20. Model-based Applications and User Interfaces

INFO 202 - 5 November 2008

Bob Glushko

Plan for INFO Lecture #20

Model-based Applications and Integration

Platforms for Model-based Applications

Model-based User Interfaces across the Document Type Spectrum

XForms and beyond

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

A MODEL-BASED APPLICATION explicitly uses a model 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 of the application provided by the "platform" on which is it implemented

Benefits of Model-Based Applications

It is easier to understand the software architecture

Generated code is of higher quality than hand-crafted code

Easier to maintain (regeneration when model changes)

How Else Could You Do It?

Why would you design and implement an application without using the information and process models inherent in the design context?

Many applications "flatten" and "deconstruct" a hierarchical document model into a relational one or into "attribute-value" pairs

But there is little separation of information models and the code that handles them in most scripting languages

The iterative, heuristic, and non-deterministic techniques in most user interface design methodologies don't emphasize the information and process models, or develop them incrementally

Reminder: The Integration Requirement

Companies have so many internal (with employees) and external (with customers and suppliers) interactions that they must automate as many as possible

This requires INTEGRATION -- the controlled and automated sharing of content, data and business processes among any services, applications, or information sources, intra- or inter-company

Integration has long been a substantial portion of the IT activities in many companies

Not Model-Based Integration

The "old" way to integrate two applications within an enterprise was to write a custom program that fit only between the two of them

This method has traditionally been called A2A (Application to Application) integration

The technical approach is usually file transfer or remote procedure call mechanisms

The low-level granularity of APIs means this tightly-coupled connection isn't model-based

Model-Based and XML-Based Integration [1]

An emerging integration philosophy and methodology explicitly uses the models on each side of the integration

The idea of "document type" has inspired a programming paradigm in which XML schemas, programming language classes, database schemas, and UML models can be treated as equivalent (because they can be created from each other via transformation)

OMG's "Model-Driven Architecture"
uses UML models of the objects or components to generate integration code

More generically, "Service Oriented Architecture" wants to treat business software functionality as "components" whose interfaces and processes are described using XML schemas that follow the web services standards

Model-Based and XML-Based Integration [2]

The XML models used by applications are composed in "building block" fashion from reusable semantic components

This component architecture facilitates the reuse of information between models and integration between the applications that produce and consume them

Custom views of information for different users, devices, or context can be created by rendering the same XML document with different transforms

Platforms for Model-Based Applications

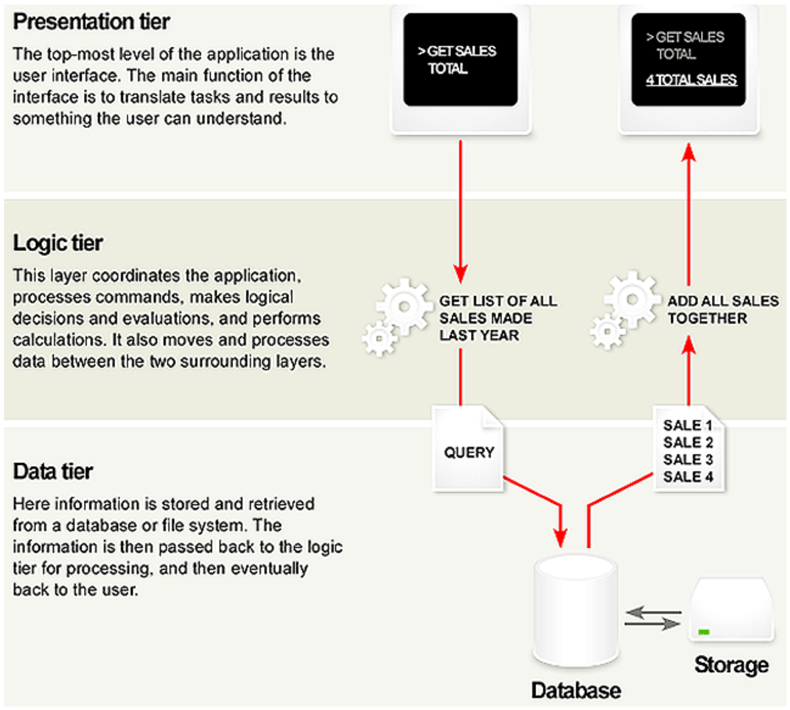
A software platform solves some class of generic problems so that application developers can focus on the context-specific parts

Operating systems, programming languages and their runtime libraries, databases, software frameworks, middleware "service bus," web browsers ... are are platforms...

