM, COM+) using RPC mechanism

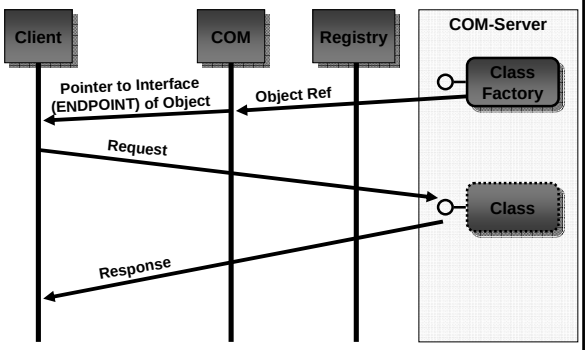
Components – Example



Note: **COM-Server may also be:**
•Local DLL loaded into the Client Process
•Remote DLL loaded into Proxy-Process

Note: This is just an example to make the use of COM clear. It is neither complete nor 100% correct.

Components – Example



Components – Example

- Endpoint in COM: COM Interface defines a contract of a software component
  - Set of Methods and Properties
  - A component may provide several Interfaces
- Each COM object must support at least one interface (IUnknown)
  - Endpoint Definition: QueryInterface, AddRef, Release
- DEMO: Windows Registry

Other CBSD Approaches

- CORBA – Common Object Request Broker Architecture
  - OMG's specification for interoperability between distributed computing nodes
  - ORB: middleware that establishes requestor-provider relationship
- Java Beans and Enterprise Java Beans
  - Notion of Virtual Machine
- Microsoft .Net Assemblies

Verteilte Web-basierte Systeme – SS 2006

Section://5

Network Services

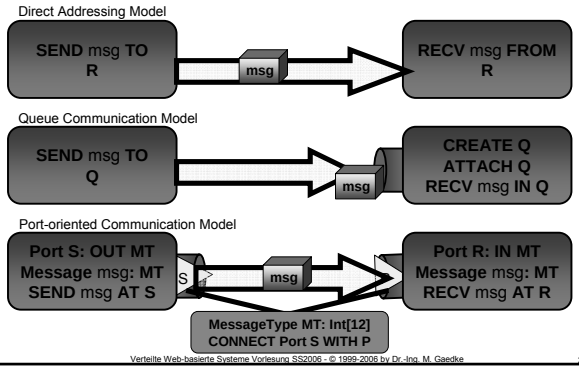
Web Computing Introduction

- Programming with distributed components on the Web
  - Calling interfaces / endpoints
  - Heterogeneous, distributed, multi-language environment
- Different approaches exists to wire distributed components
  - Wrt.: Wire Protocol, Wire Formats
  - Message passing, RPC, Web Service/SOAP

Message Passing Model

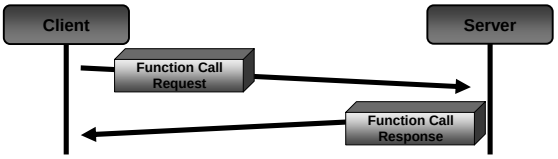
- Sender-Receiver Paradigm
  - **Message**: (Typified) Data transmitted from sender (S) to receiver (R)
  - **Symmetric** (sender and receiver know each other) vs. **Asymmetric** (only sender knows receiver)
  - Sender acts **synchronous** / **asynchronous**
- Types of Message Passing Models
  - Direct Addressing Model
  - Queue Communication Model
  - Port-oriented Communication Model

Message Passing Models



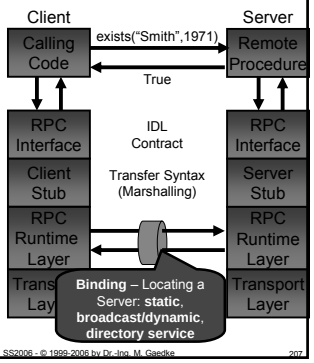
Request/Response Model

- Standard idea for Distributed Computing
  - Similar to programming-language behavior
  - In contrast to Message Passing models inherently synchronous (one operation defines communication vs. send/receive operations)



