Maintaining Complex Software
– A Practitioner’s View

Ajay Deshpande
Persistent Systems Ltd

http://www.persistentsys.com
Agenda

- Setting the Context
  - What is Software Maintenance
  - What does it mean to the Product Development industry

- Approaches and Types of Software Maintenance
  - Evolution in Client Expectations
  - Why it’s a hard problem to address

- Challenges at Persistent
  - Motivation of the teams
  - Knowledge management

- Tools we use to make the job easier
  - Static Code Analysis
  - Dynamic Code Analysis
What is Software Maintenance

- Definition from IEEE:
  - The process of modifying a software system or component after its delivery to the customer

- The modifications can be
  - To correct faults
  - Improve some attributes of the software
  - To adapt to a change in the environment.

- Important: Modifications are done after the delivery
  - Leads to iterations of software deliveries
What’s Different in Software

- Software seems very malleable
  - Can we not add this one new feature?
  - That should be an easy fix to add
- The more you change the messier it becomes
  - Software is very interconnected
- It is very hard to replace “parts” with newer / better ones
- Software is very knowledge intensive
- Software is not as mature as some of the other industries yet
Relative Costs of Software Systems

% Of Total Cost

Hardware Costs
Software Costs

Development
Maintenance

Why?

Timeline

Types of Software Maintenance

- **Perfective maintenance**: Changes made to cater for newer user needs or performance attributes

- **Adaptive maintenance**: Changes needed as a consequence of changes in the operating environment (OS, hardware, DBMS)

- **Corrective maintenance**: For the removal of faults in the software

- **Preventive maintenance**: Changes made to software to make it more maintainable (Restructuring)

![Distribution of Maintenance Activities]

- Corrective: 50%
- Adaptive: 21%
- Preventive: 4%
- Perfective: 25%
## Lehman’s Laws (Observations)

<table>
<thead>
<tr>
<th>Law</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuing change</td>
<td>A Software used in a real-world environment must change. Otherwise it becomes progressively less useful in that environment.</td>
</tr>
<tr>
<td>Increasing complexity</td>
<td>As a software evolves, its structure tends to become more complex. Extra effort must be put to keep its structure simple.</td>
</tr>
<tr>
<td>Large program evolution</td>
<td>Software evolution is self-regulating. Attributes such as size, time between releases and the number of reported errors approximately remain the same over releases.</td>
</tr>
<tr>
<td>Organisational stability</td>
<td>Over a software’s lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.</td>
</tr>
<tr>
<td>Continuing growth</td>
<td>The functionality offered by systems has to continually increase to maintain user satisfaction.</td>
</tr>
<tr>
<td>Declining quality</td>
<td>The quality of systems will appear to be declining unless they are adapted to changes in their operational environment.</td>
</tr>
</tbody>
</table>
Challenges at Persistent
Software Maintenance at Persistent

- Persistent Systems works mostly with Independent Software Vendors
  - We primarily build, manage and deploy software products for our clients (and their clients)
  - Software Maintenance is a very key activity
    - Specialized Service Offering around end of life products
- Significant challenges in this field
- Over time we have
  - Collected many best practices that have worked for us
  - Built tools to ease the life of maintenance teams
Challenges and Best Practices

- **Challenge:** People do not feel motivated to work on maintenance projects
  - Hire the right people and ensure fitment
  - Alternate the kind of work people do

- **Challenge:** Managing knowledge and keeping it accessible is hard
  - Typically we start owning systems with little or no documentation
  - Learning about the System and keeping the knowledge around is hard task
  - Search-able Websites (Wiki, Sharepoint, etc) and archived email lists have worked the best
Challenges and Best Practices

- **Challenge: Clients are constantly forced to reduce cost of sustenance**
  - We need to find better / faster ways of doing the same things
  - Automated unit tests are a must to ensure new changes do not break any of the older functionality

- **Other Challenges:**
  - Distributed teams – Time zone, geographies, cultures
  - Shortage of talent – even when job markets are low
  - Complex and some times uncommon technologies used
Sample of Tools at Persistent
Tools in Software Maintenance

- Software tools help us typically achieve these two goals:
  - Understand the structure of existing code (Program Understanding)
  - Restructure the code for better maintainability (Software Re-engineering)
- Excellent support from the open source community in this area
- We will discuss two tools which are extensively helping us
Too

Tool #1: Build++

- A tool of tools consisting of Best of Breed open source
  - Puts your software build on steroids!
- Designed to be easily integrated with regular builds
  - Does static code analysis on programs as it builds
- How does Build++ help in Software Maintenance
  - Helps us to evaluate existing code bases to start with
  - Find parts of code poorly written (coding styles violated)
  - Where code is cut and pasted from place to place
  - Detects dependencies between modules (classes, packages)
  - Detects typical anti patterns (possible defects)
### FindBugs Analysis

