Constructing and integrating data-centric Web applications: methods, tools, and techniques

Stefano Ceri
DEI, Politecnico di Milano
Stefano.Ceri@polimi.it

Ioana Manolescu
INRIA Futurs
Ioana.Manolescu@inria.fr

Joint work with: Aldo Bongio, Marco Brambilla, Sara Comai, Piero Fraternali

Outline of the tutorial

I. Problem definition: design and construction of a web-based information system
II. Layers of the solution:
   1. Data modeling: define the content
   2. Hypertext modeling: define the Web application
   3. Personalization: give a different view to different users
   4. (Presentation: add the look-and-feel)
   5. Integrating business processes
   6. Integrating Web services

At every layer:
   Example within the WebML framework
   Comparison with other techniques

I.1 Problem definition

Examples of Web-based information systems

- Commerce-oriented
  - Electronic catalogs, auctions, virtual marketplaces
- Content-oriented
  - Online newspapers, digital libraries
- Service-oriented
  - Order tracking systems, reservation systems, tourist information systems
- Community-oriented
  - Portals, message boards, technical communities

Web-based information systems

- A Web-enabled software system whose main purpose is to publish and maintain large amounts of data
  - With browsing-oriented interfaces
  - Supporting context-dependent navigation
  - Publishing contents on (arbitrarily complex) Web pages
  - With data stored by means of DBMS technology
    - Dynamic page computation from DB content
    - DBs possibly distributed, heterogeneous, and pre-existing the Web application

Trends in data-intensive Web sites

- Encoding business processes within Web applications
  - Multi-agent applications
  - Integrating the notions of: process, activity, work assignment, and the classical workflow organizations (sequence, concurrent parallelism, mutual exclusion)
- Building interoperability by means of Web services
  - The modern way of deploying distributed applications
  - A way for organizations to make their Web applications published for use by other applications
  - These issues are orthogonal
Further trends and complexity factors

- Multi-modal, multi-device applications
  - Need of modeling both the generic and the device-specific parts of the application
  - Technologies: PC, PDA, WAP phones, 3rd gen phones, Digital TV, videotext
- Personalisation and one-to-one delivery
  - Need of modeling both the generic and the user-specific components of the application
  - Known applications: myYahoo, myCDNOW,…
- Presentation
  - An essential part of any Web application
  - Should be orthogonal and addressed by presentation specialists

Running example: a bank application

- Bank clients own a personal account
  - Day-to-day operation:
    - Inspect last transaction
  - E-mail bank employee supervising the account
  - Loan application:
    - View loan conditions proposed by the bank
    - Supply applications with personal loan parameters
- Bank employees
  - Inspect client accounts and salaries
  - Respond to e-mails
  - Bank loan applications (background checks)
- Bank managers
  - Approve loan contracts

Business processes and Web services in the running example

- Encoding business processes within Web applications
  - Clients apply for loans
  - Employees must perform two parallel checks (employment history and savings history)
  - If both checks succeed, managers approve application.
- Web service support
  - One or both checks may be provided by external components (web services)
  - The loan application process in itself may be exported as an external Web service

I.3 Why current Web application development practice doesn’t solve the problem

- Lack of methods and models
  - Lack of a well-founded software engineering methods
    - Data-centric methods do not cover the hypertext front-end
    - OO methods (e.g., UML profiles) do not capture the essence of Web-based systems
  - Lack of model-driven support
    - Navigation and presentation poorly modelled
    - Lot of hand-written code
    - Big efforts are requested even for prototyping
    - The total cost of ownership is dominated by re-engineering and maintenance
Web modelling & Conallen’s UML Web extension

- UML Web modeling extension [Con00]
- Based on the UML extensibility features: stereotypes
  - Page modeling stereotypes:
    - Server page, client page, target, frameset, form
  - Navigation modeling stereotypes:
    - Link, targeted link, redirects, submits

Notations and example

Evaluation

Pros:
- Implemented in various CASE tools (e.g., Rational Rose XDE)
- Based on a universal notation (UML)
- User-extensible

Cons:
- Low-level abstractions
- Only a syntax, no specific Web semantics
- Diagrams of realistic Web applications tend to become quickly unmanageable
- Web pages are not object-oriented!