Remote Procedure Call

- **RPC** – Programming language based approach that allows applications to **synchronal** call individual functions that are located in **separate processes** (not necessarily on the calling machine) using a **small channel** for exchanging input and output data.
- **IDL** – Interface Definition Language that expresses the function's signature, including input, output, and input/output parameters.
- **Semantic**
  - **Exactly-Once Execution and Exactly-Once Delivery**



Interface Definition Language

- Endpoint Definition
- Language neutral specification allows for use by several languages
- IDL-Generator creates stubs in various languages
- Client application must know UUID (universally unique identifier) for an interface
  - COM: UUID (aka GUID) is used for CLSID and IID
- IDL is compiled away!!!
  - Interface information not available at run-time

- IDL-Example

```
[ uuid (9507dd10-b842-11ce-81e9-
0020afdd85c0),
version(1.0),
pointer_default(unique)]
interface UserDirectory
{
/* Implicit Binding */
short exists(
      [in] unsigned char *Name,
      [in] short int nYear);
/* Explicit Binding:
short exists(
      [in]handle_t hServer,
      [in] unsigned char *Name,
      [in] short int nYear);*/
}
```


Notes on RPC Approach

- Works well with small messages
- Regarding component platforms
  - DCOM platform limitation
  - CORBA, subtle incompatibilities require ORB from same vendor
  - Reliance on closely administered environments (Firewall will stop distributed computing in most cases)
- Programming difficulties in data alignment and data types
  - **Marshalling** - Process of encoding/decoding data based on wire format
  - Debugging

Verteilte Web-basierte Systeme – SS 2006

Section://6

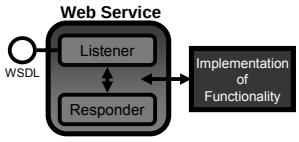
XML Web Services

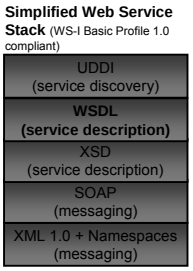
Describing Web Services

- Web Services Description Language (WSDL) 1.1
  - Status: W3C Note 15 March 2001
  - http://www.w3.org/TR/wsdl
  - Independent efforts from IBM and Microsoft
- WSDL is for describing Web Services
  - Defines XML-based grammar for describing network services as a set of endpoints
  - Describes their methods, arguments, return values and how to use

Web Services Architecture

- WSDL: Core element of the Web Service Architecture stack (Endpoint definition language)



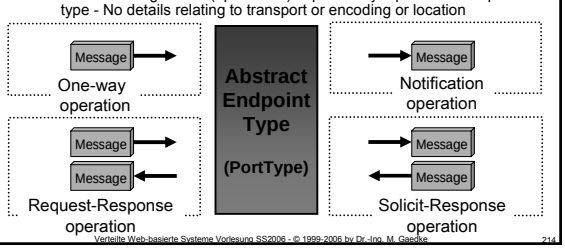


WSDL Goals

- Extensibility wrt.
  - New Transport protocols
  - New Encoding rules
- Abstraction wrt.
  - Endpoints and Messages
  - THEN mapped onto **n** concrete transports and encodings
- Reuse wrt.
  - Definitions – reuseable to create new definitions

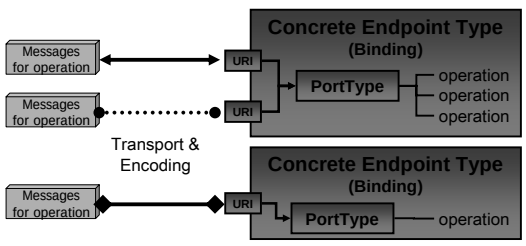
Abstract Endpoint Type

- Possibly part of a WSDL specification
  - Message
  - Operation
  - PortType (Abstract Endpoint Type)
    - Set of message flows (operations) expected by a particular endpoint type - No details relating to transport or encoding or location



