Reusability Analysis for Shipbuilding Components Modeled in XML and Java

Prof. Steven A. Demurjian, Sr.,
Jeffrey R. Ellis, Rodrigo Caballero,
Hai Lin, and Xiaopei Wang
Computer Science & Engineering Department
The University of Connecticut

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

steve@engr.uconn.edu
jre95001@yahoo.com
needham@usna.edu
http://www.engr.uconn.edu/~steve
(860) 486 – 4818

Overview

- Reusability Model, Layers, and Framework (SD)
  - Can we Constrain Reusable Components?
  - Can we Define a Reuse Model with Layers?
- Reusability and XML (DN)
  - Impact of XML Abstractions on Reuse
  - Linking Resources in XML
- Prototyping Progress in DRE and TCC (JE & SD)
  - Status of CSE367 Project
  - Work by JE on Parser and Other Capabilities
  - Planned Work for Upcoming Months
- Reusability and Genetic Algorithms (RC & SD)
June 2001. OOReuse - 3

Reusability Model, Layers, and Framework (SD)

- Application is Designed with Abstractions (Class, Interface, Package, Component)
- Apply the Reuse Framework to the Classes, Interfaces, Packages, and Components
  - Assign Generality Levels
  - Determine Related Classes, Components
  - Iterate Until Obtain Marking that is Consistent with the Goals/Objectives of Application
- Problem: Is it Possible to Organize or Compartmentalize the Application’s Classes, Interfaces, Packages, and Components From Different Perspectives Based on Requirements?
  - Compartments - Previously Called Layers
  - Represent Chunks of Functionality

June 2001. OOReuse - 4

Reuse Model Issues from Last Meeting

- Clarify/Redefine “Related” Characterization
  - Transitivity of “Related”
  - W R to A, B, and C vs. W R to A, B, or C
- Common Definitions for Package/Component
  - How is Marking of Package/Component Determined?
    - “Least” General Class Sets Package/Component Generality
  - Packages/Components Must Obey All Generality Rules of the Model/Framework
- Inclusion of Interfaces into Model/Framework
- Design Patterns and Impact on Reuse
- Layering and the Reuse Model (Our Focus)
### Problem Domain Layering Discussion from October 2000 Meeting

<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>Piping-Part</td>
</tr>
<tr>
<td>Company-Specific</td>
<td>Sub-Piping-Part</td>
</tr>
</tbody>
</table>

### Recall the Different Abstractions

<table>
<thead>
<tr>
<th>Classes and Hierarchies</th>
<th>Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>q Building Blocks of Application</td>
<td>q Organize Classes and/or Hierarchies into Named Conceptual Units</td>
</tr>
<tr>
<td>q Independent Classes of All Types</td>
<td>m Components</td>
</tr>
<tr>
<td>q Generalization and Specialization</td>
<td>q Application-Oriented</td>
</tr>
<tr>
<td>Interfaces</td>
<td>q Logical and Physical Associations of Interfaces, Classes, Packages</td>
</tr>
<tr>
<td>q “Specification” of Behavior</td>
<td>q Purpose: Represent a “Meaningful” Chunk of Functionality</td>
</tr>
<tr>
<td>q Never Instantiated</td>
<td>q This is a Design Time Component (e.g., a Non-Instantiated Bean)!</td>
</tr>
</tbody>
</table>

Note: We Must Reconcile “Our” and EB Abstraction Definitions
Abstractions and Compartments

Some Sample Compartments?

Proposed Problem Solution

- Define “Reusability Node” to Represent the Different Compartments of Application’s Classes, Interfaces, Packages, and Components
- Specify the Rules for:
  - Composition of a Reusability Node
  - Dependencies Among Reusability Nodes
- With these Rules, Result is a Reusability Graph of Reusability Nodes with Reusability Dependencies
- Reusability Graph can be Layered
- Objectives:
  - Formally Define Reusability Node, Dependency, Graph, and Layer
  - Can we Automate the Process When Given Source Code for Application?
Definition 1: Reusability Node

A Reusability Node (RN) is a Set of Classes (Cl), Packages (P), Interfaces (I), and Components (Co) Marked With Varying Levels of Generality

Entities in RN Must Satisfy Current Reuse Analysis w.r.t. General/Specific/Related, Coupling Counts, etc.

\[ \mathrm{RN}_x \] as Shown on Right

Classes, Packages, Interfaces, Components are Labeled as follows:

\[ \mathrm{G}_{x_i} \text{ for } i = 1 \ldots n_x \]

are \( n_x \) Generality levels

Assumptions: Reusability Node

Is the Makeup and Content of a RN Constrained?