Ideally, the context-specific parts of the solution that are based on the model(s) remain distinct and inspectable apart from the generic functionality provided by the platform

In this case, the platform interprets the model to determine how the software behaves -- the model configures or customizes the platform

One Minute Detour: 3-Tier Architecture


Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.
>GET SALES TOTAL
>GET SALES TOTAL
4 TOTAL SALES
Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
GET LIST OF ALL SALES MADE LAST YEAR
ADD ALL SALES TOGETHER
QUERY
SALE 1
SALE 2
SALE 3
SALE 4
Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
Database
Storage

History of the "Web Platform"

The earliest UIs were character-based transactional command languages running on terminals hard-wired to mainframes or minicomputers

The PC enabled the development of "thick" graphical UIs like Windows

Early web browsers had far less UI capability, but this "thin" platform vastly reduced application support requirements

CGI, CSS and HTML forms were introduced for more interactivity, layout control, and transactional content

Java applets, Flash, DHTML, Java/VBScript, Swing etc. emerged as (often proprietary) alternatives to standard HTML

Can You Be Too Rich or Too Thin or Too Standard?

These so-called "rich" or "smart" client application platforms are also "thick" because they require substantial client-side software

Should you rely on the standard and ubiquitous/universal client of the browser, or take on the problem of installing the "thick" client into the browser?

The goal is "rich" capability in a "thin" client -- or finding ways to eliminate the problem of installing and managing the required plug-in

Can we be rich in a standard, model-based way?

Today's Browser Platforms for Web Applications

Current web browsers are platforms for model-based applications; the browser knows what to do with instances of various document models expressed in HTML, XHTML, or XML

XForms browser "plug in" generates forms from XML models, enabling enable client-side validation and XML data interchange

Ruby on Rails embodies AJAX concepts (asynchronous Javascript, eliminating full-page refresh and enabling much finer-grained interactivity) and uses a non-XML data model to generate an extensible set of HTML forms and controllers (a “scaffold”)

Other platforms are vendor proprietary (Adobe Flash and Flex, Microsoft Silverlight) but others are open (Firefox XULRunner)

Model-Based User Interfaces

User interface design started as a distinct activity in the 1980s, and has been dominated by iterative and heuristic techniques ever since despite efforts to create a discipline of Information Architecture

In the 1990s the goal of model-based UIs emerged with the hope that "automatic generation of window and menu layouts from information already present in the application data model can relieve the application designer of unnecessary work while providing an opportunity to automatically apply style rules to the interface design"

- de Baar, Foley, & Mullet, "Coupling application design and user interface design," Proceedings of CHI'92. http://doi.acm.org/10.1145/142750.142806

Some people starting calling this the search for the "Big Red Button,", and in many cases it involved user interface modeling languages (expressed in XML) from which UIs would be generated

Meeting in the Middle

The strongest proponents of MBUI are computer scientists who are comfortable with abstract models and techniques for code generation

Many MBUI proponents work in application contexts like mobile computing where the UI presentation repertoire is limited

Opponents of the MBUI approach argue that it de-emphasizes usability concerns and undermines the creative aspect of UI design

Many Small Red Buttons?

How can we "meet in the middle" to build UIs more efficiently with more predictable quality in UIs without eliminating creativity?

An alternative to the search for the BRB is the goal of partial automation for user interface generation:

- Tools that generate prototypes from specifications
- Tools that synthesize use cases into sequence diagrams
- Tools that merge sequence diagrams to hide states that have no UI implications
- Tools that generate UI skeletons or scaffolds while enforcing layout constraints
- Tools that generate a family of UIs via "graceful degradation" or "content adaptation"
- UI Design Patterns

XML Vocabularies for Describing User Interfaces

Many XML vocabularies for describing user interfaces have been developed

- XUL -- Used by the Mozilla browser rendering engine called Gecko
- XAML - similar approach by Microsoft
- MXML - in Macromedia / Adobe Flash

Unfortunately, these three XML vocabularies describe UIs at the presentation layer, not at the information model layer, so they fall short of the vision of MBUI

UIML - academic effort, most abstract and extensible... but no commercial adoption

One Minute XUL

The elements of the XUL vocabulary include standard user interface components like menus, input controls, dialogs and tree controls, and keyboard shortcuts

