A Survey on Mining Software Archives

Survey Presentation

Jin Ung, Oh – 2007.07.11
Contents

• Introduction
• Research Trend
• Research Area
  – General
  – History Modeling
  – Data Extraction
  – Prediction
  – Effort Estimation
  – Mining Aspects
• Papers at a glance
• Related Workshop
• Available Resources
• Discussion
Introduction (1/9)

• What is Software Archive?
  – *Software Archive* is repositories containing any kinds of artifacts generated during software development process.
  – Software Configuration Management
    • *Somebody did something, how can one reproduce it?*
    • Methodology to control and manage a software development project.

  – Contains full record of project history
  – Maintained via programming environments
  – Automatic maintenance and access
  – Freely accessible in open source projects
• Valuable data for research!
Introduction (3/9)

• Why mine Software Archives?
  – Recognizing the potential benefit of mining software archives to support,
    – the maintenance of software systems,
    – improve software design/reuse,
    – Empirically validate novel ideas and techniques

(from MSR ’07 Homepage)

Real Data !
Introduction (4/9)

(from keynote speech of MSR ‘07)
Introduction (5/9)

(from keynote speech of MSR '07)
Introduction (6/9)

(From keynote speech of MSR ‘07)

"Which modules should I test most?"
Introduction (7/9)

“How long will it take to fix this bug?”
Introduction (8/9)

(from keynote speech of MSR ’07)
<table>
<thead>
<tr>
<th>Software</th>
<th>Programming language</th>
<th>History model</th>
<th>Revision ID</th>
<th>Repo. size</th>
<th>Network protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsuTeR</td>
<td>Java</td>
<td>Changeset</td>
<td>Namespace</td>
<td>(revisions)</td>
<td>custom</td>
</tr>
<tr>
<td>Aldon</td>
<td>C++, Java</td>
<td>Snapshot</td>
<td>Namespace</td>
<td>(revisions)</td>
<td>custom</td>
</tr>
<tr>
<td>Bazaar</td>
<td>Python</td>
<td>Snapshot</td>
<td>Pseudobranch</td>
<td>(patch)</td>
<td>HTTP, SFTP, FTP, ssh, custom, email bundles[8]</td>
</tr>
<tr>
<td>ClearCase</td>
<td>?</td>
<td>Snapshot</td>
<td>Namespace</td>
<td>(patch)</td>
<td>HTTP, custom (CCS), custom (MVFS filesystem driver)</td>
</tr>
<tr>
<td>Code Co-op</td>
<td>C++</td>
<td>Changeset</td>
<td>User ID-Ordinal</td>
<td>(patch)</td>
<td>email (MAP, SMTP, POP3, GMail), LAN</td>
</tr>
<tr>
<td>CVS</td>
<td>C</td>
<td>Snapshot</td>
<td>Namespace</td>
<td>(patch)</td>
<td>pserv[20][9], ssh</td>
</tr>
<tr>
<td>CVSHUT</td>
<td>C++</td>
<td>Changeset</td>
<td>Namespace</td>
<td>(patch)</td>
<td>sspl, sserv, pserv, pserv, ssh</td>
</tr>
<tr>
<td>darcs</td>
<td>Haskell</td>
<td>Patch</td>
<td>Namespace</td>
<td>(patch)</td>
<td>HTTP, ssh, email</td>
</tr>
<tr>
<td>Git</td>
<td>C, shell scripts</td>
<td>Snapshot</td>
<td>SHA-1 hashes</td>
<td>(patch)</td>
<td>custom, ssh, rsync, HTTP, FTP, email, bundles</td>
</tr>
<tr>
<td>GNU arch</td>
<td>C, shell scripts</td>
<td>Changeset</td>
<td>Namespace</td>
<td>(patch)</td>
<td>WebDAV, HTTP</td>
</tr>
<tr>
<td>Mercurial</td>
<td>Python, C</td>
<td>Changeset</td>
<td>SHA-1 hashes</td>
<td>(patch)[8]</td>
<td>HTTP, ssh, email (with plug-in)</td>
</tr>
<tr>
<td>Monotone</td>
<td>C++</td>
<td>Hybrid[7]</td>
<td>SHA-1 hashes</td>
<td>(patch)[?]</td>
<td>custom (&lt;mesync)[9]</td>
</tr>
<tr>
<td>Perforce</td>
<td>?</td>
<td>Changeset</td>
<td>Namespace</td>
<td>(patch)</td>
<td>custom</td>
</tr>
<tr>
<td>SourceHaven</td>
<td>C, Java</td>
<td>Snapshot</td>
<td>Namespace</td>
<td>(patch)</td>
<td>WebDAV, custom</td>
</tr>
<tr>
<td>StarTeam</td>
<td>?</td>
<td>Snapshot</td>
<td>MD5 hashes</td>
<td>(revision)</td>
<td>custom</td>
</tr>
<tr>
<td>Subversion</td>
<td>C</td>
<td>Snapshot</td>
<td>Namespace</td>
<td>(patch)</td>
<td>WebDAV, custom (subserve)[9]</td>
</tr>
<tr>
<td>SVN</td>
<td>Perl</td>
<td>Changeset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vault</td>
<td>C</td>
<td>Snapshot</td>
<td>?</td>
<td>(patch)</td>
<td>HTTP, HTTPS</td>
</tr>
<tr>
<td>Visual Source Safe</td>
<td>C</td>
<td>Snapshot</td>
<td>Namespace?</td>
<td>(changes)?</td>
<td>None, but can access repository files through a &quot;share&quot;</td>
</tr>
<tr>
<td>Team Foundation Server</td>
<td>C++ and C#</td>
<td>Changeset</td>
<td>Namespace</td>
<td>(patch)</td>
<td>SOAP over HTTP or HTTPS</td>
</tr>
<tr>
<td>LibreSource Synchronizer</td>
<td>Java</td>
<td>Changeset</td>
<td>Timestamp of the repository</td>
<td>(patch)</td>
<td>HTTP, File-System</td>
</tr>
<tr>
<td>Plastic SCM</td>
<td>C#</td>
<td>Changeset</td>
<td>Namespace</td>
<td>(revision)</td>
<td>custom</td>
</tr>
</tbody>
</table>
Research Trend (1/4)