What are Possible Constraints on \( \mathrm{RN}_x \) ?

Levels of Generality

\( \$ \) At Most 3 Different Generality Levels

Replication

\( \$ \) Classes in Exactly One RN

\( \$ \) Can Packages/Components be in Multiple RNs?

\( \$ \) Interfaces Can be in One or More Different RNs

\( \$ \) Do Components/Packages have Generality Level?
Objective: Reusability Node

Given Source Code (Classes, Interfaces, Packages, and Components) for Application, is it Possible to **Dynamically/Automatically** Determine all RNs?

What is Needed for this to Occur?

- Each Class, Interface, Package, Component Must be Marked with Generality Level
- Related Dependencies for Reuse are Defined
- Parser that Synthesizes All of the Source Code Dependencies

Is this Problem NP Hard? Complete?

Aside: Marking Components/Packages

A Component/Package can Contain Components, Classes, and Interfaces

- Remember, Class CL1 can be Defined in Package B and Used in Component X
- Or, Component X can be Used in Package B

Components/Packages Must Run Through Reuse Metrics/Analysis to Eliminate Bad Dependencies

Package not Marked Since Utilize Portions of the Package and Not Package Itself

Components not Marked or Assumed Most General
Definition 2: Reusability Dependency

A Reusability Dependency (RD) is a directed arrow between two Reusability Nodes, RN\textsubscript{x} and RN\textsubscript{y}, which satisfies the following:

\begin{itemize}
  \item $G_{x_i}$ for $i = 1 \ldots n_x < G_{y_j}$ for $j = 1 \ldots n_y$
  \item RN\textsubscript{x} is Most General, RN\textsubscript{y} is Least General
  \item Arrow from RN\textsubscript{y} to RN\textsubscript{x}
\end{itemize}

Issues:

\begin{itemize}
  \item Should it be $\leq$ Rather than $<$?
    \begin{itemize}
      \item $<$ is Preferable, Since it Allows for a Cleaner Ordering Among Different RNs
      \item Eliminates Ambiguity in Ordering
    \end{itemize}
  \item Should Interfaces be Excluded from Ordering?
    \begin{itemize}
      \item If Interfaces Replicated, Same Generality Level?
      \item If Interfaces Non-Replicated, Include in Ordering?
      \item Must Support Replicated Interfaces since Different Classes can Implement Same Interface Differently
    \end{itemize}
  \item Proposal regarding Interfaces
    \begin{itemize}
      \item Interfaces are Not Marked
      \item When Class “Implements” Interface, the Marking of the Interface is the Same as the Marking of Class
      \item Hence, Interface Can have Different Markings in Different Classes
    \end{itemize}
\end{itemize}
Definition 2: Reusability Dependency

Issues:

- Should the Dependencies Between $G_{x_i}$ and $G_{y_j}$ be Limited Such That $G_{x_i}$ Can Only be Dependent on $G_{y_{n_j}}$?
  - Yes, this Constrains the Constructive Requirements of the Reusability Graph (See Defn. 3)
  - This Constraint with Assumptions of RN (Slide 5) May Assist in Automated Determination of RDs
  - If Both RNs and RDs can be Automated, from Source Code, Create Reusability Graph (Defn. 3)
- That is, Highest Level of Generality in $\text{RN}_y$ Can only Dependent on Lowest Level in $\text{RN}_x$

Quantifying a Reusability Dependency

Suppose $\text{RN}_x$ has Generality Levels of: $G_0, G_1, G_2$

Suppose $\text{RN}_y$ has Generality Levels of: $G_3, G_4$

Should Dependencies (Class, Method, Attr, etc.) be Limited to from $G_3$ in $\text{RN}_y$ to $G_2$ in $\text{RN}_x$?

Or, are Limits Unnecessary?
Definition 3: Reusability Graph

A Reusability Graph (RG) is a Directed Graph with Reusability Nodes (Defn. 1) as Vertices which are Connected by Reusability Dependencies (Defn. 2)

Issue: Are Cycles Allowed?

Issues: Reusability Graph

Cycles:
- A Cycle Indicates Co-Dependency Among Multiple Reusability Nodes - D, E, and H
- Does this Imply All Must be Reused Together?

Automation
- Assume that Reusability Nodes can be Automatically (Algorithmically) Determined When Given Application’s Source Code
- Can Reusability Graph be Automatically Derived? Is this NP Hard? Complete?
- Can we Use Genetic Algorithms for Problem?
Definition 4: Reusability Layer

A Reusability Layer (RL) is a subset of the RG that represents a grouping of RNs to satisfy a particular domain or organization requirement.