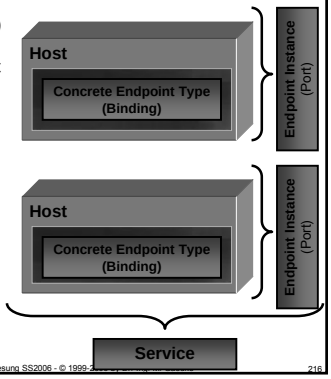
Concrete Endpoint Type

- Binding (Concrete Endpoint Type)
  - Defines transport and encoding particulars for a portType



Shift to Service Definition

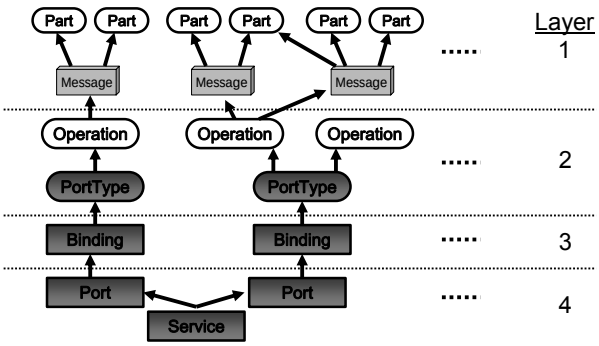
- **Port** (Endpoint Instance)
  - Network address of an endpoint and the binding it adheres to
  - Note – not necessarily a TCP port!
- **Service**
  - A collection of related endpoint instances



Describing Web Services

- All WSDL Elements belong to the WSDL namespace: http://schemas.xmlsoap.org/wsdl/
- Namespaces for WSDL Binding
  - SOAP Binding: http://schemas.xmlsoap.org/wsdl/soap/
  - HTTP GET and POST Binding: http://schemas.xmlsoap.org/wsdl/http/
  - WSDL MIME binding: http://schemas.xmlsoap.org/wsdl/mime/
  - More to come…
- Be aware of WSDL-first vs. Code-first
- Check rules at WS-I (www.ws-i.org)

From Parts to Service



Example Service / Layer 1

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
      xmlns:s="http://www.w3.org/2001/XMLSchema"
      xmlns="http://schemas.xmlsoap.org/wsdl/" …>
 <types>
   <s:schema elementFormDefault="qualified"
     targetNamespace="http://tempuri.org/">
    <s:element name="HelloWorld"><s:complexType /></s:element>
    <s:element name="HelloWorldResponse">
     <s:complexType>
          <s:sequence><s:element minOccurs="0" maxOccurs="1“
          name="HelloWorldResult" type="s:string" /></s:sequence>
      </s:complexType>
    </s:element>
    …
   </s:schema>
 </types>
```


Example Service / Layer 1

```
<message name="HelloWorldHttpGetIn" />
<message name="HelloWorldHttpGetOut">
   <part name="Body" element="s0:string" />
</message>
```


Example Service / Layer 2

```
<portType name="Service1HttpGet">
   <operation name="HelloWorld">
    <input message="s0:HelloWorldHttpGetIn" />
    <output message="s0:HelloWorldHttpGetOut"
   />
   </operation>
</portType>
```

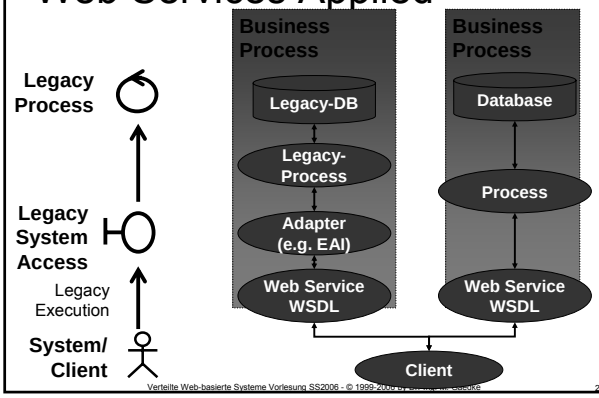

Example Service / Layer 3