• Andreas Zeller
  – Full professor at University Saarland (Germany)
  – http://www.st.cs.uni-sb.de/zeller/
  – Research Area
    – Mining Software Archives
    – Automated Debugging
    – Mining Models

• Thomas Zimmermann
  – PhD student at University Saarland (with Andreas Zeller)
  – http://www.st.cs.uni-sb.de/~zimmerth/
  – Research Area
    – History and Defects prediction
    – Anything related to history
    – Graph Theory
Research Trend (:Framework)

- OSS Projects
  - Source Codes
  - Modification Record
    - Date, Time
    - Author
    - Change log
  - Bug-Archives
    - Creation-Time
    - End-Time
    - Description
    - Classification (Type)
    - Priority / Severity
    - Reporter / Assigned
  - Mailing Archives
    - Participants
    - Mail Contents

- Framework
- Derived Data
  - Preprocessed Data
  - Model
  - Software Metrics

IDEA!

- Data Mining
- Regression Model
- ...

- Easy evaluation / validation
- Stimulate idea
Research Trend (3/4)
Research Trend (4/4)

TSE
- 2001 Does Code Decay – Assessing the Evidence from Change Management Data
- 2002 Using Version Control Data to Evaluate the Impact of Software Tools
- 2002 Visualizing Software Changes
- 2005 Analyzing the Evolutionary History of the Logical Design of OO Software
- 2005 Hipikat : A Project Memory for Software Development
- 2005 Mining Version History to Guide Software Changes
- 2005 Toward Understanding the Rhetoric of Small Code Changes

TOSEM
- 2002 Two Case Studies of Open Source Software Development – Apache and Mozilla
- 2005 Impact of Software Engineering Research on the Practice of SCM
- 2007 Representing Concerns in Source Code
Research Area

- What is studied with (about) Software Archives?

- Theoretical
- Effort Measure Estimation
- Design
- Fault Prediction
- Impact Analysis
- Pattern (Aspects)
- Data Extraction
- Tool / Automation
- History Modeling / System Suggestion
General

  - VCS can illuminate the software development process in new ways.
  - First paper that suggested to use software archives.
General


MR : Modification Record
General

• Does Code Decay? Assessing the Evidence from Change Management Data (TSE, 2001)
  ▪ Code decay: Change in large software systems becomes increasingly difficult over time.
  ▪ Metric: CDI (Code Decay Indices)

• Predicting Faults from Cached History (ICSE ‘07)
  ▪ (Predicting category)
  ▪ Shows four kinds of defect locality
    ▪ Changed-entity locality
    ▪ New-entity locality
    ▪ Temporal locality
    ▪ Spatial locality
Data Extraction

- Don’t Program on Fridays! : How to Locate Fix-Inducing Changes (WSR ‘05)
  - Fix-Inducing changes : Changes that lead to problems, indicated by fixes.