Web modelling & ER Oracle Designer

Pros:
- Implemented extension of the data modeling methodology proposed by Oracle
- Add-ons for modelling Web interfaces
  - Modules = groups of related pages
  - Entities = publishable or updatable content
  - Navigable Relationships = links between entity instances

Cons:
- Data-centric abstractions
- Rigid in modeling Web-specific features (e.g., server-side business logic, presentation)
- Web applications with realistic requirements not manageable
- Web pages are not data structures!

Example: module diagram
Process modeling within Web applications

- BEA WebLogic Workshop:
  - Visual tool for designing Web applications including: pages, actions, navigation...
  - Also Web service specification
  - Code generation for the Web application skeleton, including state management, sessions etc.
  - Runs on top of an application server
  - Includes the tools for "real" workflow management

Process modeling in BEA WebLogic Workshop

- Java + custom extensions for Web app. process
- Graphical view of the process
- Code writing still required to "fill in" each activity in the Web application

Evaluation

- BEA WebLogic Workshop: integrating layer on top of many others
  - Workflow engine
  - Data integration layer
  - Transaction manager...
- Benefits from a high-level workflow modeling
- Internals of individual activities still have to be coded by hand

II
Model-driven design of web applications

Advantages of a model-driven approach

- A rigorous modeling approach:
  - Reduces development efforts (cost and time)
  - Allows a more structured development process
  - Produces more usable and coherent applications
  - Ensures better quality documentation
  - Grants immediate and low-cost prototyping through automatic code generation
- Successful examples in other fields:
  - ER for data design
  - UML for OOA&D
  - VLSI design
  - CAD systems in the manufacturing industry

Conceptual modeling: where & why

- Data Design:
  - The application content should be modeled in a platform-independent way, then mapped to different logical models
- Hypertextual Design:
  - Hypertextual interfaces (content+navigation) should also be modeled at high level in a platform-independent way, then mapped to physical structures (page templates, data extraction components, ..).
WebML
Development process

Why data and hypertexts?
- The content of dynamic Web sites is computed by extracting it from (dedicated) data sources
  - the data schema is a "data independent" abstraction of its content
- The WEB is a gigantic network of nodes and links
  - the hypertext schema is a "technology independent" abstraction of its organization

Web Modeling Language (WebML)
- WebML: a conceptual language for high-level design of data-intensive Web applications
  - Defined in 1998, in use for more than six years
  - Adopted in many universities worldwide
  - Commercially implemented (www.webratio.com)
- Used for developing several applications:
  - www.acer-euro.com,
  - www.aceradvantage.com,
  - www.elet.polimi.it, www.image.co.uk,...

WebML
From model-driven design to code generation

WebML
Hypertext conceptual modelling with WebML
- Visual Web application modeling language
- WebML specifications consists of:
  - One data schema (E-R, UML class diagrams)
  - One or more hypertext schemas (site views)
- Presentation is dealt with a standard language (XSL rules)

II.1
Data Modelling
Data modeling concepts

- **Entity**: a class of objects in the application domain
- **Attribute**: a property of an entity
- **Relationship**: a binary connection between entities (with cardinality constraints)
- **IS A Hierarchy**: used for classification and grouping (not shown in the tutorial)

Simple bank application data schema

- **User**
  - Name
  - Password
  - Email
  - Type
- **Account**
  - Number
  - Transactions
  - Amount
  - Type
- **Transaction**
  - Date
  - Amount
  - Type
- **Group**
  - Name
- **Application**
  - Date
  - Loan
  - Type
  - Interest
- **Transaction**
  - Date
  - Amount
  - Type

II.2 Hypertext Modelling

Hypertext Modeling: purpose

- High-level modeling of:
  - A dynamic Web application
  - Interactions with the back end, business logic, and data
  - Using a simple, yet formal, visual notation
  - Enabling automatic generation of:
    - Dynamic page templates and
    - Data access and manipulation queries

Content Units

- A **WebML unit** is the atomic information publishing element
  - unitX
    - container
      - [selector]
  - It is a “view” defined upon a **container** of objects including:
    - All the instances of an entity (no selector)
    - Only the instances of an entity that meet a selection condition (with selector)

Basic content units

- **DATAUNIT**
  - entity
  - [Selector]
- **INDEXUNIT**
  - entity
  - [Selector]
- **MULTIDATAUNIT**
  - entity
  - [Selector]
- **ENTRYUNIT**
  - entity
  - [Selector]
- **SCROLLERUNIT**
  - entity
  - [Selector]
- **MULTICHOICE**
  - entity
  - [Selector]
- **HIERARCHICAL**
  - entity
  - [Selector]
Basic content units