```
<binding name="Service1HttpGet“
   type="s0:Service1HttpGet">
   <http:binding verb="GET" />
   <operation name="HelloWorld">
      <http:operation location="/HelloWorld" />
      <input><http:urlEncoded /></input>
      <output><mime:mimeXml part="Body" />
      </output>
   </operation>
</binding>
```

Example Service / Layer 4

```
<service name="Service1">
   <port
      name="Service1HttpGet"
      binding="s0:Service1HttpGet">
      <http:address
      location="http://local/Hello" />
   </port>
</service>
</definitions>
```


Web Services Applied



Homework

- What is the difference between
  - Code first

  and
  - WSDL first?
- What is WS-I ?

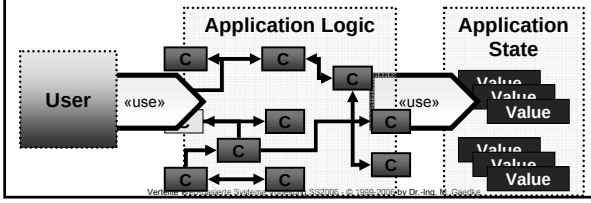
Verteilte Web-basierte Systeme – SS 2006

Section://7

Services in context

Why State Management?

- Core Communication Model of the Web (HTTP) is stateless
- Application requires state when a user traverses the information space of a Web Application



Session Design

- **Session** – Defines a context in which a user communicates with a Web Application in a defined time period
  - One Session per user
  - Assigns application state to multiple requests from one user
- Design Decision / Rules of thumb
  - Use a database to persist state
  - UUID to identify a session/user
- Physical Design: Session identifier exchange
  - Cookie, hidden variable, or encoded into the URL

Dynamic & Transactional Links

- **Dynamic Links –** Describe relationship between endpoints in the Wiring Model that exist based on state and application logic
- **Transactional Links –** Describe relationship between endpoints in the Wiring Model that exist only as a whole

Transactions And The Web

- In some Web Application scenarios you have a series of correlated operations corresponding to consecutive HTTP requests. You need to ensure that if one single operation fails, all related operations fail.
- Example:
  - Booking a flight and a hotel at the destination. The hotel is not necessary if no flight is available for the stay. Transaction processing is the technology that enables you to control the process as a **whole**.

Transactions

- **Transaction** – A unit of work that should either succeed or fail as a whole. A series of operations that behave corresponding to the ACID rules.
  - Series: BEGIN_TRANSACTION, Op1, …, OpN, COMMIT_TRANSACTION
  - ACID Rules define Atomicity, Consistency, Isolation, and Durability
- Characteristics regarding Web Applications
  - Long Running
  - Nested

Atomicity And Consistency

- **A**tomicity
  - Transaction executes exactly once and is *atomic*
  - All the work is done or none of it
- **C**onsistency
  - Transaction preserves the *consistency* of data
  - Transforming one consistent state results in another consistent state of data

Isolation And Durability

- **I**solation
  - Transaction is a unit of *isolation*
  - Concurrent transactions behave as though each was the only transaction running in the System
- **D**urability
  - Transaction is a unit of recovery
  - If a transaction commits, the system guarantees that its updates will persist, immediately after the commit.