Figure 1: Locating fix-inducing changes for bug 42233

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P(\text{fix}))</td>
<td>18.4</td>
<td>20.9</td>
<td>20.0</td>
<td>22.3</td>
<td>24.0</td>
<td>14.7</td>
<td>16.9</td>
<td>20.8</td>
</tr>
<tr>
<td>(P(\text{bug}))</td>
<td>11.3</td>
<td>10.4</td>
<td>11.1</td>
<td>12.1</td>
<td>12.2</td>
<td>11.7</td>
<td>11.6</td>
<td>11.4</td>
</tr>
<tr>
<td>(P(\text{bug} \cap \text{fix}))</td>
<td>4.6</td>
<td>4.8</td>
<td>4.6</td>
<td>5.2</td>
<td>5.6</td>
<td>4.5</td>
<td>4.5</td>
<td>4.9</td>
</tr>
<tr>
<td>(P(\text{bug}</td>
<td>\text{fix}))</td>
<td>25.1</td>
<td>22.9</td>
<td>23.3</td>
<td>23.5</td>
<td>23.2</td>
<td>30.3</td>
<td>26.4</td>
</tr>
<tr>
<td>(P(\text{bug} \neg \text{fix}))</td>
<td>8.2</td>
<td>7.1</td>
<td>8.1</td>
<td>8.8</td>
<td>8.7</td>
<td>8.4</td>
<td>8.6</td>
<td>8.1</td>
</tr>
</tbody>
</table>

Table 1: Distribution of fixes and fix-inducing changes across day of week in ECLIPSE.
Data Extraction

• Don’t Program on Fridays! : How to Locate Fix-Inducing Changes (WSR ‘05)

Use log-text, bug-introducing date, bug-fixed date to locate a fix-inducing bug

Locate bug-fix to change
History Modeling

- Modeling history to analyze software evolution (JSME ‘06)
  - Introduces a model to represent history information. (Hismo)
- Towards the Integration of Versioning Systems, Bug Reports and Source Code Meta-Models (Electronic Notes in Theoretical Computer Science ‘05)
Prediction

- Mining Version Histories to Guide Software Changes (TSE 2005)
  - Uses data mining technique to predict co-change
  - ROSE-tool (Eclipse Plug-in)

Fig. 2. Evolutionary coupling in ECLIPSE.
Prediction

- Mining Version Histories to Guide Software Changes (TSE 2005)

Fig. 7. Predictive power for ECLIPSE related to time (2001-04-28 to 2004-09-14) and releases.

Fig. 8. Predictive power for GCC related to time (1997-08-11 to 2004-08-23) and releases.
Prediction

- Mining Version Histories to Guide Software Changes (TSE 2005)

Archives → MR → Co-Change

Validation: Precision, Recall, Likelihood, Feedback

Mine Rules → Predictor

Tool Implementation (Eclipse Plug-in)
Prediction

- Mining Metrics to Predict Component Failures (ICSE ‘06)
  - Using principle component analysis on the code metrics, built regression models that predicts post-release defects.

Table 1. Projects researched

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
<th>Components</th>
<th>Code size</th>
<th>Team size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer 6</td>
<td>Web browser</td>
<td>HTML rendering</td>
<td>511 KLOC</td>
<td>14.3X</td>
</tr>
<tr>
<td>IIS W3 Server core</td>
<td>Web server</td>
<td>Application loading</td>
<td>37 KLOC</td>
<td>6.3X</td>
</tr>
<tr>
<td>Process Messaging Component</td>
<td>Application communication and networking</td>
<td>all</td>
<td>147 KLOC</td>
<td>3.4X</td>
</tr>
<tr>
<td>DirectX</td>
<td>Graphics library</td>
<td>all</td>
<td>306 KLOC</td>
<td>18.5X</td>
</tr>
<tr>
<td>NetMeeting</td>
<td>A/V Conferencing</td>
<td>all</td>
<td>109 KLOC</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 2. Research hypotheses

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>Increase in complexity metrics of an entity $E$ correlates with the number of post-release defects of $E$.</td>
</tr>
<tr>
<td>$H_2$</td>
<td>There is a common subset of metrics for which $H_1$ applies in all projects.</td>
</tr>
<tr>
<td>$H_3$</td>
<td>There is a combination of metrics which significantly predicts the post-release defects of new entities within a project.</td>
</tr>
<tr>
<td>$H_4$</td>
<td>Predictors obtained using $H_3$ from one project also predict failure-prone entities in other projects.</td>
</tr>
</tbody>
</table>
For each project, we can find a set of complexity metrics that correlate with post-release defects – and thus failures. There is no single set of metrics that fits all projects. Predictors obtained from PCA are useful in building regression models to estimate post-release defects. Predictors are accurate only when obtained from the same or similar projects.