- **DATAUNIT**
  - **Author**
    - first name: XXX
    - last name: YYY
    - photo:

- **INDEXUNIT**
  - **Index of Authors**
    - Thomas Mann
    - Gunther Grass
    - Gerdi Weikum

- **MULTIDATAUNIT**
  - **All Authors**

- **ENTRYUNIT**
  - **Insert Your Data**
    - Fname
    - Lname

- **SCROLLERUNIT**
  - **Browse Authors**
    -_staff: prof

- **MULTICHOICE**
  - **Choose Authors**
    - 1. Web Applicat.
    - 2. Systems

- **HIERARCHICAL**

Unit input and output

- **IN**
  - unitX

- **OUT**
  - entity

  
  [selector (par 1, ..., parN)]

- **Each unit exposes input and output parameters**
- **Input is required to compute the unit itself**
- **Parameters pre-defined for the unit**
- **Other parameters required by the selector of the unit**
- **Output can be used to compute other unit(s) depending on the current unit**

Navigation: contextual links

- **source unit**
- **target unit**

  - A contextual link is an oriented connection between two units (source unit and target unit), normally rendered by means of anchors or submit buttons
  - Purpose of a contextual link:
    - Allowing the user to move from one place to another
    - Transporting information from one place to another (in the form of link parameters)
    - May activate a computation (see later)

Default link and selector parameters

- Whenever possible, link and selector parameters are inferred from the diagram and need not be explicitly specified
- Diagrams become simpler and more readable
- Example:

```
Author <--Book [Author2Book]
```

Transport links

- A transport link has a default context that is passed to the target unit immediately after the display of the source unit, without the user intervention
- The user cannot change the default context and therefore the link is not rendered with an anchor
Data and multidata units

- Data units
  - Publish information about one single instance

- Multidata units
  - Present multiple instances of an entity (set of objects)

Index and scroller unit

- Index unit:
  - Publish an index of elements (set of objects)
  - Output parameter: OID of the object selected by the user

- Scroller unit:
  - For browsing a set (block) of objects
  - Output parameter: the set of OIDs (possibly 1) of the current block of objects

Entry unit

- Unit for describing input forms that allow information submission by the user
- Content is shipped to other units via outgoing links (normally one), using link parameters
- Typically translated into HTML using the <form> tag and the associated submit button

Multichoice and hierarchical index Unit

- Multichoice units:
  - Publish indexes of elements (set of objects) among which the user to select one or more elements (with checkboxes)

- Hierarchical units:
  - Publish an index of elements, with entries organized hierarchically using entities connected by relationships
  - Allow the user to select one element from any level of the hierarchy

Rationale

DISPLAY UNITS
- Data vs Multidata units: show one vs show many
- Index vs Scroller units: choose a keyword vs scan a list
- Index variants:
  - Index unit for choosing one in a list
  - Multi-choice index unit for choosing many in a list
  - Hierarchical index unit for choosing one in a tree

ENTRY UNIT
- Provides the capability for data entry

WebML composition: scroller + data

- The entity is the same for the scroller and the data unit
WebML composition: entry + scroller + index

Paging the result of a search

Hypertexts in the large: siteview, area, page

- Siteview: a set of pages and/or areas forming a coherent view of the site. Multiple site views can be defined on the same data model for different users or publication media
- Area: a set of logically homogeneous pages
  - Examples: Sections of a portal: Sport, Music, HiTech, ...
  - Areas can be nested, so that sub-areas can be defined inside areas
- Page: a container of one or more pieces of information shown to the user at the same time

Example

Example of paged hypertext

Non contextual links

- A non contextual link is a link between pages
- No context (information) is transported

The user can browse from a page to another one via an anchor (e.g., >>books)
Each siteview must contain a page marked as "Home"

- Landmark pages: globally visible pages. The user can jump to them from everywhere in the site view
- Equivalent to a non contextual link implicitly defined from every other page in the site view to the landmark page

A site view is a set of pages and/or areas forming a coherent view of the site
- Multiple site views can be defined on the same data model
- Different site views can be published for different types of users and for different types of output devices
- Site views can be
  - Public: everyone can enter
  - Private: access control with password protection is enforced