Customizable "skins" in Mozilla are different stylesheets applied to the XUL components

XUL "Periodic Table"
at http://www.hevanet.com/acorbin/xul/top.xul) is excellent reference

```
<toolbar id="a-toolbar" >
      <label value="This is a toolbar:" />
      <toolbarseparator />
      <toolbarbutton label="Button" accesskey="B" oncommand="alert('Ouch!');"/>
      <toolbarbutton label="Check" type="checkbox" />
      <toolbarbutton label="Disabled" disabled="true"/>
      <toolbarbutton label="Image" image="images/betty_boop.xbm"/>
    </toolbar>
```

Model-Based UIs and the Document Type Spectrum

Many platforms (and vocabularies) for model-based applications are form-based and targeted toward transactional document types

"E-book" readers are a notable exception, designed as platforms for narrative document types to provide "book-like" display and interactive functionality

"E-Books" and Publications with Dynamic Structure

The simplest case of structured publication is publishing a single document in a way that lets the user interact with it by exploiting its (content) component structure

This can easily be done in an ordinary web browser by building a dynamic "table of contents" with links at different hierarchical levels to the information components

E-book readers differ in how much structural and presentation fidelity they enable and in functionality that "goes beyond the printed book"

Richer functionality in user interfaces for "book-like" documents include structure-based search, "viewspecs" for filtering on component types, visualizations or simulations of content, and enhanced navigation

Many e-book platforms exist, and many have adopted XML formats (native or interchange) with Open eBook
(but the Amazon Kindle has a proprietary non-XML format)

Designing for Multiple Platforms

Multi-platform User Interfaces

Why do some applications or services need to run on multiple platforms?

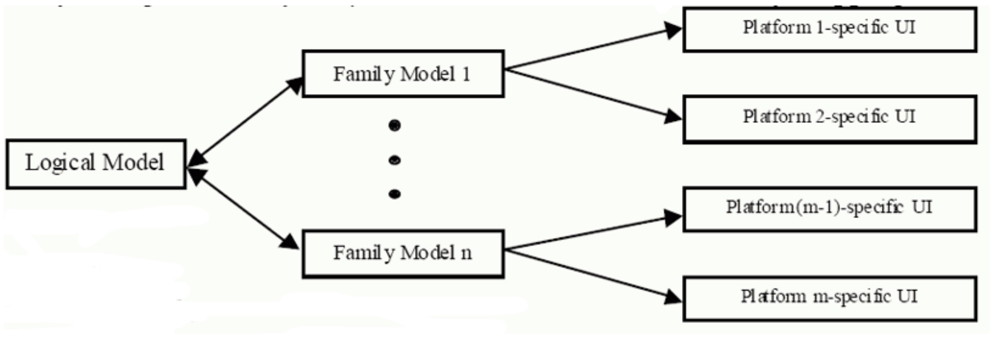
These platforms vary in screen size, resolution, input methods... which collectively determine the repertoire of feasible user interface controls

What are the costs and benefits of designing separate UIs for each platform vs a "design once and adapt" approach?

The tailoring or adaptation to each platform can be done at "design time" or at "run time"

Models that describe the capabilities of "device families" are essential, especially when devices proliferate and have short useful lives (like cell phones)

"Device Families" and UIs



Adapting Content to Multiple Devices

It is often necessary to adapt the content along with the UI when an application needs to run on a family of devices or platforms

This content adaptation can also be done in advance (design time) or on demand (run time)

Multimedia content poses severe challenges, especially because of the need for "graceful degradation"

E-Forms

True model-based UI approaches are most promising for E-Form applications (especially those using XForms, a W3C specification)

Countless applications and services use a "fill-in-the-web-form" paradigm to automate processes that previously relied on printed forms

Filling out a form is creating a valid instance of the document type, and often the application is little more than "Webifying" a document interface to a legacy printed or client-server document application

(For all but the simplest forms, however, complexity arises in the mapping of the logical model to the set of interactions needed to collect the instance)

XForms

Unlike HTML forms, in XForms there is a separation between the conceptual model that defines the information being collected and the presentation model that defines the form controls and appearance

This has many advantages ("Why XForms" paper):

- "Multiple environments" - the same model can be used in a Web form, a voice-driven form, or a printed form
- "Multiple devices" & "Accessibility" - no assumptions about the device on which the form will be rendered
- "Machine use and automation" - forms as strongly-typed service interfaces