Table 3. Metrics and their correlations with post-release defects. For each module \( M \), we determine how well the metrics correlate with \( M \)'s post-release defects. Bold values indicate significant correlation.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Correlation with post-release defects of ( M )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module metrics — correlation with metric in a module ( M )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td># Classes in ( M )</td>
<td>0.531 0.612 0.712 0.066 0.438</td>
</tr>
<tr>
<td>Function</td>
<td># Functions in ( M )</td>
<td>0.131 0.699 0.761 0.164 0.511</td>
</tr>
<tr>
<td>GlobalVariables</td>
<td># global variables in ( M )</td>
<td>0.023 0.664 0.695 0.108 0.460</td>
</tr>
<tr>
<td>Per-function metrics — correlation with maximum and sum of metric across all functions ( f ) in a module ( M )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines</td>
<td># executable lines in ( f )</td>
<td>Max -0.236 0.514 0.585 0.496 0.590</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total 0.131 0.709 0.797 0.187 0.560</td>
</tr>
<tr>
<td>Parameters</td>
<td># parameters in ( f )</td>
<td>Max -0.244 0.372 0.547 0.043 0.346</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total 0.116 0.689 0.706 0.152 0.478</td>
</tr>
<tr>
<td>Arcs</td>
<td># arcs in ( f )'s control flow graph</td>
<td>Max -0.209 0.375 0.587 0.527 0.444</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total 0.127 0.679 0.802 0.158 0.484</td>
</tr>
<tr>
<td>Blocks</td>
<td># basic blocks in ( f )'s control flow graph</td>
<td>Max -0.245 0.347 0.585 0.546 0.462</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total 0.128 0.707 0.787 0.158 0.472</td>
</tr>
<tr>
<td>ReadCoupling</td>
<td># global variables read in ( f )</td>
<td>Max -0.005 0.582 0.633 0.362 0.299</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total -0.172 0.676 0.756 0.277 0.443</td>
</tr>
<tr>
<td>WriteCoupling</td>
<td># global variables written in ( f )</td>
<td>Max 0.043 0.618 0.392 0.011 0.450</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total -0.128 0.629 0.629 0.230 0.406</td>
</tr>
<tr>
<td>AddToHashSetCoupling</td>
<td># global variables whose address is taken in ( f )</td>
<td>Max 0.237 0.491 0.412 0.016 0.263</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total 0.175 0.145 0.024 0.357 0.372</td>
</tr>
<tr>
<td>ProcCoupling</td>
<td></td>
<td>0.000 0.443 0.037 0.530 0.034 0.537</td>
</tr>
<tr>
<td>PmIn</td>
<td></td>
<td>0.245 0.465 0.046 0.566 0.451 0.538</td>
</tr>
<tr>
<td>PmOut</td>
<td></td>
<td>0.165 0.529 0.100 0.283 0.094 0.469</td>
</tr>
<tr>
<td>Complexity</td>
<td></td>
<td>0.121 0.323 0.111 0.425 0.118 0.383</td>
</tr>
</tbody>
</table>

Figure 2. Comparing predicted and actual rankings.
Prediction

- Mining Metrics to Predict Component Failures (ICSE ‘06)
Effort Estimation

- Inferring Change Effort from Configuration Management Databases (Int’l Symposium on Software Metrics 1998)
  - Methodology and algorithm for historical analysis of the effort necessary for developers to make changes to software.
  - System under study was “decaying” at a rate of 20% per year.
Mining Aspects

- Mining Aspects from Version History (WSR ‘06)

Figure 1. Mining workflow
Papers at a glance

• How Documentation Evolves Over Time (IWPSE ‘07)
• How Design Predicts Failure (WSR ’06)
• Predicting Faults from Cached History (ICSE ’07)
• Dynamine: Finding Common Error Patterns by Mining Software Revision Histories (ESEC / SIGSOFT FSE ’05)
• Using Version Control Data to Evaluate the Impact of Software Tools – A Case Study of the Version Editor (TSE ’02)

