Requirements to UML to Engineering Drawing Traceability Mechanisms

Cari Bever
Nick Oben
November 28, 2006
Overview

- Problem Statement
- Shortcomings of Present Day Tools
- Proposed Approach
- Project Scope and Objectives
- Software Architecture
- Example
Problem Statement

- Often, the underlying cause of catastrophic and expensive failures is minor mistakes or omission in communication of design intent.
- Almost all grave software problems can be traced back to conceptual mistakes made before the programming was initiated.
Shortcomings of Present Day Tools

1. Support for separation of design concerns (e.g., from the beginning, topology/connectivity concerns are connected to geometry concerns) is weak.

2. There is a lack of comprehensive support for spatial reasoning. As such, the tools are not easily extensible to layers of services.

3. Support for traceability of requirements to the engineering system itself is nonexistent.
Proposed Approach
Project Scope and Objectives

Modeling and Visualization of the Washington, D.C. Metro System – the first-cut implementation will:

1. Focus on early stages of design, where component selection, positioning and connectivity are the principle concerns.
2. Represent ontologies as UML class diagrams.
3. Not consider system- and component-level behavior.
4. Not consider assignment of functions to components.
Software Architecture Design: GUI Design

<table>
<thead>
<tr>
<th>UML Class Diagram Panel</th>
<th>Engineering Model Panel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Table of Requirements</td>
<td></td>
</tr>
</tbody>
</table>
Software Architecture Design: GUI Design

Proposed Traceability Model

- Requirement
- Design Concept
- Engineering Object

UML Class Diagram Panel
Engineering Model Panel

Table of Requirements
Software Architecture Design: Java Delegation Event Model

- Based on the Publish-Subscribe design pattern
- The DEM refers to publishers as “event sources” and subscribers as “event listeners”
Software Architecture Design: Listener-Driven Events
Software Design Architecture: Listener-Driven Events (cont)
Software Design Architecture: Listener-Driven Events (cont)
Software Architecture Design: Violet UML Editor

- Supports the drawing of Class Diagrams, Sequence Diagrams, Use Case Diagrams, State Diagrams, and Object Diagrams
- Completely free, not platform-dependent
- Built on a graph framework
- See http://horstmann.com/violet/
Washington, D.C. Metro System Example