Reusability Analysis for Shipbuilding Components Modeled in XML and Java

Prof. Steven A. Demurjian, Sr.
Jeffrey R. Ellis
Computer Science & Engineering Department
The University of Connecticut

Prof. Donald M. Needham
Computer Science Department
U.S. Naval Academy

steve@engr.uconn.edu
JRE95001@mail.snet.net
needham@usna.edu
http://www.engr.uconn.edu/~steve
(860) 486 – 4818

Overview

Layering and Its Impact on Reusability (SD)
- Can we Constrain Reusable Components?
- How are Reusable Dependencies Assessed?

Linking Resources in XML (DN)
- How are XML Documents Lined?
- What is the Impact of Links on Reuse?

Distributed Object Computing Using SOAP (SD)
- What is the Technology?
- Is it Relevant for Reuse? Overall EB Work?

Prototyping Progress in DRE and TCC (SD & JE)
- Status of CSE300 Project
- Work by JE on Parser and Other Capabilities
- Planned Work for Upcoming Months
Layering and Reusability (SD)

- Problem Domain Layering
  - Motivation from October 2000 Meeting
- Recasting - Transition from
  - Levels of Generality
  - Layers of Generality
- Issues to Resolve:
  - What are Limitations Across Layers?
  - How are Dependencies Tracked?
  - Can There be Different Layered Perspectives for “Same” Design?
    - Can a Class be G in One Layer and S in Another?
    - What are Implications on Reuse in this Case?

Problem Domain Layering
Discussion from October 2000 Meeting

Notion of General/Specific Impacted by Layering?

<table>
<thead>
<tr>
<th>Layer</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common</td>
<td>Person</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>Part</td>
</tr>
<tr>
<td>Shipbuilding</td>
<td>Not OK Piping-Part</td>
</tr>
</tbody>
</table>
Recall Levels of Generality

- **Root classes** for Items, ItemDB, etc., which are Most General. (G^0)
- **Inventory Control/Other Components.** (G^1)
- **Classes Specific to Grocery Store Domain.** (G^2)

**Specific Applications for Big Y or Shaw’s or Stop/Shop (S)**

- Dependencies Encouraged Up Through the Layers
  - From S to G(0) or G(1)

Transition to Layers of Generality

- **Root classes for Items, ItemDB, etc., which are Most General.** (G^0)
- **Inventory Control/Other Components.** (G^1)
- **Classes Specific to Grocery Store Domain.** (G^2)

- **Big Y or Shaw’s or Stop/Shop (S)**

- **Limit Interactions Among Layers**
  - Only Adjacent Layer Allowed
  - Impose Strong Reuse Penalty Otherwise

- **Issues**
  - Can Layer Contain Multiple Levels of G?
  - Red Class with G(1), G(2), G(3)
  - Within Level, Established Reuse Principles Apply

- Can There be Layers within Layers?
### Another Example of Levels of Generality

<table>
<thead>
<tr>
<th>Root classes for Items, ItemDB, Inventory Control Other Components.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classes for Large Supermarket</strong></td>
</tr>
<tr>
<td><strong>Classes for Specialty Supermarket</strong></td>
</tr>
<tr>
<td><strong>Classes for 24 Hour Convenience</strong></td>
</tr>
<tr>
<td><strong>Big Y or Shaw’s or Stop/Shop</strong></td>
</tr>
<tr>
<td>Assume: Each Layer Can Contain Multiple Levels</td>
</tr>
<tr>
<td>What Happens When Different Groupings within Layers?</td>
</tr>
<tr>
<td>Can Groups Interact?</td>
</tr>
<tr>
<td>What are Dependency Issues?</td>
</tr>
<tr>
<td>Do We have Lattice Rather Than a Layers and Levels?</td>
</tr>
</tbody>
</table>

### Recall the Different Abstractions

<table>
<thead>
<tr>
<th>Classes and Hierarchies</th>
<th>Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building Blocks of Application</td>
<td>Organize Classes and/or Hierarchies into <strong>Named</strong> Conceptual Units</td>
</tr>
<tr>
<td>Independent Classes of All Types</td>
<td></td>
</tr>
<tr>
<td>Generalization and Specialization</td>
<td></td>
</tr>
<tr>
<td>Interfaces</td>
<td>Components</td>
</tr>
<tr>
<td>“Specification” of Behavior</td>
<td>Application-Oriented</td>
</tr>
<tr>
<td>Never Instantiated</td>
<td>Logical and Physical Associations of Classes, Packages</td>
</tr>
</tbody>
</table>

