Towards a Better Understanding of Software Evolution
An Empirical Study on Open Source Software

Guowu Xie        Jianbo Chen        Iulian Neamtiu

University of California, Riverside
Motivation

• Software evolution expensive
  – 50%..90% of product cost

• Understanding software evolution essential
  – Models useful for developers, managers

• Long-lived open-source programs available
  – Sendmail, GNU programs, Samba, etc.
Contributions

• Analyzed 70 years of software evolution
  – 7 programs, 653 official releases
• Test Lehman’s eight software evolution laws
  – First study to consider each law, variety of metrics
• New observations
  – Parallel evolution and growth
  – Power laws in software change
Applications examined

criteria: size, lifespan, actively maintained, widely used

<table>
<thead>
<tr>
<th>Program</th>
<th>Time frame (years)</th>
<th>Last release (kLOC)</th>
<th>Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samba</td>
<td>15</td>
<td>1,046</td>
<td>189 x</td>
</tr>
<tr>
<td>Sendmail</td>
<td>15</td>
<td>88</td>
<td>3.4 x</td>
</tr>
<tr>
<td>Bind</td>
<td>9</td>
<td>322</td>
<td>1.9 x</td>
</tr>
<tr>
<td>OpenSSH</td>
<td>9</td>
<td>52</td>
<td>4.1 x</td>
</tr>
<tr>
<td>SQLite</td>
<td>8</td>
<td>65</td>
<td>3.8 x</td>
</tr>
<tr>
<td>Vsftpd</td>
<td>8</td>
<td>16</td>
<td>2.3 x</td>
</tr>
<tr>
<td>Quagga</td>
<td>5</td>
<td>48</td>
<td>1.1 x</td>
</tr>
</tbody>
</table>
“Confirmed” Lehman laws
Continuing change

Samba

- modifications > additions > deletions
- implementation changes: 16x..30x interface changes
### Increasing complexity

<table>
<thead>
<tr>
<th>Module size</th>
<th>Calls per function</th>
<th>Cyclomatic complexity</th>
<th>Common coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Absolute</td>
<td>Normalized</td>
</tr>
<tr>
<td>➜</td>
<td>➜</td>
<td>➜</td>
<td>➜</td>
</tr>
</tbody>
</table>

Bad news: local decreases due to size increases
Self regulation

OpenSSH

• large changes followed by small corrections
• same results when using function changes as metric
Continuing growth

- similar graphs when using *modules* or *definitions* as metrics
- size decreases interesting (subsystem rewriting)
Laws not confirmed
Declining quality

“software quality appears to be declining, unless proactive measures are taken to adapt the software to its operational environment”

<table>
<thead>
<tr>
<th>External Quality</th>
<th>Internal Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defects</td>
<td>Defect density</td>
</tr>
<tr>
<td></td>
<td>Complexity</td>
</tr>
</tbody>
</table>
Invariant work rate

“rate of productive output tends to stay constant”

1. Average changes per day (Lehman) *not invariant*
2. Growth rate, change rate *not subsiding*
Conservation of familiarity

“Incremental system growth constant or declines”

<table>
<thead>
<tr>
<th>Net module growth</th>
<th>Function growth rate</th>
<th>Total # of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Arrow]</td>
<td>![Arrow]</td>
<td>![Arrow]</td>
</tr>
</tbody>
</table>
### Growth Rate

<table>
<thead>
<tr>
<th>Theory (Turski)</th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \frac{\Delta S}{\Delta t} \approx t^{-2/3} ]</td>
<td></td>
</tr>
</tbody>
</table>

**Feedback system**

- **Growth Rate**
  - Theory (Turski): \[ \frac{\Delta S}{\Delta t} \approx t^{-2/3} \]
  - Practice:

**Graphs**

- **Samba**
  - Delta Growth Rate vs. Time

---

Guowu Xie, Jianbo Chen, Iulian Neamtiu  
*Towards a Better Understanding of Software Evolution: An Empirical Study on Open Source Software*  
14
New observations
Parallel evolution

**Bind**

- Super-linear growth (development branch)
- Linear growth (maintenance branch)

**Samba**

- Super-linear growth (development branch)
- Linear growth (maintenance branch)
Power laws in software changes

which parts of the code change, and how frequently?
Consequences of our study
Consequences for practitioners

• Continuous monitoring is essential
  – prevent hot-spots, complexity increases

• Project planning
  – parallel evolution
  – continuous growth: super-linear (development), linear (maintenance)
Questions for researchers

• Closed-source software

• More precise operational definitions
  – Are all laws expected to hold?

• Other languages, e.g., C++, Java, Ruby
Conclusions

• Study of long-term evolution
  – 7 applications, more than 70 years

• Tested Lehman’s software evolution laws
  – Variety of metrics
  – Reconcile contradictory results in prior work

• Benefits researchers and practitioners