FindBugs Analysis generated at: Sun, 6 May 2007 03:12:12 -0400

<table>
<thead>
<tr>
<th>Package</th>
<th>Code Size</th>
<th>Bugs</th>
<th>Bugs p1</th>
<th>Bugs p2</th>
<th>Bugs p3</th>
<th>Bugs Ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall (736 packages), (16445 classes)</td>
<td>963957</td>
<td>3901</td>
<td>259</td>
<td>3642</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.activation</td>
<td>1688</td>
<td>34</td>
<td>5</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.copyobject</td>
<td>71</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.corba</td>
<td>2118</td>
<td>33</td>
<td></td>
<td></td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.dynamicany</td>
<td>2287</td>
<td>16</td>
<td>3</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.encoding</td>
<td>5652</td>
<td>55</td>
<td>1</td>
<td></td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.interceptors</td>
<td>1979</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td>41</td>
</tr>
<tr>
<td>com.sun.corba.se.impl.io</td>
<td>3438</td>
<td>47</td>
<td>2</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.ior</td>
<td>1207</td>
<td>14</td>
<td>2</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.ior.iirp</td>
<td>457</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>com.sun.corba.se.impl.javax.rmi.CORBA</td>
<td>337</td>
<td>3</td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.logging</td>
<td>9374</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>com.sun.corba.se.impl.naming.cosNaming</td>
<td>799</td>
<td>27</td>
<td>1</td>
<td></td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>Source: findbugs.sourceforge.net</td>
<td>690</td>
<td>37</td>
<td>4</td>
<td>33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>com.sun.corba.se.impl.oapoa</td>
<td>2102</td>
<td>31</td>
<td>1</td>
<td></td>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>
Tool #2: Test Suite Generator

- Tool based on static and dynamic code analysis
  - From the source code, it generates simple test cases
  - Using Dynamic Code Analysis it can execute these test cases to find “faults”

- How does TSG help in Software Maintenance
  - Helps in generating basic test cases when none exist in the first place
  - Can save a large amount of effort/time by automatically generating unit test cases
  - The generated test cases are complete - can be run against the source code from the word go
  - Can perform Combinatorial Tests which can reveal potential faults in the code
TSG: Using the Tool

Install TSG Plugin in Eclipse (IDE)

Configure TSG for your (compiled) Eclipse Project

Create Positive Test Cases (PTCs) for selected source code

Modify PTCs (if needed): Default values, Assert Comparisons

Run the test Case to test your code

Generating Positive Test Cases

Select (same) source for Combinatorial testing

Creates Report and Fault Revealing Test Cases.

Get Details of the test from the generated Report.

Later Run FRTCs to reproduce the faults

Regression / Combinatorial Testing
Screenshot: Creating Test Cases

Integrated in the IDE
<table>
<thead>
<tr>
<th>Id</th>
<th>Tested Method Signature</th>
<th>Status</th>
<th>Parameter Creation Algorithm</th>
<th>Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>4792</td>
<td>public void core'util1.iterateArray(core'util1.bean.Bean1[])</td>
<td>Fault</td>
<td>‣ Create PTC instance of type 'TestUtility1' and run setup methods under it. Retrieve field 'util1' of type 'core'util1. Utility1' under it. &lt;br&gt;&quot; Create a new instance of Map 'java'util1.Map'.&lt;br&gt;' Call setter 'setMap' and pass above created instance of type 'java'util1.Map' to it.</td>
<td>java.lang.NullPointerException&lt;br&gt;at core'util1.Utility1.iterateArray(Utility1, jav&lt;br&gt;at sun.reflect.DelegatingMethodAccessorImpl.invoke(Method.java&lt;br&gt;at ecel.input.EclalInputRunner.testExp</td>
</tr>
<tr>
<td>5696</td>
<td>public void core'util1.iterateArray(core'util1.bean.Bean1[])</td>
<td>Fault</td>
<td>‣ Create a new Array instance of type 'core'util1.bean.Bean1'.&lt;br&gt;&quot; Initialize all elements under array.&lt;br&gt;' Set 'U' element under under array of type 'core'util1.bean.Bean1' to null.</td>
<td>java.lang.NullPointerException&lt;br&gt;at core'util1.Utility1.iterateArray(Utility1, jav&lt;br&gt;at sun.reflect.DelegatingMethodAccessorImpl.invoke(Method.java&lt;br&gt;at ecel.input.EclalInputRunner.testExp</td>
</tr>
<tr>
<td>4848</td>
<td>public void core'util1.iterateArray(core'util1.bean.Bean1[])</td>
<td>Fault</td>
<td>‣ Create PTC instance of type 'TestUtility1' and run setup methods under it. Retrieve field 'util1' of type 'core'util1. Utility1' under it. &lt;br&gt;&quot; Create a new instance of Map 'java'util1.Map'.&lt;br&gt;' Call setter 'setMap' and pass NULL for parameter of type 'core'util2.bean.Bean2'.&lt;br&gt;&quot; Created a new instance of type 'class core'util1.bean.Bean1' and assigned it to 'U' element under array.</td>
<td>java.lang.NullPointerException&lt;br&gt;at core'util1.Util...</td>
</tr>
</tbody>
</table>
Summary on Tools

- Tools like these help tighten the development process
- Try to do preventive maintenance early on
  - Tools help us identify weak parts in the code
- Inculcate these habits while developing new code
- A host of other tools on various platforms available for
  - Source code navigation
  - Extracting design from code
  - Database schema re-engineering
  - Code execution visualization
Some Final Notes
In Conclusion

- The Rate of Creation of Software is much higher than the Rate of Retiring Software
  - Software Maintenance is here to stay
- Types of Maintenance: New User Requirements, Defect Fixing and Adaptation
- Lehman’s laws: Software must change and will become more complex as we go along
- Software Maintenance can get to be a thankless job
  - Keeping teams motivated is the biggest challenge
- Tools out there can ease the Software Maintenance process
Leave You With the Million Dollar Question – Maintain or Replace ??

Thank You