**Note:** We Must Reconcile “Our” and EB Abstraction Definitions
Abstractions and Layering
How are Layers Organized?

Issues for Layering and Reusability

- What are Limitations Across Layers?
- How are Dependencies Tracked?
- Can There be Different Layered Perspectives for “Same” Design?
  - Can a Class be G in One Layer and S in Another?
  - What are Implications on Reuse in this Case?
- Can All Layerings Coexist Together?
- Are Layerings Application Specific?
- Can we Develop a Model for Levels and Layers?
  - Inter-Layer vs. Intra-Layer Reuse
  - Inter-Level vs. Intra-Level Reuse
Linking Resources in XML (DN)

Creation/Description of Links Between Resources
- Associate Metadata With a Link
- Express Links that Reside in a Location
- Separate from the Linked Resources
- Runtime Traversal via Link Base

Virtual Relationship? (Impact on Reuse?)

Doc1
Link Base
DocA
DocC
DocB

XML Linking Language (XLink) Ver 1.0

XLink Allows XML Documents to:
- Assert Linking Relationships Among More Than Two Resources
- Associate Metadata With a Link
- Express Links That Reside in a Location Separate From the Linked Resources
- XML Provides Linking Data Structures

XLink Provides a Minimal Link Behavior Model

Higher-level Applications Layered on XLink May Specify More Sophisticated Rendering and Processing Treatments
What is an XML Link?

- An *Explicit* Relationship Between Resources or Portions of Resources
- Made Explicit by an XLink Linking Element Asserting the Existence of a Link
- When a Link Associates a Set of Resources, Those Resources Are Said to **Participate** in the Link
  - Such Links Are Able to Associate All Kinds of Resources, Not Just XML-Encoded Ones

Extended Links Can Associate Remote Resources

- Local to Remote Resource A
- Local to Remote Resource B
- Local to Remote Resource C

Extended Link Associates 3 Resources
Link Arcs, Traversal, and Behavior

**Traversal:** Using or Following a Link.
- **Starting Resource:** The Source From Which Traversal Is Begun
- **Ending Resource:** The Destination

**Arc:** Information About Traversing a Pair of Resources, Including Traversal Direction, Application Behavior Information

Extended Links

Offer 3rd-Party Arcs and Links With Arbitrary Num of Participating Resources. Can Include Elements for:
- Pointing to Remote Resources,
- Containing Local Resources,
- Specifying Arc Traversal Rules

**XLink** Provides Semantics for Finding Linkbases
- Documents Containing Collections of Inbound and Third-party Links Are Called Link Databases, or **Linkbases**.
Extended Links Can Associate Resources and Provide Traversal Rules

Extended Link Associates 3 Resources
Provides 2 Traversal Rules

XLink Element Type

The Type Attribute Indicates the Element Type:

- Dictates the XLink-Imposed Constraints That Such an Element Must Follow and
- The Behavior of XLink Applications on Encountering the Element
Traversing Extended Links

An Extended Link May Indicate Rules for Traversing Among Its Participating Resources by Means of a Series of Optional Arc Elements.

- The Behavior Attributes Specify the Desired Behavior for XLink Applications to Use When Traversing to the Ending Resource.

- Example: a Resource Might Represent a Person but in the Context of One Arc It Might Have the Role of Mother and in the Context of a 2nd Arc It Might Have the Role of Daughter

Looking Ahead

Do Arcs or Type, Semantic and Behavior Attributes provide Static Typing Info Usable in Aiding Reuse Evaluation of XML Designs?