• Determining Implementation Expertise from Bug Reports (MSR ‘07)
• Recommending Emergent Teams (MSR ‘07)
• Correlating Social Interactions to Release History During Software Evolution (MSR ‘07)
• Mining CVS Repositories to Understand Open-Source Project Developer Roles (MSR ‘07)
• Finding Relevant Applications For Prototyping (MSR ‘07)
Research Framework : Revisited

- OSS Projects
  - Source Codes
  - Modification Record
    - Date, Time
    - Author
    - Change log
- Bug-Archives
  - Creation-Time
  - End-Time
  - Description
  - Classification (Type)
  - Priority / Severity
  - Reporter / Assigned
- Mailing Archives
  - Participants
  - Mail Contents

- Framework
- Derived Data
  - Preprocessed Data
  - Model
  - Software Metrics

IDEA!

- Data Mining
- Regression Model
- ...

- Easy evaluation / validation
- Stimulate idea
Related Workshop

• Mining Software Repository (MSR)
  ▪ http://msr.uwaterloo.ca/msr2007/
  ▪ MSR Mining Challenge 2007
    • For Eclipse / Mozilla
    • 1) Mining / Formulate mining questions.
    • 2) Predict for Eclipse the number of bug/changes that will happen between February 1 and April 30, 2007.

• MSR Mining Challenge 2006
  ▪ Bug (defect) analysis and prediction
  ▪ Change impact, propagation, and coupling analysis
  ▪ Process analysis
  ▪ Team structure and interaction analysis
Related Workshop

- PROMISE
  - PROMISE ’05, ‘06, ‘07 (3rd)
  - http://promisedata.org/
  - Homepage to maintain data for researchers to publish their data and the tools used to make their conclusions.
  - Paper and data categories
    - Open source
    - Effort estimation
    - Defect estimation
    - Models
Available Resources

• MSR 2007 Challenge Data
  • Eclipse CVS repository (!) (retrieved on 2006-11-25)
  • Eclipse Bugzilla export (in XML)
  • Firefox/Mozilla CVS repository (retrieved on 2006-12-17)
  • Firefox/Mozilla Bugzilla export (in XML)
  • http://msr.uwaterloo.ca/msr2007/challenge/

• PROMISE Repository
  • Many paper’s data (in arff format)

• OSS Projects
  • Apache Software Foundation (http://www.apache.org)
  • Source Forge (http://sourceforge.net)
Avail
• Eclipse

Available Resources
• Eclipse bugzilla export (in XML)
Available Resources

- Apache Projects

Apache Projects

- HTTP Server
- ActiveMQ
- Ant
- APR
- Beehive
- Cavelle
- Cocoon
- DB
- Directory
- Excalibur
- Felix
- Forrest
- Germinado

Welcome!

The Apache Software Foundation

Welcome

Ant 1.7.0

SVN Repositories

Access the Source Tree (Subversion)

Anyone can checkout source code from our public Subversion repos. To do so, simply use the following command (if you are using a GUI client, configure it appropriately):

```bash
svn co http://svn.apache.org/repos/asf/ant/[project]/trunk/ ant-[project]
```

Modules available for access are:

- **ant** - The "main" Ant module.
  http://svn.apache.org/repos/asf/ant/core/trunk/
- **antunit** - Test framework for Ant.
Available Resources

- SourceForge

[SourceForge Image]

Most Active Projects - Last Week

Rank  | Project   | Anonymous CVS Access
---    |----------|----------------------
1      | Azureus  |
2      |          |
3      |          |
4      |          |
5      |          |
6      |          |
7      |          |

This project's SourceForge.net CVS repository can be checked out through anonymous (pserver) CVS with the following instruction set. The module you wish to check out must be specified as the `modulename`. When prompted for a password for anonymous, simply press the Enter key. To determine the names of the modules created by this project, you may examine their CVS repository via the provided web-based CVS repository viewer.

```
cvs
-d:pserver:anonymous@azureus.cvs.sourceforge.net:/cvsroot/azureus
login

cvs -z3
-d:pserver:anonymous@azureus.cvs.sourceforge.net:/cvsroot/azureus
co -P modulename
```

Information about accessing this CVS repository may be found in our document titled, "CVS (Version Control for Source Code)."

Updates from within the module's directory do not need the `-d` parameter.
Discussion

• What can we do?
  • Process
  • Model
  • Refactoring
  • Impact Analysis
  • …