6. The Document Engineering Approach

DE + IA (INFO 243) - 11 February 2008

Bob Glushko

Plan for Today's Class

Modeling Methodologies
The Document Engineering Approach
What Modeling Methodologies or Approaches Might Specify

- Processes / Activities / Steps -- what to do and when to do it
- Artifacts -- the documents or other representation of the results of the processes / activities / steps; different parts or views of the overall model
- Meta-models -- models that specify the type of information to be recorded in the artifact
- Notations -- the presentational constructs used in the modeling artifacts
- Tools -- technology used to create the artifacts

The Document Engineering Approach
The D-O-C-U-M-E-N-T Checklist [1]

D -- data types and document types (paying special attention to the former when they are used across the latter as the "glue" to connect processes)

O -- organizational transactions and processes (the "business processes", described coarsely like "drop shipment" or precisely like "PIP 3A4")

C -- context (types of products or services, industry, geography, regulatory considerations -- the ebXML "context dimensions" described in section 8.2 of Document Engineering)

U -- user types and special user requirements (these are "people" user types)

The D-O-C-U-M-E-N-T Checklist [2]

M -- models, patterns, or standards that apply or that are needed

E -- enterprises and eco systems (e.g., trading communities, standards bodies, other frameworks that help scope the case study)

N -- the needs (business case) driving the enterprise(s)

T -- technology constraints and opportunities (legacy or interoperability concerns from existing technologies or implementations; new or improved processes or outcomes enabled by technology)
The primary purpose of modeling is to better understand some existing system or environment and its entities and to describe this understanding so it can be communicated.

This collective or composite understanding is the domain model.
Most domains or systems to be analyzed/built are too big or complex to be understood "all at once" -- multiple modeling perspectives are implied by different parts of the model, and these might require more than one analyst with different skills.
The Modeling Artifacts

So the steps of the modeling methodology develop the domain model "in pieces" -- the modeling artifacts -- and the methodology is designed so that these pieces logically fit together to create the complete model.

The Modeling Artifacts in Document Engineering [1]

<table>
<thead>
<tr>
<th>Phase</th>
<th>Artifact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzing the Context</td>
<td>UML use case diagrams</td>
</tr>
<tr>
<td>Analyzing/Designing Business Processes</td>
<td>Business Domain View Worksheet</td>
</tr>
<tr>
<td></td>
<td>UML use case diagrams</td>
</tr>
<tr>
<td>Analyzing/Designing Business Collaborations</td>
<td>Business Process Area Worksheet</td>
</tr>
<tr>
<td></td>
<td>UML activity diagrams</td>
</tr>
<tr>
<td>Analyzing/Designing Business Transactions</td>
<td>Business Transaction View Worksheet</td>
</tr>
<tr>
<td></td>
<td>UML sequence diagrams</td>
</tr>
<tr>
<td>Applying Patterns to Business Processes</td>
<td>Document checklist</td>
</tr>
</tbody>
</table>
Meta-models

A meta-model is an abstract model that specifies the type of information to be collected and recorded during the modeling activity.

Rigorous or highly-formal methodologies generally have a meta-model with prescriptive processes for "populating" it.

Meta-models enforce consistency in a set of models, enabling them to be compared or to interoperate.
The ebXML Process Metamodel

Notations

A notation is needed to depict or represent the objects and processes in your model.

At its simplest, a notation is a set of graphical elements (usually boxes) and lines that connect two or more of them to indicate a relationship.

There is no single "lingua franca" model notation suitable for all users and purposes.
UML as a Modeling Notation

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing and documenting the structure and behavior of (software) systems.

UML is the synthesis of a variety of object-oriented modeling concepts and notations and is now endorsed by the Object Management Group. It is a language in that it can be used to define different types of models.

UML has a number of standard notations or diagram types that all are based on the same underlying meta-model (and it can also be used to define additional notations).

The primary use of UML in document engineering is for describing models of documents and business processes.

UML Class Diagram for "Wonton Soup"
XML as a Modeling Notation

XML has rapidly emerged as a preferred format for creating models that are vendor and notation-neutral.

XML's key benefits from this perspective are the ease with which XML instances can be transformed and the use of models expressed as XML schemas to guide code generation by serving as templates for program objects.

Using the concepts introduced in this lecture, XML is a meta-modeling language or maybe even a meta-meta-modeling language.

Each XML schema language is a meta-language for creating models that are expressed using it.