- Spice (Extended Java-Script) is Used to Specify XML Document Behavior. Does Spice Play a Role in our Reuse Focus?
- How Does Dynamic Behavior of XML Linkbases Impact XML Reuse?
- What Other XML Abstractions/Dependencies Exist that Impact Reuse?
Distributed Object Computing Using SOAP (SD)

SOAP Stands for Simple Object Access Protocol

SOAP is a Cross-Platform Approach for

- Remote Method Calls
- Serialize and De-serialize Objects Using XML

SOAP is Hardware, OS, and Platform Independent

- http://www.w3.org/TR/SOAP/

Supported by Apache SOAP V 2.0

- Java Library for Both Client and Server
- Application Need Not Parse nor Create XML
- Server Implemented As a Servlet
- Client SOAP Calls Easy to Construct Using API

What is SOAP?

Lightweight Protocol for Info. Exchange of in a
Decentralized, Distributed Environment

XML Based Protocol That Consists Of:

- Envelope as a Framework for Describing
  What’s in Message/How to Process It
- Set of Encoding Rules for Expressing Instances
  of Application-defined Datatypes
- Convention for Representing Remote Procedure Calls and Responses

SOAP Works Over HTTP - Advantages

- Unlike Other Middleware, Doesn’t Require
  Changes to Network Routers & Proxy Servers
- Usage of HTTP Which is a Universally
  Deployed Protocol
SOAP Process

SOAP Request Would Be Processed in the Following Steps:

1. Get a Request on the Listen Port
2. Parse the Request for the Method Id to Call
3. Consult a Configuration File for What Class/function to Call to Handle the Request
4. De-serialize the Parameters for the Method Call
5. Call the Function With the Given De-serialized Parameters
6. Serialize the Return Value From the Function and Send It Back to the Requestor

SOAP is Simple to Understand, Implement and Deploy

Basic SOAP Sample

Here is a SOAP Request:

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Basic SOAP Sample

... and the Matching Response is:

HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: mmm

<SOAP-ENV:Envelope
"SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/
"></SOAP-ENV:Body

<Price>34.5</Price>
</m:GetLastTradePriceResponse
</SOAP-ENV:Body
</SOAP-ENV:Envelope

SOAP Conclusions/Potential

SOAP Being Universally Utilized in Industry
  IBM, Microsoft, Compaq etc.
SOAP Easy to Utilize and Understand
SOAP Standard Still Evolving
Performance Definitely Poor Compared to RMI
What is Potential Role of Soap in Our Project?
  Can Soap be Used to Exchange XML Documents?
  Can Reusable XML Components be Modeled in SOAP for Effective Exchange?
  EB: Is There a Role for SOAP in Overall Process?
**Summary of Prototyping Milestones**

- Results and Status of CSE300 Project
  - Work by H. Lin and X. Wang on DRE
  - Work by S. Moore on TCC
- Work by JE on Parser and Other Capabilities
- Focus of Effort on DRE
  - Redesign GUI of DRE
  - A More User-friendly Interface
  - Parser Independent OO Modeling
- Focus of Togethersoft Control Center
  - Utilization for Reusability Analysis
  - Design and Partial Implementation Via Plugins

**Shore Up Java Support**

- Current Parser is C++ Upgrade
- Add Java-Specific Constructs
  - Interface
  - Package
  - Java Bean (Component)
  - Inner Class (Invisible to Outside World)
- Modularize Parser
  - Allow For Future Replacement
- New Common Data Structures
  - Provide Communication Ability Across Modules
Summary of Prototyping Milestones
Metrics Scheme & Calculation

- Translation to New Data Structures
- Remove All GUI Traces
- Incorporate Multiple Levels of Generality
  - Original Research Presented General vs. Specific
  - Later Research Expanded to Spectrum (G0, G1, … Gx/Specific)
- Incorporate Package/Interface/Component Measurement Ability
  - Extension of Price Research Required

Summary of Prototyping Milestones
Graphical User Interface Upgrades

- Professional-Looking Design
  - Previous GUI Was Experiment in Swing
  - Stability, Appearance, Modular
- Allow Flexibility in User Selections
  - Workspace to Allow Full Source Code Measurement
  - Selectively Choose Java Files From Multiple Directories
  - Coupling Analysis Granularity
- Integrated Help System
  - Intuitive Interface
  - Detailed Instructions
  - Theory Explanation
  - Interpretation of Metrics Discussion
- Ease of Use
Interface Design

GUI Subsystem Layout Structure

Use JTabbedPane to Display the Metrics Calculation Result and Provide the Ability for Future Simulation Work