Client's login and facility for sending e-mail to the bank

Client's view of his accounts and loan applications

Data Model = ER Model, with entities, relationships, attributes, and generalisations. Entities can have complex attributes.
- First Hypertext Model expressing navigations from entities to entities along relationships.
- PAGE-SCHEMES to publish multiple instances, and UNIQUE PAGE-SCHEMES to publish a given single page.
- AGGREGATIONS as concepts that aggregate entities or other aggregates; may have simple selectors
  - Typically associates page-schemes to entities or to aggregates.
- UNION NODES represent heterogeneous entities.
- LISTS represent INDEXES.
Other Approaches

STRUDEL [FFKLS98]
- **Data Model** = OSM (graph)
  - Nodes = identifiers or values
  - Name-labelled links
  - Objects are grouped in collections. Collections populate repositories.
  - Query Language = StruQL with a query part for binding nodes and arcs and a construction part for building new graphs (with primitives create, link and collect).
  - The query declaratively defines SITE GRAPHS with content which is either new or extracted from repositories.
  - HTML is produced by rendering the site graphs with generic templates
  - separation of content and navigation from presentation.

Weave [FLSY99, YFIV00]
- **Data Model**: underlying data is relational
- **Hypertext model**: labeled oriented graph
- Nodes belong to two categories:
  - Pages (internal nodes) -- grouped in collections
  - Data fragments (leaves)
- Conjunctive parameterized queries, e.g. Part(PartKey)
- Links connecting nodes:
  - Hyperlinks, or data fragment nesting (internal node -> internal node)
  - Containment of data fragments within a page (internal node -> leaf)

Global Parameters

- Global parameters model information stored globally or in the user session
- A context parameter is defined by:
  - Name
  - ID
  - Duration (User session or Application)
  - Value type: can be either:
    - A Printable value (integer, string, ...)
    - An Entity (thus, the parameter can assume an OID value of that entity)
  - Default value [optional]

Set and Get Units

- **SET**: allows to set the value of a parameter
- **GET**: allows to retrieve the value of a parameter

Bank application hypertext: data-intensive features (3)

- Client’s view of recent transaction in an account

Operation Units

- These units model operations (either built-in in WebML or customized)
- Each operation has:
  - input from one or more incoming links
    (one is a normal link, the others are transport links)
  - two kinds of output links
    - **OK** link if the operation completes correctly
    - **KO** link if the operation fails
Built-in operations

**CREATE**
-> Create unit

**DELETE**
-> Delete unit

**MODIFY**
-> Modify unit

**CONNECT**
-> Connect unit

**DISCONNECT**
-> Disconnect unit

In/out flow: create

value1 → attribute1
Nothing
OID of the new object

value2 → attribute2

In/out flow: modify

value1 → attribute2
OID(s) of the object(s) to modify

OID(s) of the modified object(s)

Bank application hypertext: data-intensive features (4)

- Employees register new clients or new accounts

Bank application hypertext: data-intensive features (5)

- Managers assign employees to manage accounts

II.3 Personalization
WEB personalization (one-to-one WEB delivery)

- Giving to each user a different view of the WEB, depending on:
  - Who is the user
  - Where is the user
  - Which device is being used
  - When the user takes places
  - How the user behaves
- Scope of action:
  - A different WEB site
  - A different page layout
  - Different items of information on a page

Modeling Personalization in WebML

- Personalization has three facets:
  - Access control: login/logout operations for user recognition
  - Site view assignment: based on the group the user belongs to, some site views are accessible (1 or more site view per Group)
  - Content customization: user- or group-dependent content assignment
- [Presentation can also be customized but we do not cover this aspect]

User / group model

- Each User can belong to one or more Groups (predefined entities in the structural model)
- Each user has one default Group
- Each group has one associated Siteview

Other user/group models

- BEA WebLogic:
  - Users
  - Roles and Organizations:
    - Several Users participate to a Role
    - A Role is defined within the context of an organization
    - An Organization is a context within which the behavior of several roles is defined
- Strudel, Weave, Araneus:
  - "User" may be defined as one or several ordinary entity (classes); no special modeling
  - Personalization=parameterizing

Login/ logout

- A site-view may contain a page allowing users to login
- Each secured site-view should allow users to logout
- Changing Role (i.e. group) dynamically is allowed

CurrentUser and CurrentGroup

- Each WebML project has two predefined global parameters:
  - CurrentUser: the OID of the currently logged User
  - CurrentGroup: the OID of the Group of the currently logged user
- Login and Logout operations automatically set / unset these two parameters
Page personalization (user-level)
- Personalization can be achieved with appropriate data design
- Hypertext can reflect structure, and thus provide personalization

After login, the CurrentUser is identified, thus the index shows user’s preferred articles.