Conceptual Model of "Book" Encoded as XML Schema

```xml
...  
<xs:element name="Book">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Title"/>
      <xs:element ref="Authors"/>  
      <xs:element ref="ISBN"/>
      <xs:element ref="Publisher"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Authors">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Author" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
But Using XML != "Modeling with XML"

Some people think that "modeling" with XML means "writing a schema given a set of instances" or "inferring a schema from a single instance"-- many software tools support this sort of thing.

But schemas developed without a stage of conceptual design are rarely very useful because they are too closely tied to the particular instances used, which may not be representative.

Sometimes schemas went through a stage of conceptual design but once the schemas are implemented the conceptual information isn't available to schema users.

Tools

Tools can be as simple and informal (pencil and paper) or complex and formal (CAD and CASE software).

There is no single tool suitable for all users and purposes.

Tools can be general purpose, not tied to specific methodologies (word processors, spreadsheet, presentation packages).

General purpose tools can use templates to support specific notations or processes.

Tools can be dedicated to particular notations or processes.
Starting Through the "Snake"

Setting the Context

Any Document Engineering project worth doing will involve some set of document types and information components that take part in some set of business processes.

Because "no document (or process) is an island" there will always be some point at which the documents and processes you care about will intersect or overlap with some that you don't care about.

We'll call the CONTEXT whatever characteristics of the situation define what is in or out of scope, inside or outside of the boundary in which our solution has to work.
Context is a Point of View

Context and Selecting Patterns

A business process pattern implies a set of documents and some regular choreographies of document exchanges.

A pattern can be thought of as a *typical cluster or configuration of requirements*.

Selecting an appropriate pattern will help expose the information requirements, rules and constraints for our subsequent document analysis and design.

Choosing a pattern suggests which document payloads we'll need to find or design and in which business processes we are likely to deploy them.

How we describe context influences what patterns we identify and how we apply them.
Document Analysis

The first phase in developing conceptual models of documents is *Document Analysis*

"Document" analysis is the standard term but is a bit of a misnomer – much of what we will analyze isn't packaged as "documents" and looks more like "data" – we are going to stretch the meaning of "document" to fit everything

Document-level activities of *Inventory* and *Sampling* determine what documents and information sources we'll analyze in more detail

The *Harvesting* and *Consolidation* activities work on the smaller content components identified or extracted in the preceding activities

After consolidation to a set of semantically distinct candidate components, we begin the design phase
Example Harvest: Syllabus

<table>
<thead>
<tr>
<th>Announcement</th>
<th>Text</th>
<th>the content of an announcement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnnouncementDate</td>
<td>gDate</td>
<td>the date on which an announcement is posted</td>
</tr>
<tr>
<td>AssignmentDueDate</td>
<td>gDate</td>
<td>the date on which an assignment is due</td>
</tr>
<tr>
<td>AssignmentLink</td>
<td>URL</td>
<td>a link to additional information about an assignment</td>
</tr>
<tr>
<td>ClassDate</td>
<td>gDate</td>
<td>the day on which class is held</td>
</tr>
<tr>
<td>ClassEvent</td>
<td>String</td>
<td>something special that’s happening on that class day (quiz, exam, assignment due, etc.)</td>
</tr>
<tr>
<td>ClassLocater</td>
<td>String</td>
<td>the initials of the instructor who is teaching for a particular class</td>
</tr>
<tr>
<td>ClassmainLink</td>
<td>URL</td>
<td>a link to the index for a class</td>
</tr>
<tr>
<td>ClassSlidesPrintableLink</td>
<td>URL</td>
<td>a link to a printable set of slides for a class</td>
</tr>
<tr>
<td>ClassTitle</td>
<td>String</td>
<td>the title of a particular day of class</td>
</tr>
<tr>
<td>CourseMeetingBuildingName</td>
<td>String</td>
<td>the building in which class meets</td>
</tr>
<tr>
<td>CourseMeetingDay</td>
<td>Enumeration</td>
<td>a day on which class meets (Monday, Tuesday, Wednesday, Thursday, Friday)</td>
</tr>
<tr>
<td>CourseMeetingEndTime</td>
<td>gTime</td>
<td>the time at which a class ends on a particular day</td>
</tr>
<tr>
<td>CourseMeetingRoomNumber</td>
<td>String</td>
<td>the room number in which class meets</td>
</tr>
<tr>
<td>CourseMeetingStartTime</td>
<td>gTime</td>
<td>the time at which a class starts on a particular day</td>
</tr>
<tr>
<td>CourseName</td>
<td>String</td>
<td>the name of the course</td>
</tr>
<tr>
<td>CourseNumber</td>
