Finding Reusable Software Components in Large Systems

WCSE 96

James M. Neighbors
Bayfront Technologies, Inc.
neighbrs@netcom.com
Motivation

- work performed between 1980 and 1992

- need reusable software components for a KB forward engineering system (Draco)

- extract reusable software components from existing systems that contain hard to get problem domain knowledge

- large systems must have a lot of domain knowledge

- Goal: manual/semiautomatic extraction and KB encoding of domain knowledge. Can we find its location in a large existing system?
Hypotheses

- economics of large systems - more reengineering than anything else, fit into that context

- problem domains of large systems - too many of them. Focus on existing structure to provide a context for manual extraction of domain knowledge.

- architecture of large systems - focus on "cells" (subsystems) formed by tradeoff between functional decomposition and API decomposition
Experimental Method

- gather interconnection data

- analyze interconnections to form subsystems
  1. cross references
  2. diagrams

- ask system developers
  1. What have we included that doesn't belong here?
  2. What have we not included that does belong here?
Results: Data Collection

- development issues of large systems
  system function, suite of programs, source code availability, version (features), configuration (hardware), nonstandard language usage, lack of documentation

- interconnection data

<table>
<thead>
<tr>
<th>system</th>
<th>role</th>
<th>size</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telcom / Datacom switch</td>
<td>Software Architect (full time)</td>
<td>4M SLOC 3,800 modules</td>
<td>Pascal, C, assembly</td>
</tr>
<tr>
<td>CAD/CAM</td>
<td>Consultant (part-time)</td>
<td>2M SLOC 3,394 modules</td>
<td>FORTRAN, C</td>
</tr>
<tr>
<td>CAE/CAD</td>
<td>Manager (full-time)</td>
<td>4M SLOC 7,089 modules</td>
<td>FORTRAN, C</td>
</tr>
</tbody>
</table>

- maintenance programming
Result: Subsystem Analysis

- determining subsystems
  1. failure: decomposition
  2. failure: intermodule data flow analysis
  3. success: module name pattern matching
  4. success: reference context

- subsystem structure
Conclusions

• subsystems validated by users - they use output

• subsystems are good for reengineering manpower loading

• subsystems are big (average 35 modules with 17,000 source lines)

• subsystems serve as focus for KB extraction

• subsystems may be used to (re-)construct object hierarchies