Issue: What is Scope of a Layer?

Since RN_A depends on RN_D which depends on RN_E which depends on RN_H which depends on RN_B, should all form the layer?
## Issues: Reusability Layer

**Cycles:**
- Recall a Cycle Means Co-Dependency Among Multiple Reusability Nodes - D, E, and H
- If All Must be Reused Together then …
  - Will this Impact Layering - Must Layer Always Contain all Nodes in Cycle?
  - Can Layer Contain only Part of Cycle?

**Automation**
- Intuitively, Automating Layers Doesn’t Make Sense
- Layer is Domain Specific and Dictated by Designer

---

## Future Work

**Issues for Layering and Reusability**

- What are Limitations Across Layers? **None??**
- How are Dependencies Tracked?
  - Defn. of RN and RD
- Can There be Different Layered Perspectives for “Same” Design? Can All Layerings Coexist Together? Are Layerings Application Specific?
  - Yes to all Three - As shown on Previous Slide?
- What are Issues to Develop a Reuse Model for Levels and Layers?
  - Formal Model and Analysis
  - Inter-Layer vs. Intra-Layer Reuse
  - Inter-Level vs. Intra-Level Reuse
- Is there an EB Example to Demonstrate Ideas?
Reusability and XML (DN)

June 2001 *IEEE Computer*(p.66): “XML Leaves Unresolved How to Best Specify Intersystem Mappings”

- “Analysts need Tools to record Intersystem Relationships and Mappings to Community Standard DTDs or Schemas and to make them Available for Reuse”

XML Abstractions Impacting Reuse

- SOAP, Extending XML Datatypes
- Linking Resources in XML as applied to Reuse
  - Extended Links
  - Traversal Rules

XML, OO, and Reuse

Our Approach to Analyzing Reuse Examines Communication Between OO Classes

Simple Object Access Protocol (SOAP) Messages use XML Documents to Allow Internet Communication Between Applications

- Can SOAP Messages Form the Basis of our Reuse-Focused Communication Analysis, or
- Does Another Aspect of XML Meet our Communication Analysis Interests?
**The Nature of XML Data Types**

- Currently, An XML Complex Type May be Extended Only Through Appending
- Will Emerging XSDL Support *Semantic* Inheritance?
  - Negative Overriding/Partial Inheritance
    - Allows Designer to Inherit some Properties, and Suppress Other Properties
  - Is Semantic Inheritance Critical to our Reuse Interests?

**Linking Resources in XML**

- 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
What is an XML Link?

An XML Link is 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

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: Associate Resources and Provide Traversal Rules

- Local to Remote ResourceA
- Local to Remote ResourceB
- Local to Remote ResourceC

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 the XLink Application upon Encountering the Element

Does such Behavior play a Role similar to Method Invocation in terms of Evaluating Reuse?
Future Work
XML Reuse Considerations

- Do Arcs, Type, Semantic or Behavior Attributes provide Static Typing Info Usable in Aiding Reuse Evaluation of XML Designs?
  - Spice (Extended Java-Script) has been 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?

Prototyping Progress in DRE and TCC (SD & JE)

- Summary of Prototyping Milestones
  - Results and Status of CSE367 Project
    - Work by H. Lin on DRE
    - Work by X. Wang on TCC
  - DRE Status
    - GUI Finalized
    - Simulator Completed
    - Parser Integrated
    - Help Subsystem Completed
  - Togethersoft Control Center Status
    - Developed Extensive Set of Plug-ins
    - UML Design Level Reusability Definition
### Schedule *(Spring 2001)*

#### Data Structures
- JE & Team
- September-October 2000
- *January-February 2001*

#### Java Parser
- Team
- October-November 2000
- *February-May 2001*

#### Metrics Scheme & Calculator
- JE
- November-December 2000
- *March 2001*

#### GUI
- Team
- October-December 2000
- *February-May 2001*

### Schedule *(Spring 2001)*

#### Help System
- Team & JE
- December 2000
- *March-April 2001*

#### Simulation Tool
- JE
- *February-May 2001*

#### Together Control Center Application
- Team & JE
- September-December 2000
- *February-May 2001*

#### Characterization File
- JE
- January 2001
- March 2001
Main Application Window

Main Window – Setting Generality
Main Window – Setting Relations

Main Window – Choosing Simulation Options
Help Subsystem

HTML-based Documents Describe Tool & Theory

Using DRE v2.01

Click on a topic or scroll down to view the help information.