Effects On Web Services

- Transactional Links requirements
  - Declaration of Web Services to be executed within a transaction
  - Specify transaction properties on every method of a Web Service
- Practical Approach – Transaction Properties describing Context
  - Supported, NotSupported, Required, RequiresNew
  - ITX (identifier) for Internet Transaction
  - Other Approaches are possible

Context Description Approach

- Supported
  - Transaction exists implies the method will run in its context
  - No transaction implies the method will not run within a transaction
- Not Supported
  - The method will not run within a transaction
- Required
  - Transaction exists implies the method will run in its context
  - No transaction implies a new one will be started
- Requires New
  - A new transaction will be started on each call.

Transaction Design

Some Rules of Thumb

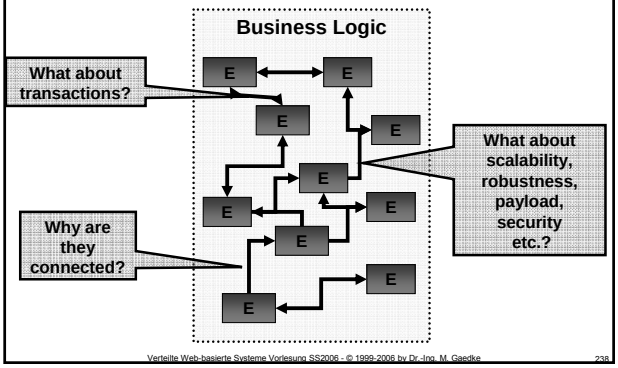
- Use a supporting System
  - e.g. TP-Manager, Database
- Design Components with transactions in mind
  - Transactions are powerful but imply overhead
  - Not every component 'requires' a transaction
- Be aware of the Transactional Semantics of the underlying system or database
  - Long-lived Locks in the database will kill performance
  - Look for blocking and deadlocks when testing

Verteilte Web-basierte Systeme – SS 2006

Section://8

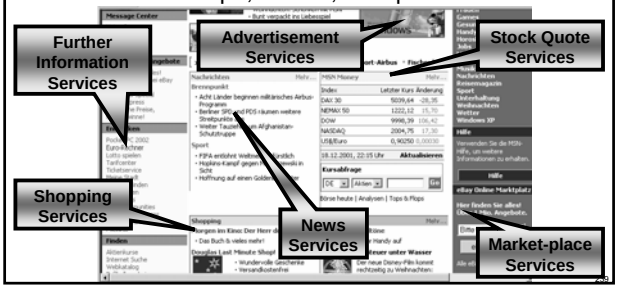
Web System Aspects

Introduction: Distribution Aspects



Portal Accessing Systems

- **Portal** – Web Application that provides uniform access to different content source
  - Screen Scrape, WSRP, Uberportal etc.



Some Observations

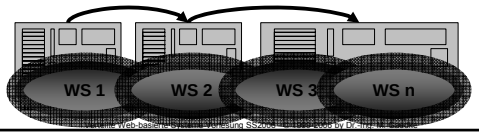
- Communicating Processes may be *evolving*
  - Hosted on physical machines
  - Mediated by both physical and logical channels
  - Physical hosts and media are subject to failure by various hazards
  - Software and hardware configurations may change during the lifetimes of Business Processes
  - **Business Process Communication must be robust against changes intended (Version Update) or not (Failure)**

Some Observations

- Communicating processes may be *mobile*
  - Long-lived Business Processes will undergo multiple resource (re-)allocations
  - A process may begin its activities on one resource and continue on another
  - Mobility increases the challenge to Business Process robustness

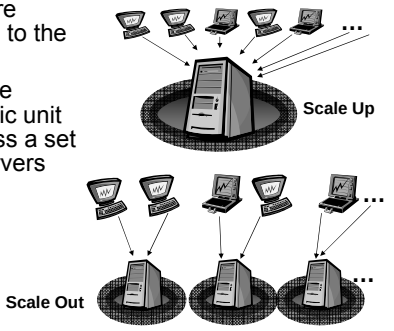
Simple Web Service Chain

- Web Service WS 1 provides functionality using WS 2, WS 2 provides…
  - Like a chain: The weakest element influences the overall behavior
- **Hops** - Represents the number of network nodes involved from the source WS to the destination WS. Example shows 2 Hops, 4 Web Services



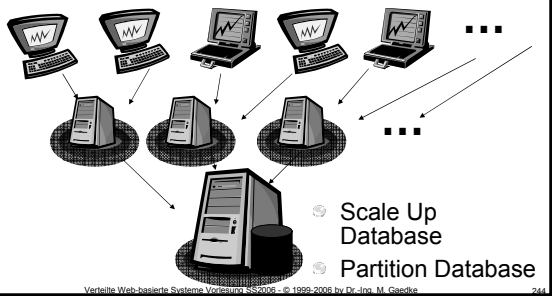
Considering Scalability

- **Scale Up:** More "power" added to the machine
- **Scale Out:** The application logic unit is cloned across a set of identical servers



Scale-Out and Partition

- Scale out Web Servers and scale up Database



- Scale Up Database
- Partition Database

Partition Database

- Functional – Each functional area of a site gets its own database
