Discussion of a Large-Scale Open Source Data Collection Methodology

Michael Hahsler and Stefan Koch
Department of Information Business, Vienna University of Economics and BA
{michael.hahsler|stefan.koch}@wu-wien.ac.at

Presented at the HICSS-38, January 3-6, 2005
Hilton Waikoloa Village, Big Island, Hawaii

Motivation

• Bazaar as a new development paradigm
  – Large number of developers
  – Rigorous peer review / parallel debugging
  – Evolving software products / frequent releases

• Existing Studies
  – Ideological debates
  – In-depth analyses of single projects

• Successful project hosting sites
  – Control and manage OSS development
  – Virtual communities

→ Methodology for large scale quantitative investigations to analyze the new development paradigm
Outline

• Methodology
  – Data retrieval
  – Metrics

• Possible analyzes
  – Community/Project/Participant level
  – Effort and Productivity

• Discussion of advantages and disadvantages

Data Retrieval

• Data sources
  – Version control system (e.g., CVS or SVN)
  – Additional information (e.g., from project web pages, bug tracking systems)

• Collected data
  – Consistently aggregated data
  – Easy to access for analyzes (e.g., in a relational database)

• Example: Sourceforge.net
  – 20,000+ projects (in 2002)
  – Extract information from summary page of each project
  – 8,791 projects using CVS actively
  – Download of 33 GB of version control information
Metrics

Many metrics are possible. Some commonly used metrics are:

- Lines-of-code (LOC, NCSS)
- Commits (associated with change requests)
- Participating programmers
- Active time spend on the project
- Development indicator (planning, alpha, stable,...)
Possible Analyzes

Participant Level
- LOC, Commits
- Activity Patterns
- Programming Style

Project Level
- Distribution of Effort (Inequality)
- Cooperation on Files
- SW Evolution
- Coordination
- Productivity
- Effort Estimation

Community Level
- Distribution of Inputs / Outputs
- Relationship Inputs / Outputs
- Co-Participation in Projects
- Team
  - Distribution of Effort (Inequality)
  - Cooperation on Files
- Community
  - Available assets
  - Developers
  - Activity
    - Active time spent on projects
    - Number of commits
    - Collaboration
  - Outcome
    - Size (e.g., LOC)
    - Value (e.g., number of downloads of a software product)
    - Community (human capital)

Examples: Community Level

Histogram of project size in Sourceforge.net: Power-law distributions are common as a result of positive feedback loops
Examples: Project Level

- **Productivity**: Does activity of developers depend on project status, teamwork or well-known “core“ developer?

- **Programming practices**: e.g., software patterns, frameworks

- **Software Evolution**
  
  *Law of SW evolution (for commercial development): Growth rate decreases over time caused by growing complexity*

  - Also valid for OSS? Or do OSS development practices enable super-linear growth?
  - 39% of the Sourceforge.net projects exhibit super-linear growth. Why?

Examples: Participant Level

- **Distribution of effort within a project team**
  - Lorenz curves
  - Gini coefficient

  20% of the developers do typically 80% of the coding

Lorenz curves for distribution of commits within two projects from Sourceforge.net
Effort Estimation

- Effort is needed for comparison of OSS to traditional development
- For OSS effort is typically unknown (no time sheets)

Effort estimation for projects on Sourceforge.net

<table>
<thead>
<tr>
<th>Model</th>
<th>Median effort per project</th>
<th>Total effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>COCOMO</td>
<td>2.02 PY</td>
<td>160,020 PY</td>
</tr>
<tr>
<td>Rayleigh-Norden</td>
<td>0.69 PY</td>
<td>5,965 PY</td>
</tr>
</tbody>
</table>

PY ... person-years

Problems
- Models are calibrated for commercial software development
- Often has restrictions which make it incompatible with OSS

Discussion

Collect data automatically for a large number of projects developed by a community

- Advantages:
  - Starting point for effort estimation
  - Allows statistical tests for differences between projects
  - Analyzing community aspects

- Drawbacks:
  - Uncertainty about data quality
  - Need to adapt retrieval component for each hosting site/service