- Adding Files to the Project
- Setting Ownership of Objects
- Setting Relation/Bag/Referent/Instance
- Viewing Ownership/Relationship Information
- Change the Number of Levels of Ownership
- Duplicate the Reusability Measure
- Java for Current Project
- Open a Saved Project

Adding Files to the Project

The DRE will parse selected Java files and present a list of classes contained within. The user begins by instructing the DRE as to the specific files to be parsed. Simply click on the blue plus icon on the toolbar. A file-open dialog box will be presented. Locate the desired files and click on “OK.” The parsed Java classes will appear in the left explorer pane.

Reuse in Togethersoft Control Center

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 at Design Level

- Incorporate G/S and Related at UML Level
- Users can Execute Plug Ins for Each
- Actual Properties for UML Classes Modified
Reusability in Together CC
**Future Work**

Prototyping, Together CC, and XML

- Version Releases
  - v2.00 – December 2000 (Unreleased)
  - v2.01 – April 9, 2001
  - v2.02 – June 12, 2001

**Future Work**

- Continue TCC Research
- Implement Recommendations
  - Usability Requests
  - Feedback
  - C++?
- DRE Research
  - Displaying Coupling Counts
- Masters’ Thesis – XML and Reuse?

**Reusability and Genetic Algorithms Objectives (RC & SD)**

- Design and Implement a New DRE Parser Using Java Compiler Compiler Parser Generator Tool
  - Use of the Java Grammar Specification Allows Easier Maintenance of DRE Parser
  - Improvement of Current Parser: Isolation of Parser and Data Structures
- Use a Genetic Algorithm to Find a Near Optimal Class Generality Marking
  - Can Genetic Algorithm be Utilized to Automate Determining Marking?
  - Class Marking That Minimizes a Function of the Coupling Counts
  - User Presented “Actual” Generality Level of Classes Based on Cross-References in Code
Java Compiler Compiler

**Javacc** is a Parser Generator Tool
- Compiles a Grammar Specification into Java
- Augments Grammar Specification with "Semantic Actions - Java Code"
- Resulting Program Recognizes Java Source Code and Generates Java Code That Creates DRE Data Structures Objects

```
Java Grammar
```

```
JavaCC
```

```
DRE Parser
```

```
DRE Parser .java
```

```
DRE Compiler Compiler
```

```
Population
```

```
Problem: Current GA with Limited Assumptions
```

Genetic Algorithm

**Stochastic Search Method Inspired in Natural Evolution**
- Population-based Random Variation/Selection
- Applied to Data Structures with Goal/Initialization

**Characteristics:**
- Genetic Operators (Selection, Crossover, Mutation)
- Population of Chromosomes (Bit Strings)
- Encode Candidate Solutions in Search Space
- Selection Based on the Fitness of Each Chromosome

**Problem:** Current GA with Limited Assumptions Tends to Yield all General Classes!
Near Optimal Class Generality Marking

Genetic Algorithm and Reuse

Towards Guided Software Reusability Assessment
Multi-Step Iterative Process:

- Determine Set of Assumptions Regarding Generality
  Levels of Generality that are Required
  Related Implied by Dependencies or Set by User
- Key Issue is to Define Constraints that Will Underlie the GA Fitness Function
- Run the Genetic Algorithm on Code for Fixed Population and Number of Generations
- Evaluation of Results - Potential to Rerun with Changed Assumptions, Population, Generations
- Converge Towards “Reasonable” Marking
Future Work
Genetic Algorithm and RNs, RDs, RG

Current Independent Study - Rodrigo Caballero
Investigating GA Work and Ph.D. Research

- Examining Related Research in Refactoring, Reuse Metrics, etc.
- Conducting Experiments using GA Algorithm regarding Assumptions, Fitness Functions, etc.
- Considering Role of RNs and RDs in Arriving at “Realistic” and “Meaningful” Marking
- Towards “Guided Reusability Assessment”
  - Allow SWE Input re. Assumptions, Generality, Related, etc.
  - Apply GA Algorithms Based on SWE Input
  - Provide Integrated Environment to Guide SWE

Future Work - Reusability - Summary

Short Term (By 9/2001)
- Prototype (DRE/TCC) - JE and XW
- GA Work - RC
- Initial Formal Reuse Model - SD
- XML/Reuse - DN/JE
- Journal Paper (All)

Long Term (By 5/2002)
- Completed Prototypes
- Progress on GA Work
- Formal Reuse Model
- MS Thesis on XML and Reuse
- Conference and 2nd Journal Articles

Research - Through 2004
- Focus on Comprehensive Reuse Framework
- Guided Reusability Assessment/GA Analysis
- Component Repository with Semantic Search Capabilities
- Leverage Domain/Organization Specificity
- UML, XML, and Java Components