  - Dedicated hardware to certain functions
  - Class of hardware per function
- Tables - Huge scale opportunity for large tables
  - Some modern database management systems provide special support for this
- Read-only Databases
  - Data changes do not occur often, e.g. Lecture Catalog
  - Use of Replicated Databases

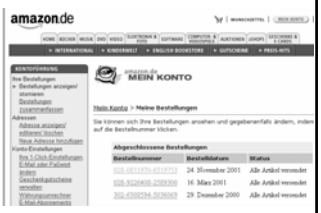
Partition Web-Tier

- Like Functional Database Partition
  - E.g. Partitions: *search*.Business.com, *www*.Business.com etc.

- DNS Host Names or Hardware Solutions exist to distribute traffic to dedicated server/clusters
  - Simple Approach: DNS Round-Robin
  - E.g. www.myserver.com refers to several IP Addresses / physical servers

Using Messaging Approaches

- Provide a high degree of scalability by decoupling the user experience from the backend processing
  - Asynchronous processing
- Example:
  - Order process consists of 3 stages (Producing, Packaging, Shipping) – usually takes some days
  - After ordering – User can check status of progress

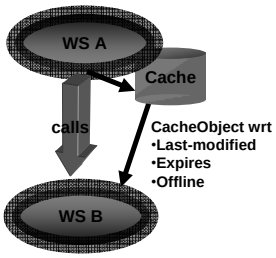


Using Messaging Approaches

- Use queue- or port-oriented communication model where applicable
- Using asynchronous programming techniques whenever there are:
  - Opportunities for parallel processing
  - Batch type of operations
  - Interfacing with legacy applications
  - Real-time Operations

Considering Optimization

- Optimization ≠ Performance Tuning
- Reducing WS-Calls
  - Use caching or offline content generation
  - Check which navigation scenarios
- **Caching Approach**
  - "Good Enough Hit" Function – at Olympics site



Example: Caching Candidates

Optimization

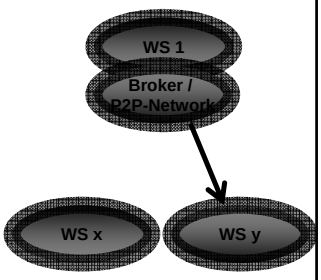
- Often Wiring Models include dynamic or transactional relationships
- Integration – Still a bit of black magic or "Art and Experience"
- Different approaches exist
  - Model dependent optimization is possible

Broker and P2P Approaches

- Communicating processes may be *brokered by dedicated service (Broker)* or *an "intelligent" network (P2P network)*
  - Communications among business processes will often be requests for or provisions of Web Services
  - The consumer and provider likely do not have *a priori* knowledge of one another
  - Services will be mediated by **Brokers or the underlying P2P network**

Dynamic WS Discovery

- Web Service calls Web Service mediated by Broker (respectively P2P network)
  - Criteria may be quality, context, price, etc.
  - Requires classification system or metadata
- Broker could use UDDI automatically on request
- P2P discovery by content-based routing (e.g. for WSDL)



Verteilte Web-basierte Systeme – SS 2006

Section://9

Federated Systems

Introduction

- Goal: Bring business processes together
  - Globalize the Component-based View
  - Extend processes with external (potentially unknown) partners
- Idea: Federating Web Applications (respectively their Logical Units)
  - Take identity and access management (IAM) into account
  - Define protocols to support inter-organizational information exchange in a standardized way

WAM (Federation Model)