Modeling personalization by active rules (recall VLDB 2000 in Cairo...)
- Event:
  - User's login
  - Access to given page (incl. home)
- Condition:
  - Query
- Action:
  - One-to-one delivery of content/presentation
  - Selective tracking of user’s behavior
- Available in many products, including DYNAMO AND I.SELL, BROADVISION, MICROSOFT SITE SERVER, IBM WEBSPHERE (supported by design tools and wizards)

Rules supported by Dynamo and I-sell
When users make access to ultra-resistant dresses, propose sport shoes.

Presentation is orthogonal
- Presentation styles are associated to units and pages; they can be built by means of specific tools.
- Essential for “impressing” end users.
- Not covered in this tutorial.

II.4 Presentation (sketch)
II.5 Integrating business processes into Web applications

**Sample business process**
- Bank clients require a loan from the bank
  - Supply personal information and desired loan type
- Bank employees must perform two checks on the loan application
  - Client’s employment (salary) history
  - Client’s savings history
  - Checks may be performed in parallel
- Bank managers use the result of checks to decide loan approval or refusal.

**Modeling processes**
- Many workflow languages/models
- Some are quite complex [AHK+02]
- Common basis: Workflow Management Coalition (WfMC)
  - Users
  - Activities
  - Sequence
  - Pre- and post- conditions
  - AND-split, AND-join
  - OR-split, OR-join
  - Loop

**Sample business processes to enforce via a Web application**

**Splitting a business process among several classes of users**

**Synchronization required by the sample business process**
- Bank client
  - Bank employee
  - Bank manager
- Record the new loan application.
### How to enforce workflow constraints into a Web application?

- Enforcement through a Web application:
  - *Implicit workflow control through hypertext links*
- Enforcement through a *data-intensive web application*:
  - Use the underlying database
    - *Implicitly* process encoding and control through workflow-related entities
    - *Explicitly* process encoding and control through workflow metadata
- Workflow management systems:
  - Workflow primitive constructs + variables.

### Workflow enforcement mechanisms in a data-intensive Web application

- **Implicit process control**
  - Hypertext links: control through Web interaction
  - Application data
- **Explicit process control**
  - Workflow metadata
II.5.1 Integrating business processes into Web applications:

Implicit process control through hypertext links and application data

Example: "Apply" activity in the client site view

Bank client

Client login page

Loan application page

Bank client site view

Provide
- Personal info
- Loan info

Loan info

Hypertext for registering application completion

Constraints through hypertext links: summary

- Simple and powerful for synchronizing the actions of a single user
  - B2C applications
  - Most on-line application procedures (graduate program in Stanford, French tax collection system)
- Coupled with some notion of link enabling
  - WebML: the content of an unit can be computed only when all required inputs are available
  - Strudel & others: very similar. Computations of data fragments are linked through parameter passing.

Implicit process control through application data

- Based on the presence of:
  - Some entity instance ("data passing")
  - Some relationship instance ("data connection")

- Based on the changing state of: an entity instance
  - State change translates to changing the value of an attribute ("entity state")

- Equivalent, sometimes alternative ways
  - Their applicability depends on the process and data model design.

Workflow constraints through data passing

- In some processes P:
  - For each activity \( A \in P \):
    - There exists an entity \( E_A \) in P's data model s.t.
      - An instance of \( E_A \) is created iff an instance of \( A \) is enabled
  - We say that \( A \) is an activity-isomorphic entity

Example:
Example of workflow control through data passing (1/3)

- In such processes, process control can be enforced using instances of the $E_A$'s.
- Instances of $E_A$'s act as Petri net tokens.

Example of workflow control through data passing (2/3)

- In such processes, process control can be enforced using instances of the $E_A$'s.
- Instances of $E_A$'s act as Petri net tokens.

Example of workflow control through data passing (3/3)

- In such processes, process control can be enforced using instances of the $E_A$'s.
- Instances of $E_A$'s act as Petri net tokens.

