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"
   1. Logical Design
   2. Physical Design
   3. DTD
   4. Schemas
   5. Example
3. User Interface Experience
   1. Presentation Design
   2. Navigation Design
   3. Dialogue Design
4. Web System
   1. Introduction
   2. Endpoint Design of Processes
   3. Endpoint Design of Services
   4. Components: Services
   5. Network Services
   6. XML Web Services
   7. Services in Context
   8. Web System Aspects
   9. Federated Systems
5. Further Readings

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

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)
  - Do not encapsulate interactions with actors!

- **control**
  - Models coordination, sequencing, transactions, and control of other objects (e.g. Business Logic)

Transforming Example

**Analysis Activity**

- `specify`
- `check Availability`

**eConcierge System**

- `Customer`
- `Request Room`
- `Room`
- `Availability Handler`
- `Conferrency Room`

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

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

- `Creation requires approval by Management Level > 2`
- `Room Order`
- `Food Order`
- `Conference Room`
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
- Access at different times returns the same result
- Access at different times may return different results

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

Interactive vs. Passive
- Unidirectional from provider to consumer
  - No (very little) decision for the consumer
- Interactive
  - Information exchange in both direction from provider to consumer and vice versa
  - Symmetric, e.g. video-conferencing
  - Asymmetric, e.g. options to choose from, Wiki principle

Passive vs. Interactive
- 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)
Aspects of Interaction

Human Computer Interaction (HCI)
- User is interacting with the computer in order to accomplish something
- It’s not audio. It’s not video. It’s user control and dynamic experience

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.

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

What is 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

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
- Contex-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

Manipulate / Render
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
- 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, notes, firewall perimeter etc.

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


Further information available at Lecture Web Site

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

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

- For each Information System Element
  - Information System Architect (ISA)
  - Designer responsible for mapping ISE
  - Component Engineer
  - Use Case Engineer

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
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

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
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...

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

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

- 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"?>
Document Type Definition

- **Document Type Definition Syntax**
  - DocTypedef ::= <!DOCTYPE S Name (S ExternalID)?
    S? (I (Markupdecl | DecSep) | S)? >
  - DecSep ::= 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

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

#### Diagramming Technique 2

- **(A,B)** Concatenation/Series B after A
- **(A|B)** Selection A or B

**Example**

```
Account: (Room, (Minibar,Food+)?, Total)
Total= (#PCDATA)
```

#### 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** specification

**Example**

```
<account AccountID="a3499bd0z">  
  <room number="R101"/>  
  <minibar/>  
  <food price="10.00"/>  
  <food price="15.00"/>  
  <total>25.00</total>
</account>
```
### Mixed Specification

- **Syntax:**
  
  ```
  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**

- **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 tryenum (true|false|don'tknow) '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 | #VALUE) AttValue)
  ```

- **Example:**
  
  ```
  #Required – Attribute must be specified
  #Implied – Attribute is optional
  #Fixed – Required attribute; value is specified in quotes
  #AttrValue – 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)

Entity Examples

Common Entity Declarations
- &lt;ENTITY &lt;#38;&gt;
- &lt;ENTITY &gt;#38;&gt;
- &lt;ENTITY &amp; #38;#38;&gt;
- &lt;ENTITY &apos;#39;&gt;
- &lt;ENTITY &quot;#34;&gt;

Character and Entity Reference
- Character: &lt;
- Entity (Declaration above): &lt;

Parameter-Entity Reference for order.dtd:
- Declaration: <!ENTITY % minibar.items "book | cdrom"> Usage: <!ELEMENT item (%shop.items;)+>

Mean: 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>&#13;<month=13</month>

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.

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 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)
  - `<person xmlns:P="urn:person">
    <P:firstName>Franz</P:firstName>
    <P:lastName>Huber</P:lastName>
    <P:code>4711</P:code>
  </person>`
- Customer (schemas applied)
  - `<person xmlns:C="urn:customer">
    <C:code>0815</C:code>
  </person>`
Complex Type Definitions II

- Use of Complex Types - Example:
  ```
  <xsd:schema id="PurchaseOrderType" targetNamespace="urn:customer" xmlns="urn:customer">
    <xsd:element name="PurchaseOrder" type="xsd:complexType"/>
    <xsd:complexType name="PurchaseOrder" mixed="false">
      <xsd:sequence>
        <xsd:element name="billTo" type="USAddress"/>
        <xsd:element name="shipTo" type="USAddress"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
  ```

  - Ref Attribute (here its value is comment) indicates a Reference to elsewhere declared Element (global Element)
  - Comment Element here is optional due to Occurrence Constraint

Complex Type Definitions III

- Element occurrence constraints:
  - minOccurs = 1
  - maxOccurs = 42
  - Default value for minOccurs = 1

Example:
- minOccurs = 1, maxOccurs = 42
  - Default value for minOccurs = 1
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

- 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:restriction>`
      - `<xsd:simpleContent>`
    - `<xsd:complexType>`

Element simpleContent

- 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>`

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>`

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.

Example: Room Entity

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

```xml
<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>
```

Number: String, PK
Name: #PCDATA
Descr: Text: #PCDATA
Booked

References
- DTD or XML-Schema, e.g. ID and IDREF
  ```xml
  <booked roomref="R10" by="MrG/>
  <Guest gid="MrG">...
  ```

- Shift to Physical Design
  - If XML many opportunities... for enhancements
  ```xml
  <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>
    ...
  </rdf:RDF>
  ```

Rethinking: The Room Entity

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