Interface Design

Framework for On-Line Help Facility

HTML-Based OO-Reuse, DRE Metrics, and Tool
OO Modeling in DRE

Four Major Components in the System

- GUI
- Data Manager
- Metrics Calculator
- Parser

Data Abstraction

Nodedata Encapsulates the Properties & Methods to Present a Tree Node, and the Id Field to Entity Object
Data Abstraction

Entity is for package/class/variable/method/innerclass

Property is for Characteristics of selected node

Future Work - Online Help System

Future Work

- Online Help System
Review of DRE’s Fall 2000 Milestones

**Data Structures - Completed**
- JE & CSE300
- September-October 2000

**Java Parser - Completed**
- JE
- October-November 2000

**Metrics Scheme & Calculator - In Progress**
- JE
- November-December 2000

**GUI - Almost Completed by H. Lin and X. Wang**
- CSE300
- October-December 2000 (-January 2001 JE)

Review of DRE’s Fall 2000 Milestones

**Help System**
- Framework Completed by H. Lin and X. Wang
- November-December 2000 (-January 2001 JE)

**Simulation Tool**
- JE
- December 2000-February 2001 (-March 2001)

**Together Control Center Application**
- CSE300 & JE
- September-December 2000 (-March 2001)

**Characterization File**
- JE
- December 2000
Plans for DRE in Spring 2001

Continued Effort by Jeff Ellis
- Review and Revise CSE300 Prototype
- Fine-Tune and Complete Baseline Prototype
- Planning for Layering and Levels of Generality
  - What are the Key Reuse Concepts?
  - How are Dependencies Tracked?
  - How Does Dependency Analysis have to Change?

CSE367 Projects
- DRE
  - Two to Three Student Semester Project
  - H. Lin and at Least One Other
  - Focus on Help and Other Bells/Whistles
- Together CC - To be Discussed Shortly

Reuse in Togethersoft Control Center
CSE300 Project by Steven Moore

Integrates Visual Modeling via IDE
- Implements UML/Supports Java and C++
- Source Code Editing
- Model/Code Synchronization
- Open API and Plugin Support (Written in Java)

Single Integrated Design/Development Environment with Reusability Assessment

Leverage TCC to Support Reusability Calculations
- Perform Calculations With Minimal User Interaction
- Have Calculations Stay With Project
- Display Results in Standardized Fashion
- Give User Quick Feedback From Changes
QA in TCC and plugins

- TCC Supports Quality Assurance Via Audit Analyses and a Set of Source Code Metrics
- Allows Customization Via User-definable QA Plugins
  - Audit Plugins
    - Source Code Syntax Checking
    - Displays Qualitative Results in a Report Pane
  - Metrics Plugins
    - Source Code Counting Operations
    - Displays Quantitative Results in a Report Pane

QA Plugin Hierarchy

- Utilize by Overriding TCC’s Metricspluginimpl Class
  - Plugins are Loaded When TCC is First Started
  - User Selects “Custom” Metrics Which Automatically Invokes Plugin
    - Prepare() Method
      - Executed Once When Execution Started
    - Run() Method Executed for Each Class and Package in Project
Status to Date/Future Plans

- Developed 10 Plugins - Partial Implementation
  - 1 Module to Specify G/S Classifications
  - 1 Module to Specify R/U Relationships
  - 8 Coupling Count Modules
- Spring 2001 Independent Study with S. Moore
  - Plugins Can be Any Java Software
  - Looking at Reusing DRE Code in Plugins
    - Exploit Revised Data Classes/Organization
    - Employ Java Parser
  - Seek Other Plugins to Modify UML Diagrams to Collect Reuse Dependencies Across Tool
  - Fully Integrate Reuse into TCC

Concluding Remarks/Looking Ahead

- Progress and Future Direction
  - Are we on Track?
  - What is Next Major Steps?
- Long-Term Objectives (1-2 Years Out)
  - Planning for XML Reusability?
  - R. Caballero Ph.D. Work
  - Other Research Opportunities in Reuse
- Sabbatical Leave for S. Demurjian - Approved
  - September 2001 to May 2002
  - Looking for Opportunities to “Work” in Industry