Other examples of workflows enforced by data passing

- Your favorite (XML) query processing demo
- Supply query
- Parse query
- Optimize query
- Run query

Workflow constraints through data connections

- In some processes $P$:
  - For each activity $A \in P$
  - There exists an entity $E_A$ in $P$'s data model s.t.
    - A user is connected to an instance of $E_A$ iff the user can start performing $A$ on the corresponding case

Workflow constraints through data connections to users

- Such processes can be implicitly controlled:
  - A user can start working on an activity iff she is connected to the corresponding entity instance
- Example:
Workflow constraints through data connections to users

- Such processes can be implicitly controlled:
  - A user can start working on an activity iff she is connected to the corresponding entity instance
- Example:

Other example of workflow enforced through data connection

- Order processing system: connection to an Order instance
- Example of process control through entity state

Workflow control through entity state

- In some processes \( P \)
- For each activity \( A \in P \)
  - There exists an entity \( E_A \) in \( P \)'s data model, an attribute \( \text{attr} \) of \( E_A \), and a value \( v_A \) of \( \text{attr} \), such that
    - Activity \( A \) is enabled when and only when \( \text{attr} \) has taken the value \( v_A \)
- If there is a single entity with the above property for all activities, we call it **case-isomorphic entity**
  - Annotating this entity = encoding case advancement

Example of process control through entity state

- Loan application modified:
  - Loan.application.status \( \in \{ "started", "checked", "approved", "rejected" \} \)
- User creates an Application
- After creation, Loan.application.status = "started"

An employee can start a check of an Application having status="started"
- At the end, Loan.application.Status="checked"
Example of process control through entity state

- Loan application modified:
  
  \[ \text{Application.status} \in \{ \text{"started", "checked", "approved", "rejected"} \} \]

- An employee can start an approval of an Application having status="checked"
- At the end, Application.Status="approved" or Application.Status="rejected"

Remarks on implicit workflow enforcement

- The possibility to implicitly enforce \( P \) depends on
  
  - The design of the process \( P \)
  - The hypertext design (enforcement by links)
  - The design of \( P \)'s data model (enforcement by data)

- For many processes, many sensible design choices are OK
- Different process fragments may be controllable differently

Implicit enforcement using combined methods: conference mgmt. system

Conference mgmt. system: the time dimension

- The conference management system is driven by numerous deadlines
- Every deadline is a time precondition (obvious semantics)
- In the following, we ignore time for simplicity
Implicit enforcement using combined methods: conference mgmt. system

Data passing

Data connection
Implicit enforcement using combined methods: conference mgmt. system

Discussion of implicit enforcement (1)
- Back to the loan application example:

Discussion of implicit enforcement (2)
- Cannot be fully enforced through links:
  - Several users

Discussion of implicit enforcement (3)
- Cannot be enforced through data passing:
  - No entity(ies) in the data model allows to correctly control the two parallel checks
  - If we use Application, parallelism is disabled

Discussion of implicit enforcement (4)
- Enforcement through entity state requires modifying the data model:
  - Add C1 and C2 attributes to Application, with values in: ("NotStarted", "Started", "Finished")

Discussion of implicit enforcement (5)
- Add C1 and C2 attributes to Application, with values in: ("NotStarted", "Started", "Finished") and C1="Finished" and C2="Finished"
Remarks on implicit workflow enforcement

- The possibility to implicitly enforce P depends on
  - The design of the process P
  - The hypertext design (enforcement by links)
  - The design of P’s data model (enforcement by data)
- For many processes, many sensible design choices are OK
- Different process fragments may be enforceable differently
- Generality unclear (complex expressions)
- Once encoded, the process model is less visible from the hypertext

Process management using a workflow meta-model

- The idea: encode a generic workflow meta-model in the Web application data model
- Case advancement is recorded in instances of the workflow meta-model
- Preconditions = logical expressions (queries) over the workflow meta-model

Complete data model for the bank loan example

Workflow meta-model for WebML

WebML workflow primitives for explicit workflow enforcement

- Start/end an activity
- Start/end a case (may be: success, or abort)
- These operations are macros - e.g., "StartActivity/StartCase":
  - Create a new Case, connect it to the given process
  - Create an ActivityInstance
  - Connect it to the activity of the given name
  - Connect it to the case
Illustration in WebML: loan application process

- Clients apply for a loan
- Bank employees perform two checks in parallel
- A manager approves or rejects the application

Generic hypertext for Apply

- Client site view

Complete hypertext for Apply

- What happens inside the Client site view

The "ready" check

- Built-in, meta-level predicate over a Case
- Show only the Cases for which it is correct to start now the activity "CheckSalary"
- Implementation: logical expression over the WF meta-model, derived from the Process structure:
  - There exists a completed "Apply" activity for the case and there exists no "CheckSalary" activity for the case
  - In general, automatic derivation of "ready" may include user-specified preconditions that are part of the process specification.