  ```xml
  ...
  ```
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

User Interaction Experience

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!

**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**
- Audience
- User Model
- Look&Feel Model
- User Interface Model (UIM)
- Information Space

**Tiles**
- HEADER
- BODY
- Current Topic Headline
- Collaboration
- Author
- Abstract
- Related Links
Tiles

Core Tiles

Tiles – Structure and arrange UI Elements

Controls

Controls are UI Components

Control Aspects

Control Examples

Copyright © 1999-2006 Dr. Martin Gaedke
Simplified Control Model

UI Element
- Tile
- Control
- Decorator

Text
- Text building blocks
- Inline elements

Interactive
- Form Elements
- Applets
- ActiveX Control

Navigation
- Hypermedia structures
- External links
- Internal links
- Navigation pattern

Composite
- Composition of UI elements
- Consider rules for composing elements, e.g. WAI compliance

Interactive Controls

SelectOne
- SelectOne
- Text
- Text
- MultiSelect
- MultiSelect
- Text
- Text

XForms Source: XForms 1.0
W3C Working Draft
28 August 2001
http://w3.org/TR/xforms

Shift to Physical Design

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

- XHTML
- WML
- Voice XML

Look&Feel Model
Other

Physical Design Examples

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

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

Verteilte Web-basierte Systeme - SS 2006

Section://2
Navigation Design

© 1999-2006 by Dr.-Ing. M. Gaedke
Hypermedia Linking

- **Link** – Association between a Source Node and one or more 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:**

![Diagram](image)

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 NUCAHTH, 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
### 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

- 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
  - Analysis 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**

**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

**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.

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

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.

**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.

**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?

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

Solution:
- Structure the Home Page in such a way that space is devoted to newest additions. 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

3. Access by Search Request
   - Provide search facilities
   - Solution

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

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

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)

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), Backbone, 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:

```
<street>
  <street_name>Street</street_name>
  <number>123</number>
</street>
```

Selecting One Flavor

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

Choose flavorful chocolate implies select only one.
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

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

Wiring Distributed Endpoints

- **Distributed Endpoints** – Endpoints are deployed to remote hosts
- **Deployment** – describes distribution of processes and endpoints to processing unit (hosting node, host)

Examples

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
- **Highly Dynamic Approaches**
  - Highly Dynamic Approaches
  - E.g., using Broker or Peer to Peer Networks (P2P networks)

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
Part V - Chapter 5: Web System: Endpoint Design of Processes

Distributed Business Process

- Network of hosted functionality
- e.g. Web Server
- e.g. DBMS
- e.g. Registry
- e.g. Web Service

Business Process Deployed

- Business Process
- Participant
- Call
- Communication & Contract Negotiation
- Endpoint Logical Entity to access functionality on the Server

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

- Web Server
- Information Space
- Business Process
- User Agent
- Request for Operation
- Add
- Modify
- Navigate
- Process Data

Process Model and the Web

- Web Server
- Information Space
- Process Design
- Process Data
- Modify
- Add
- Navigate
- Business Process

Process Model in Detail

- Web Server
- Information Space
- Process Model
- Modify
- Add
- Navigate
- Wiring
- Endpoint (Process Unit)
ABC of an Endpoint

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

Service Model

1. **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)
2. **SOA Environments / Implementations**
   - Components as Services
   - Network Services
   - XML Web Services

![Diagram](image-url)
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

Client

CreateObject (CLSID)

COM

Registry

Find Server

EXE

Instantiate COM-Server

Retrieve Class-Factory

Create

Remote DLL loaded into Proxy-Process

COM-Server

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 show the use of COM, it is rather complex and it works
Components – Example

- Client
- COM
- Registry

- COM-SERVER
- Request
- Response
- Class
- Factory

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

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
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

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)

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) ]
/* Implicit Binding */
interface IDemo_
{ { 
    [in] short exists( short int nYear );
    [in] unsigned char *Name,
    [out] short exists( short int nYear );
    [out] unsigned char *Name,
    [in] unsigned char *Name,
    [in] short int nYear );
}
```

Remote Procedure Call

- RPC – Programming language based approach that allows applications to synchronous 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

Message Passing Models

- Direct Addressing Model
  - SEND msg TO R
- Queue Communication Model
  - SEND msg TO Q
- Port-oriented Communication Model
  - Port S: OUT MT
  - Message seq: MT, SEND msg AT S
  - Port R: IN MT
  - Message seq: MT, RECV msg AT R
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 – reusable 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)

Example Service / Layer 1

```xml
  <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>
</definitions>
```

Example Service / Layer 2

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

Example Service / Layer 3

```xml
<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

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

Web Services Applied

<table>
<thead>
<tr>
<th>Legacy Process</th>
<th>Legacy System Access</th>
<th>Legacy Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Part V ► Chapter://5 ► Web System: XML Web Services

Homework

- What is the difference between
  - Code first
  - WSDL first?

- What is WS-I?

Verteilte Web-basierte Systeme - SS 2006

Section://5/7

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

© 1999-2006 by Dr.-Ing. M. Gaedke
### 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

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

### Isolation And Durability

- **Isolation**
  - Transaction is a unit of **isolation**
  - Concurrent transactions behave as though each was the only transaction running in the System
- **Durability**
  - 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

Not Supported
- The method will not run within a transaction
- 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

Introduction: Distribution Aspects

- What about transactions?
- What about scalability, robustness, payload, security etc.?
- Why are they connected?

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

Part V

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
- 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

- 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)

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 units 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
WAM Core Relations

- **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.
- **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
- **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*

- Realm of company A (or business unit A)
- Realm of company B (or business unit B)
- Federation

WAM Examples (1)

- Example of a service relationship between two companies

WAM Examples (2)

- Example of trust relationships between services

WAM Examples (3)

- Example of functionality links between services

Further Readings

- 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

© 1999-2006 by Dr.-Ing. M. Gaedke
Literature


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/HypermediaHome Page.htm