<td>String</td>
<td>the alphanumeric designation of the course</td>
</tr>
<tr>
<td>CourseOverview</td>
<td>Text</td>
<td>a description of the course</td>
</tr>
<tr>
<td>DepartmentName</td>
<td>String</td>
<td>the name of the department in which the class is listed</td>
</tr>
<tr>
<td>EnrollmentLimitations</td>
<td>String</td>
<td>a description of the people who may enroll in this course</td>
</tr>
<tr>
<td>GradingNotes</td>
<td>Text</td>
<td>extra information about grades</td>
</tr>
<tr>
<td>GradePercentages</td>
<td>Number</td>
<td>the percent which a given grade is assigned towards the final grade</td>
</tr>
<tr>
<td>HomeUpdatedBy</td>
<td>String</td>
<td>the initials of the person who updated the home page of the class website</td>
</tr>
<tr>
<td>HomeUpdatedDate</td>
<td>gDate</td>
<td>the date on which the course home page was last updated</td>
</tr>
<tr>
<td>Instructor</td>
<td>PersonName</td>
<td>the name of the instructor</td>
</tr>
</tbody>
</table>

Example Consolidation: Event Calendar

<table>
<thead>
<tr>
<th>Name</th>
<th>Semantic Description</th>
<th>Source 1</th>
<th>Source 2</th>
<th>Source 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>The title of the event</td>
<td>X</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Start Date</td>
<td>The date of the event, or the first date of a recurring event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>End Date</td>
<td>The last date of the event</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>The location of the event</td>
<td>X</td>
<td>(merged with synonym Venue)</td>
<td>x</td>
</tr>
<tr>
<td>Speaker</td>
<td>Name(s) of the person(s) speaking at the event</td>
<td>X</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>The description of the event</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Speaker Title</td>
<td>The title of the speaker</td>
<td>x</td>
<td></td>
<td>(renamed homonym Title)</td>
</tr>
</tbody>
</table>
Document Component Models

Document Assembly Models
Another Assembly of Same Component Model
Implementing Models in Document Engineering

Many of the important patterns for Document Engineering are used when encoding assembled document models as XML schemas.

This is a two-stage process: encoding assembled document models as physical ones, and then applying formatting or style transformations to create instances with desired properties.

When instances implemented in different technologies are generated or re-generated from models, they can more readily interoperate because of their common conceptual components.

Model-Based Applications

Creating an information or process model is a significant investment in capturing context-specific (or application-specific) requirements in a technology-neutral and robust way.

The abstraction in a good model makes it simpler and easier to work with than the specific technologies of implementation.

The model can be viewed as a specification for generating code or configuring an application.

Ideally, the context-specific parts of the application that are based on the model(s) remain distinct and inspectable apart from the generic functionality provided by the "platform" on which it implemented.
A Disclaimer

XML+XSL+CSS is well suited for a large class of document-centric applications where browser-based deployment is essential because of administrative and security concerns.

XML is also optimal for transactional applications, especially where the transactions are mostly carried out by computers with only exceptions being handled by people.

But not all applications can or should fit into a native or exclusive XML framework – not every application is entirely document-centric, a publication or a form.

Using XML Specifications to Design and Drive Applications [1]

XML's is both a cause and effect of significant changes in how we think about the Web and distributed applications.

The idea of "document" has significantly expanded while becoming more unified around XML.

Business information in databases and other enterprise applications can now be externalized as XML documents, giving a common view to content aggregation, management and distribution.

The XML documents (publications or forms) used by the application are composed in "building block" fashion from reusable semantic components, which facilitates reuse and integration.
Using XML Specifications to Design and Drive Applications [3]

XML's value as both a source and target format means that it is often produced by converting non-XML information; and XML documents are often transformed to meet the requirements of other contexts, programs, or devices.

The idea of "document type" has inspired a programming paradigm is emerging in which XML schemas, programming language classes, database schemas, and UML models can be treated as equivalent.

XML schemas are increasingly being used to generate or configure application code, especially for user interfaces.

Readings for 13 February

Chapter 3 of Document Engineering [Textbook, 86-100]

Do Some Business Models Perform Better than Others?, Peter Weill, Thomas W. Malone, Victoria T. D'Urso, George Herman and Stephanie Woerner