Generic hypertext for SalaryCheck

- Employee site view

Complete hypertext for SalaryCheck

- Employee site view
Remarks on explicit workflow enforcement

- The possibility to explicitly enforce a process gives the following advantages:
  - Separation of concerns (process design vs hypertext design)
  - Evolvability (can modify activities and processes separately)
  - Tracking and auditing (by looking at the meta-data)
  - More visibility of the process (through explicit process enactment operations)
  - Full generality (with potential for automatic generation of parts of the workflows and/or correctness checking)
- But meta-data management is an overhead.

Advantages of case selection through application data

- Most natural using:
  - Case-isomorphic entities
  - Activity-isomorphic entities
- Takes advantage of user intuition
  - “The form I have to fill”
  - “The application I have to check”
  - “The file on which I am working”
- Hypertext is normally simpler
- Example: Application is isomorphic to the case
Pull vs Push Styles

- Up to here, “pull work” style:
  - Ready and ReadyCase are in “pull work” style:
    - Users choose their cases, inspecting the past activities
  - "Push" style also possible (push data and/or work):
    - Push data: Manager assigns doc to Employee(s)
    - Push work: Manager assigns Translate activity to Employee(s) push data AND/OR push work

Inspecting the past vs. preparing the future

WebML Unit for Push style process control

- New WebML unit: Assign
  - Creates a new ActivityInstances
  - Connects it to the current case
  - [Connects to a given user]
  - [Connects Activity-isomorphic data to the ActivityInstance, and to the user]

Example: document translation workflow

Manager site view in Push style

Employee site view in Push style

Summary of Push style

- Useful and natural in circumstances when “the future can be foreseen”
  - When it is possible to decide:
    - Which activity will take place next
    - Who will perform it
  - When activity- or case-isomorphic entities exist

- Not always possible
  - When workflow evolution depends on other users’ future choices (next task is for the manager…)
  - In the absence of appropriate entities in the data model

II.6 Integrating Web services in a Web application
Web services: paradigm for interaction

- Based on XML messages
- Standards:
  - SOAP: XML messaging
  - WSDL: one-message or two-messages operations
  - UDDI: "yellow pages" of web services
  - WSCL: simple model for correlating messages into conversations (dialect of SOAP and WSDL)
  - WSFL, WSCI, BPE4WSL:
    - Specifying workflows of web services
      - Complex, structured conversations
      - Complex message correlation model
      - Roles, transaction properties, recovery, ...

Simple WSDL interactions

- Example: employees use a remote web service in for the savings check
  - Request message: client's savings history and loan reimbursement period
  - Response message: maximum loan amount with acceptable risk rate

WSDL: a standard for simple interactions

- Central component of a web service: operation = 1 or 2 messages

Synchronous vs. asynchronous two-message operations

- Two-message operations may be used
  - Synchronously: no action is taken between 1st and 2nd msg
  - Asynchronously: action is performed between the msgs

Extending WebML to support interactions with web services

- High-level specification
  - Data model: specific entities modeling the interaction with services
  - Hypertext model: primitives of the graphical language for extensions for handling services
- Run-time support for Web services
  - Ability of exchanging messages
  - Support for conversations
  - Transferring data between the underlying data model and messages (XML)

Default WebML data model for supporting Web services
New WebML constructs for supporting Web services

- One WebML operation for each usage of Web service operations
- Marks for operations that start, resp. end conversations

WebML service operations are macros

1. Create a new Conversation instance
2. Create a new Operation instance
   - Name="getMax", ...
3. Connect the Operation to the Conv. Instance
4. Compose parameters on the incoming links into XML messg
5. Send XML messg; block waiting for answer
6. Decompose XML answer
   - may involve manipulating the underlying data
7. Export selected items from the answer as parameters of the outgoing links

The savings check using a Web service: data model

- Scenario: the GetMaxAmount WS is called
  - Request: savings history of the applicant
  - From the Response, an Estimate instance is extracted

Hypertext for the savings check using a Web service

- Employee site view

News subscription using Web Services: data model

- Scenario:
  - Users subscribe to news by calling the one-way WS operation “Subscribe”
  - News updates arrive through a notification WS operation “Update”
- Data modeling: new Subscription and UpdateMsg entities
Hypertext for financial news subscription using Web services

- Client site view

Client summary page
Make financial news subscription
Read news

Financial news subscription page
Subscribe

Read financial updates page
All subs.