- WebComposition Architecture Model
  - Introduced in 2005 by Gaedke and Meinecke
  - Consists of several models
  - Applies UML-like notation in combination with OCL
- Six core entities
  - Can be connected by protocols
  - Nested within zones/realms
  - Each Connection is **labeled** with a shortcut, which is used for detailed description (cf. OCL) in addition to the graphical notation
  - Labels and their details are stored in a dedicated database, i.e. labels once defined can be reused in later projects

WAM Core Entities (1)

- *Service*
  - Represents the system's distributed (atomic or composite) components
  - E.g. SOAP Web service

- *Application*
  - Allows users to interact with the overall system
  - E.g. Web applications or portals

- *Data Provider*
  - Distinguish between the services and the underlying systems that serve as the actual data sources
  - Connected to service or application with undirected line

- *Process Unit*
  - Connected systems that perform functionality beyond data management
  - E.g. software that performs computations or triggers events

WAM Core Entities (2)

- *Security Realm*
  - Envelopes applications, services, data provider and process untis as organizational zones of control – as such functions as identity and access management context
  - E.g. defines set of roles and permissions
  - Realms might be nested
  - Implemented e.g. as a Security Token Service

- *Identity provider*
  - Store for accounts/identities (of known users as well as applications)
  - Allow to authenticate the members of the realm – issues security tokens
  - E.g. through login forms or Web service interfaces
- Name Label
  - These label represent a naming context for each entity
  - Naming-Labels might be used as shortcut for a detailed description of these entities

Name Label

WAM Core Relations

[i-label]

- *Invocation (Communication Profile)*
  - Potential accesses on services and applications
  - Labels indicate the designated communication protocols (label acts as a shortcut for a detailed description of the communication relationship)
  - E.g. SOAP via HTTP, SOAP via SMTP, WS-I compliance etc.

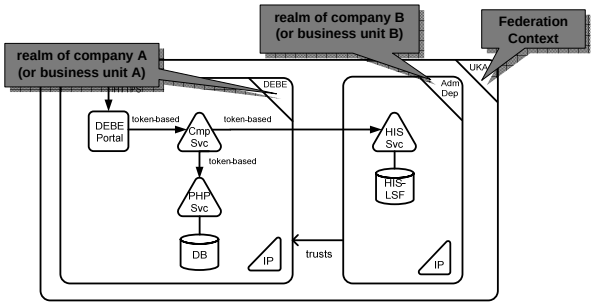
[t-label]

- *Trust (Trust Profile)*
  - Trust-label separate realms that form a federation
  - STS of the trusting realm accepts the tokens originating from the trusted realm (label acts as a shortcut for a detailed description of the trust relationship)
  - Identities of the foreign requestors can be mapped to tokens that are locally valid – these relationships are defined for the trusts labels

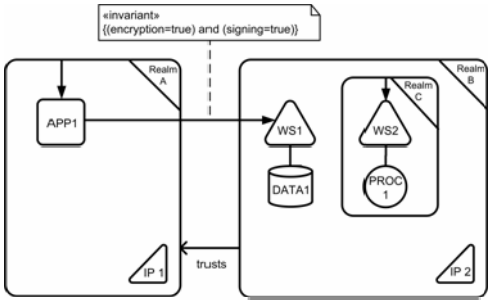
[f-label]

- *Functionality (Functionality Profile)*
  - Links Web Service technology with functionality
  - E.g. technology in use for calling process unit or data provider
- OCL might be used to describe details of Invocation and Trust

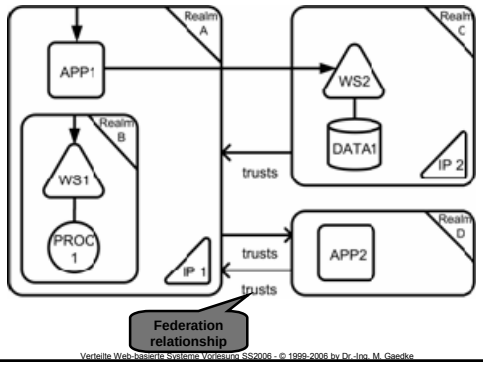
WAM-Modeling Example*



WAM Examples (1)



WAM Examples (2)



Verteilte Web-basierte Systeme – SS 2006

Chapter://6

Further Readings

Literature

- Chapter 3, 6: Thomas A. Powell, *Web Site Engineering*, Prentice Hall PTR