But Wait, There's More

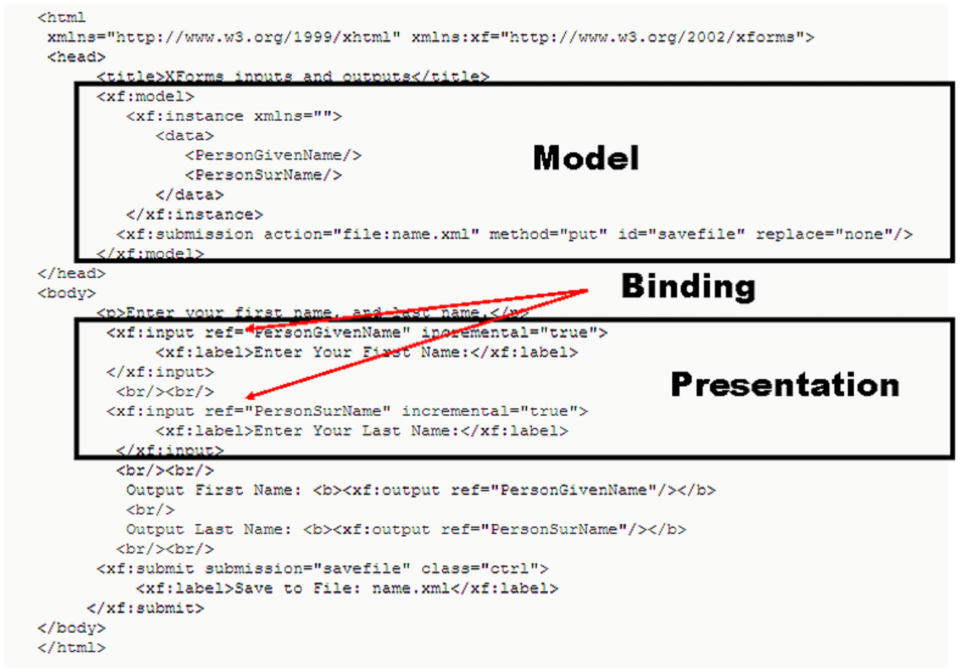
"Input validation"

"Avoid round trips"

"Internationalization and localization"

no longer will it be necessary to hope
"that no one will ever mention crêpes flambées or aïoli,
no one will have a name like Antonín Dvořák, Søren Kierkegaard, Stéphane Mallarmé or Chloë Jones,
and no one will live in Óbidos or Århus, in Kromìøíž or Øster Vrå, Průhonice or Nagykõrös, Dalasÿsla, Kırkağaç or Köln."

XForms Example



```
<html
 xmlns="http://www.w3.org/1999/xhtml" xmlns:xf="http://www.w3.org/2002/xforms">
 <head>
      <title>XForms inputs and outputs</title>
      <xf:model>
         <xf:instance xmlns="">
            <data>
               <PersonGivenName/>
               <PersonSurName/>
            </data>
         </xf:instance>
        <xf:submission action="file:name.xml" method="put" id="savefile" replace="none"/>
      </xf:model>
</head>
<body>
      <p>Enter your first name, and last name.</p>
       <xf:input ref="PersonGivenName" incremental="true">
            <xf:label>Enter Your First Name:</xf:label>
       </xf:input>
        <br/><br/>
       <xf:input ref="PersonSurName" incremental="true">
            <xf:label>Enter Your Last Name:</xf:label>
        </xf:input>
        <br/><br/>
         Output First Name: <b><xf:output ref="PersonGivenName"/></b>
         <br/>
         Output Last Name: <b><xf:output ref="PersonSurName"/></b>
        <br/><br/>
      <xf:submit submission="savefile" class="ctrl">
          <xf:label>Save to File: name.xml</xf:label>
     </xf:submit>
</body>
</html>
```