Web applications vs. Web services

- Web applications may include calls to Web services (seen)
- Web applications may also implement Web services
- Scenario: set up a WS for salary checks

Web services implemented by Web applications

- Bank application viewpoint: SalaryCheck is a WS (black box)
- SalaryCheck provider viewpoint: SalaryCheck is a Web application driven by
  - Financial experts
  - External msgs.
Web applications vs. Web services

- In general, any unit in WebML can be exported as a WS req-resp operation:
  - Input: all unit parameters
  - Output: default XML-ization of unit contents
- More interestingly, complex fragments of hypertext (processes involving users, data, WSs) can be wrapped as Web Services and exported further

Web Service Conversations

- Several Web Service calls responding to the same application needs
- Examples:
  - Trip schedule: sequence of calls to several services, one for each part of the trip
  - Subscription to news services: one subscription, several notifications
  - The workflow underlying a conversation can be arbitrarily complex
  - Equivalent names: WS “coreography”, WS “composition”

BPEL4WS: Emerging Web Service conversation standard

- Promoted by Microsoft, IBM, and BEA
- Born as convergence of previous standards, in particular:
  - XLANG (Microsoft), a block-structured language with basic control flow structures such as sequence, switch (for conditional routing), while (for looping), all (for parallel routing), and pick (for race conditions based on timing or external triggers.)
  - WSFL (IBM) - almost identical to the workflow language used by IBM’s MQ Series Workflow, allowing nested graphs (but acyclic, the only iteration supported is on one activity which is performed until exit conditions are met.)
- Specifically concerned with Web service composition

Two styles of specification in BPEL4WS

Graph Style (WSFL)

```
<flow>
  <sequence>
    <activityA>
    <activityB>
  </sequence>
</flow>
```

Activities in BPEL4WS are:
- Basic: normally the invocation of WSDL operations
- Structured: arbitrary compositions of basic and structured activities.

BPEL4WS and WebML

- All specifications in BPEL4WS can be expressed by means of WebML hypertexts, with few limitations:
  - all (for parallel routing) launching operations in parallel is not supported
  - pick (for race conditions) is only supported as a user choice or a Web service call but not as an internal mechanism such as a timer or a database trigger
- Therefore, Web Service composition can be expressed by means of workflow primitives
  - As illustrated in the tutorial, process control can be implicit or explicit
  - WF “case” = WS “conversation”

Summary

- Existing Web application design models include:
  - Data
  - Hypertexts
  - Personalization
  - (Presentation)
- Extended the WebML model to cover
  - Workflow (process design)
  - Web service usage and composition
- Paradigm for extension - minimal, carefully designed concepts:
  - Appropriate data modeling (WF and WS meta-models)
  - New primitives (WF and WS-specific)
References


FINALE:
How it works

WebRatio Site Development Studio

- WebRatio: Case tool and software architecture for model-driven Web development
- Design enviroment: GUI for editing the conceptual model of a Web application and produce specifications written in XML.
- Code generator: extensible XSL-based processor transforming XML specifications into page templates, business components, data queries and (optionally) DDL scripts for database creation.
- Run time support: MVC-2 application framework running on top of J2EE and MS .NET, based on generic parametric components configured using automatically generated XML descriptors.
WebRatio architecture

Data Design
Hypertext
Presentation
Design

XML
XSL
Data Mapping
XML

Automatic
code generation

Deployment
information

Third party
Presentation
tools

JSP templates
Deployment config files
Page & operation actions
XML descriptors

XSL style sheet library

Java class library
Tag library
XSL for XML descriptors

Web Ratio resources (papers, manuals, …)
Web modeling compendium & bibliography

www.webml.org

More information...

- www.webml.org
  WebML resources (papers, manuals, …)
  Web modeling compendium & bibliography

- www.webratio.com
  WebRatio can be downloaded for academic use