- Chapter 7, 8, 9, 15: David Lowe and Wendy Hall, *Hypermedia and the Web – an Engineering Approach*, John Wiley & Sons
- Chapter 10, 11, 12, 14, 16: Ian Sommerville, *Software Engineering*, Addison-Wesley
- Chapter 1-4: Scott Seely, *SOAP – Cross Platform Web Service Development Using XML*, Prentice Hall
- Chapter 1, 2, 5: William L. Oellermann, Jr., *Architecting Web Services*, apress
- Chapter 2, 3, 4, 5: M. Mühlhäuser, A. Schill, *Software Engineering für verteilte Anwendungen*, Springer
- Chapter 9: I. Jacobson, G. Booch, J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 1999
- Chapter 7, 8: S. W. Ambler, *Process Patterns – Building Large-Scale Systems Using Object Technology*, Cambridge University Press

Literature

- K. Fellner und K. Turowski, *Approach to Solve Content-related Conflicts in Component-based Business Application Systems*, 12th International Conference Software & Systems Engineering and their Applications (ICSSEA'99), Paris, 1999.
- M. Fowler, Kendal Scott: *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Addison-Wesley Longman, Inc, 1999
- M. Gaedke, M. Beigl, H.-W. Gellersen und C. Segor, *Web Content Delivery to Heterogeneous Mobile Platforms*, in Lecture Notes in Computer Science (LNCS), Springer Verlag, vol. 1552(Advances in Database Technologies), S. 205-217, 1998.
- H.-W. Gellersen, F. Lyardet, M. Gaedke, D. Schwabe und G. Rossi, *Patterns and Components: Capturing the Lasting amidst the Changing*, Active Web Conference, U.K., 1999.
- D. M. Kristol und L. Montulli, *HTTP State Management Mechanism*, IETF, Network Working Group, RFC RFC 2109, Feb. 1997 1997. ftp://ftp.isi.edu/in-notes/rfc2109.txt
- D. A. Menasce, V. A. E. Almeida, *Capacity Planning for Web Services*, Prentice Hall, 2002.
- B. Oestereich: *Objekt-orientierte Software Entwicklung mit der UML*, Oldenbourg Verlag, 1997

Literature

- K. Ostebye, Hierarchical Structure through Navigation Side Bars, Workshop on Hypermedia Development - Design Patterns in Hypermedia, Hypertext Conference (HT99), Darmstadt, 1999.
- G. Rossi, D. Schwabe und F. Lyardet, Improving Web Information Systems with Navigational Patterns, 8th International World Wide Web Conference (WWW8), Toronto, Ontario, Canada, 1999.
- J. Sametinger, Software Engineering with Reusable Components. Berlin: Springer, 1997.
- R. Steinmetz und K. Nahrstedt, Multimedia : computing, communications, and applications. Upper Saddle River, NJ: Prentice Hall, 1995.
- C. Szyperski, Component software: beyond object-oriented programming. Reading, Mass.: ACM Press; Addison-Wesley, 1997.
- A. S. Tanenbaum, Computer networks, 3rd ed. Upper Saddle River, N.J.: Prentice Hall PTR, 1996.

==========

Further information available at Lecture Web Site

==========

Web References

- URLs
  - OOHDM, WebML, WebComposition, etc. via: http://webengineering.org/
  - Patterns Home Page: http://hillside.net/patterns/
  - Patterns for E-commerce applications: http://www-di.inf.puc-rio.br/~schwabe//papers/Europlop00.pdf
  - Hypermedia Design Patterns Repository: http://www.designpattern.lu.unisi.ch/HypermediaHomePage.htm