Productivity Decline in System of Systems Software Development

Ramin Moazeni
A. Winsor Brown
Barry Boehm
Table of Contents

• Incremental Development Productivity Decline (IDPD)
  – Overview
  – Quality Management Platform Project
    • Overview
    • Data Collection
    • Data Analysis – Productivity Trends
  – Exploration of IDPD Factor
  – IDPD Ranges
  – Effect of IDP on number of increments

• Conclusion
• Future Research
Incremental Development Productivity Decline (IDPD)

• Overview
  – The “Incremental Development Productivity Decline” (IDPD) factor represents the percentage of decline in software producibility from one increment to the next.
  – The decline is due to factors such as previous-increment breakage and usage feedback, increased integration and testing effort.
  – Another source of productivity decline is that maintenance of reused previous build software is not based on equivalent lines of software credited during the previous build, but on the full amount of reused software.
    • Build 1: 200 KSLOC new, 200K Reused@20% yields a 240 K ESLOC “count” for estimation models.
    • Build 2: there are 400 KSLOC of Build 1 to maintain and integrate
  – Such phenomena may cause the IDPD factor to be higher for some builds and lower for others.
Incremental Development Productivity Decline (IDPD)

• 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% change in requirements across all builds
    • A factor of 2 decrease across 4 builds corresponds to an average build to build 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 for Windows and Windows Vista.
    • Large agile-development projects that assumed a zero IDPD factor
Quality Management Platform (QMP) Project

• QMP Project Information:
  – Web-based application
  – System is to facilitate the process improvement initiatives in many small and medium software organizations
  – 6 builds, 6 years, different increment duration
  – Size after 6th build: 548 KSLOC mostly in Java
  – Average staff on project: ~20
Quality Management Platform (QMP) Project

• Data Collection
  – Most data come from release documentation, build reports, and project member interviews/surveys
  – Data include product size, effort by engineering phase, effort by engineering activities, defects by phase, requirements changes, project schedule, COCOMO II driver ratings (rated by project developers and organization experts)
  – Data collection challenges:
    • Incomplete and inconsistency data
    • Different data format, depends on who filled the data report
    • No system architecture documents available
The slope of the trend line is -0.76 SLOC/PH per build.
Across the five builds, this corresponds to a 14% average decline in productivity per build. This is smaller than the 20% Incremental Development Productivity Decline (IDPD) factor for a large defense program.
Most likely because the project is considerably smaller in system size and complexity.

### Quality Management Platform (QMP) Project
#### Data Analysis – Productivity Trends

<table>
<thead>
<tr>
<th>Build</th>
<th>Size (KSLOC)</th>
<th>Effort (PH)</th>
<th>Productivity (SLOC/PH)</th>
<th>Productivity Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>69.12</td>
<td>7195</td>
<td>9.61</td>
<td>0.0%</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>9080</td>
<td>7.05</td>
<td>-26.6%</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>6647</td>
<td>2.86</td>
<td>-59.4%</td>
</tr>
<tr>
<td>4</td>
<td>39.2</td>
<td>8787.5</td>
<td>4.46</td>
<td>56.1%</td>
</tr>
<tr>
<td>5</td>
<td>207</td>
<td>31684.5</td>
<td>6.53</td>
<td>46.5%</td>
</tr>
<tr>
<td>6</td>
<td>65</td>
<td>16215.1</td>
<td>4.01</td>
<td>-38.6%</td>
</tr>
</tbody>
</table>

![Productivity Trend Diagram](image)
Exploration of IDPD Factor
Explorations of the IDPD factors on several projects, the following sources of variation were identified:

<table>
<thead>
<tr>
<th>Higher IDPD (Less Productive)</th>
<th>Lower IDPD (More Productive)</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></td>
</tr>
<tr>
<td>Next increment has more previous increments to integrate/interact with</td>
<td>Next increment touches less of previous increments</td>
</tr>
<tr>
<td>Staff turnover reduces experience level</td>
<td>Current staff more experienced, productive</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>
IDPD Ranges

- Based on experience with similar projects, the following impact causes and ranges are conservatively stated:
  - Savings due to more experienced personnel, assuming reasonable initial experience level (5-20%)
    - Variation depending on personnel turnover rates
  - Increases due to code base growth and deferred complexity
    - Breakage, maintenance of full code base (20-40%)
    - Diseconomies of scale in development, integration (10-25%)
    - Requirements volatility; user requests (10-25%)
  - Best case: 20% (from above -20+20+10+10) more effort (IDPD=6%)
  - Worst case: 85% (from above -5+40+25+25) (IDPD=23%)
- In any case, with fixed staff size, there would be either a schedule increase or incomplete builds.
Effects of IDPD on Number of Increments

- Model relating productivity decline to number of builds needed to reach 8M SLOC Full Operational Capability
- Assumes Build 1 production of 2M SLOC @ 100 SLOC/PM
  - 20000 PM/ 24 mo. = 833 developers
  - Constant staff size for all builds
- Analysis varies the productivity decline per build
  - Extremely important to determine the incremental development productivity decline (IDPD) factor per build
Conclusion & Future Work

• Design deficiency and code breakage causes productivity declines
  – If the original design is insufficient to accommodate additional modules, and a re-architecting effort was necessary to put this project back on track
  – Inserting new code into the previous build adds effort to read, analyze, and test both the new and old code in order to ensure nothing is broken, this extra effort may be mitigated by experienced staff

• Using COCOMO II Cost Drivers to normalize new size and effort by:
  – Product Effort Multipliers
  – Personnel Effort Multipliers

• Calibrate Equivalent New Size
  – Calculate equivalent new size based on CodeCount™ “Diff” for each increment and compare that with actual size
  – Use the results to adjust parameters for calculating equivalent new size with integration rework consideration
Q & A

• Questions?
• Comments?
• Thank you very much