IDPD and Software Architecture

Daniel Link and Ramin Moazeni
Incremental Development Productivity Decline (IDPD)

An overall decline in productivity of the increments

IDPD factor: the percentage of decline in software productivity from one increment to the next

Reason of decline: previous-increment breakage, usage feedback, increased integration and testing effort, all charged to current-increment budget
Incremental Development Productivity Decline

- Example: Site Defense BMD Software
  - 5 builds, 7 years, $100M
  - Build 1 productivity over 300 SLOC/person-month
  - Build 5 productivity under 150 SLOC/person-month
    - Including build 1-4 breakage, integration, rework
  - 318% requirement change across all builds
  - A factor of 2 decrease across 4 builds corresponds to an average IDPD factor of 20% productivity decrease per build
- Similar IDPD factors have been found for
  - Large commercial software such as multi-year slippage in delivery of MS Word and Windows Vista
IDPD Model

- EKSLOC(I) = KSLOC(I-1)*(0.4*DM+0.3*CM+0.3*IM)
  - EKSLOC(I) is based on what is adapted and reused from the previous increment
- NKSLOC(I) is the new code written for increment I
- KSLOC(I)=EKSLOC(I)+NKSLOC(I)
- IDPD(I)=NKSLOC(I)/KSLOC(I)

- For this study, given that EKSLOC can't be determined, we use only new SLOC (NSLOC)
  - Productivity = NSLOC/t
Effects of IDPD on Number of Increments

- Model relating productivity decline to number of builds needed to reach 8M SLOC Full Operational Capability
- Assume Build 1 production of 2M SLOC @100 SLOC/PM
  - 20000 PM/24mo. = 833 developers
  - Constant Staff size for all builds
- Going from IDPD=10 to IDPD=20 increases schedule by $\frac{8}{5}=1.6$, or 60%
Exploration of IDPD Factor

Sources of variations based on experience on several projects

<table>
<thead>
<tr>
<th>Higher IDPD (less productivity)</th>
<th>Lower IDPD (more productivity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort to maintain previous increments; bug fixing, COTS upgrades, interface changes, all reused SLOC, not ESLOC</td>
<td></td>
</tr>
<tr>
<td>Next increment requires previous-increment modifications</td>
<td></td>
</tr>
<tr>
<td>Next increment spun-out to more platforms</td>
<td>Next increment touches less of previous increments</td>
</tr>
<tr>
<td>Next increment has more previous increments to integrate/interact with</td>
<td>Current staff more experienced, productive</td>
</tr>
<tr>
<td>Staff turnover reduces experience level</td>
<td></td>
</tr>
<tr>
<td>Next increment software more complex</td>
<td>Next increment software less complex</td>
</tr>
<tr>
<td>Previous increments incompletely developed, tested, integrated</td>
<td>Previous increments did much of next increment’s requirements, architecture</td>
</tr>
</tbody>
</table>
# Software Categories and IDPD

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Expected IDPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Applications that may build on infrastructure software</td>
<td>+</td>
</tr>
<tr>
<td>Infrastructure Software</td>
<td>Software that provides services to other software</td>
<td>++</td>
</tr>
<tr>
<td>Platform Software</td>
<td>Drivers integrated with hardware</td>
<td>0</td>
</tr>
<tr>
<td>Firmware</td>
<td>Simple firmware that cannot be updated</td>
<td>0</td>
</tr>
<tr>
<td>Support Software</td>
<td>Non-deployable throwaway code</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Normalized Productivity Trendlines

- Cisco streaming
- Cisco unnamed
- XP 1
- XP 2
- QMP
- System of Systems
- ODA
- Vu 5
- Linear (Cisco streaming)
- Linear (Cisco unnamed)
- Linear (XP 1)
- Linear (XP 2)
- Linear (QMP)
- Linear (System of Systems)
- Linear (ODA)
- Linear (Vu 5)
Software Architecture and its Recovery

- Architecture = Structure and Reason (Design Decisions)
- Knowing helps Maintenance
- Prescriptive vs. Descriptive
- Drift, Decay, Erosion
- Actually Reflected = ?
- Recover from Artifacts!
- Different Paradigms (e.g., Programming Patterns, Concerns), Different Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDC</td>
<td>Development Pattern Matching</td>
</tr>
<tr>
<td>ARC</td>
<td>Concern Oriented (Topic Modeling)</td>
</tr>
<tr>
<td>PKG</td>
<td>Package Structure</td>
</tr>
<tr>
<td>RELAX</td>
<td>Concern Oriented (Text Classification)</td>
</tr>
</tbody>
</table>
Measures of Architectural Similarity

**a2a**
- Overall Architecture

**c2c**
- Components (Clusters) Within an Architecture
RQ1 (IDPD and Overall Architecture Change)
• Is there a correlation between the IDPD and the architectural change between two versions of a system?

RQ2 (IDPD and Component Change)
• Is there a correlation between the IDPD and the change in the components between two versions of a system?
Data Collection Challenges

• Increments with limited compilability (e.g., ancient tools required)
• Recovery methods take a long to run and crash
• Similarity measures crash

• Hofstadter’s Law: “It always takes longer than you expect, even when you take into account Hofstadter's Law.”
Apache Hadoop Versions
## Side Observation: PSLOC/LSLOC

<table>
<thead>
<tr>
<th>System</th>
<th>Versions</th>
<th>Average Ratio</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Chukwa</td>
<td>8</td>
<td>1.34</td>
<td>0.04</td>
</tr>
<tr>
<td>Apache Tomcat</td>
<td>9</td>
<td>1.48</td>
<td>0.00</td>
</tr>
<tr>
<td>Apache Hadoop</td>
<td>19</td>
<td>1.39</td>
<td>0.03</td>
</tr>
<tr>
<td>Type</td>
<td>Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLOC Difference % and a2a_RELAX</td>
<td>0.81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLOC Difference % and a2a_ACDC</td>
<td>0.91</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLOC Difference % and c2c_ACDC</td>
<td>0.69</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IDPD is caused by previous-increment breakage, usage feedback, increased integration and testing effort, all charged to current-increment budget.

In this study, we have also observed that Architectural change contributes to IDPD.

This is evident from the high correlation between Architectural change and IDPD and SLOC sizes.

High architectural change correlates to high SLOC sizes. However, this would not conclude higher productivity as high architectural change may also contribute to requirement change, code breakage, handling interface changes and more testing effort which would further reduce the productivity.
References


Publications

• Moazeni, R, Link D, Chen C, and Boehm B. Software Do mains in Incremental Development Productivity Decline.” ACCEPTED for publication, ICSSP 2014

• Moazeni, R, Link D, & Boehm B. “COCOMO II Parameters and IDPD: Bilateral Relevances” ACCEPTED for publication, ICSSP 2014


Publications


Questions