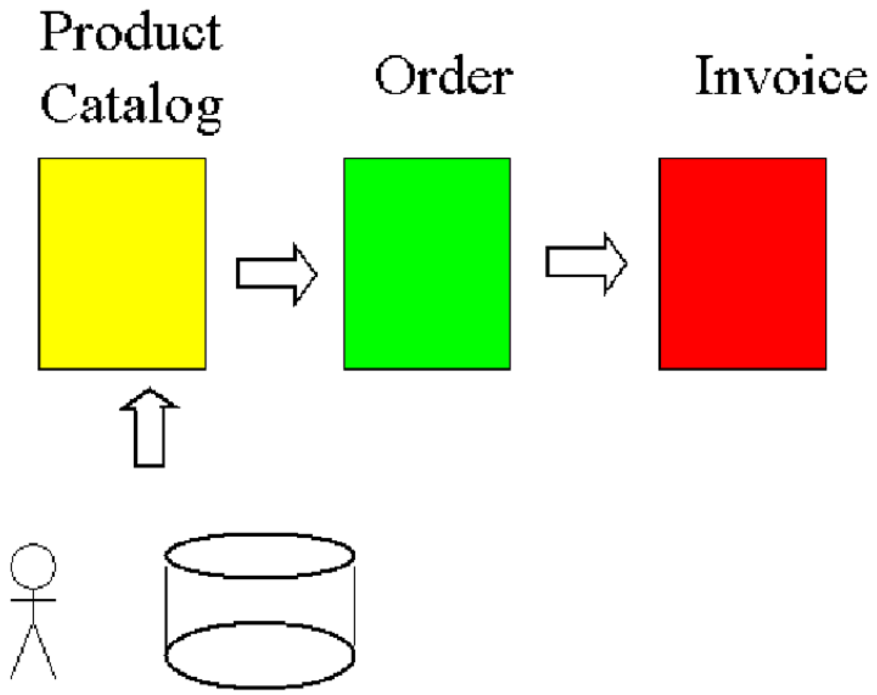
Some XForms Resources

XForms Recommendation from the W3C

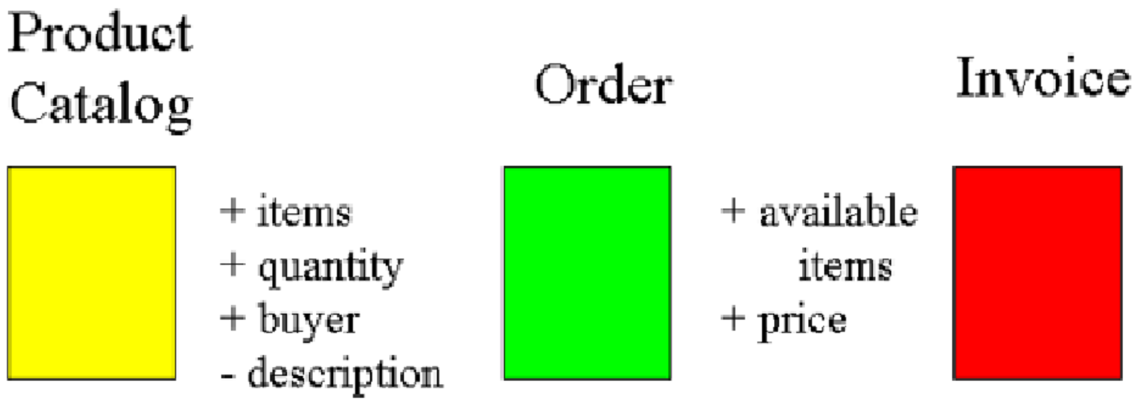
XForms if you know HTML forms

Example on previous slide (use "View Source" to see it)

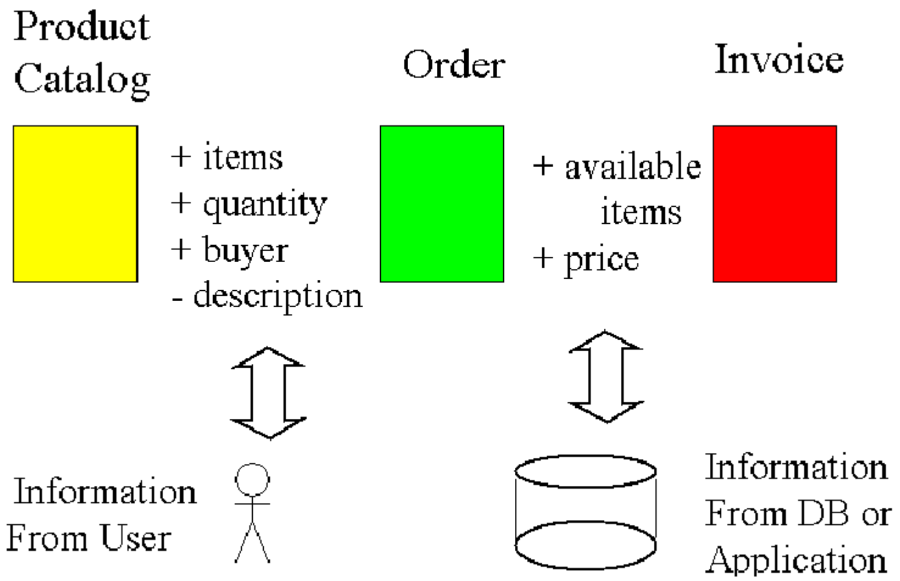
Forms and Workflow Applications



Model Components Reused in Transactions



Component-based User or Application Interfaces



Business Rules in User Interfaces

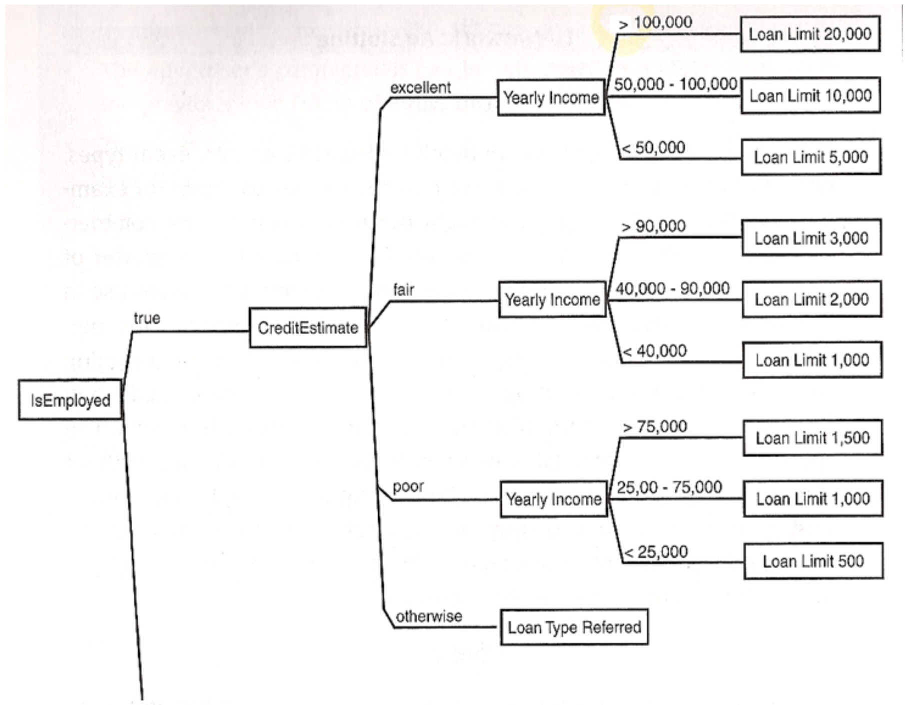
XForms can validate input based on the built in XSD datatypes and on other types of rules that XSD can express

But more complex types of rules, constraints, and "decision logic" are needed in "information-intensive" industries

"Business rules" usually refers to these more complex types of assessments, especially when they are carried out by "rules engines" and "decision services" so that the rules can be more efficiently created, maintained, and reused

Fair Isaac (of FICO credit score fame) claims that its Blaze Advisor is the "world's leading business rules management system"

Business Rules Depicted as "Decision Tree"



"Active Document" Applications and Platforms

"The document is the application"

Information retrieval, data acquisition, transactions, workflow, or archiving processes are embedded in the document

The document serves as the user interface to the multiple applications that are related to the information

The document structure and appearance can be changed dynamically for different users, contexts, or uses

"Active Document" Platforms

IBM Lotus Forms (formerly "Workplace Forms")

- Uses XFDL, extension of XForms to support enterprise application requirements like multi-page forms, attachments, security, auditability

Adobe Intelligent Document Platform

- "Leveraging the richness of PDF with the power of XML"
- XFA (XML forms architecture) is dictated by an XML data structure inside a PDF file

Justsystems xfy

Readings for Lecture #20

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, Introduction to Information Retrieval, Sections 19.4, 8.5.1-8.7

Marc Resnick and Misha Vaughan, "Best Practices and Future Visions for Search User Interfaces"

Anne Aula, Studying User Strategies and Characteristics for Developing Web Search Interfaces, Chapter 4