COOSS: An Initial COCOTS Extension Model for Estimating Cost of Integrating Open Source Software Components

Lin Shi, Celia Chen, Qing Wang, Barry Boehm
Center of Systems and Software Engineering, University of Southern California
Institute of Software Chinese Academy of Sciences
Agenda

Background  Motivation  COOSS Overview

Reference  Future Work  Conclusion
Over 17 billion open source components downloaded from public repositories in 2014

At least 70 percent of new enterprise Java applications will be deployed on an open source Java application server by the end of 2017.

by Gartner reports
By 2016 the vast majority of mainstream IT organizations will leverage open source software (OSS) components in mission-critical IT solutions.

by Gartner reports

41.70% of people plan to deploy an Open Source solution in 1-2 years
> 56% of companies using OSS will collaborate with competitors
> 50% of all purchased software will be Open Source in 5 years

by Black Duck Software and North Bridge Partner’s survey (2015)
<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• No cost</td>
<td>• Need very experienced staff to integrate</td>
</tr>
<tr>
<td>• Extensive community of developers involved</td>
<td>• Vulnerable to threats</td>
</tr>
<tr>
<td>• Source code is readily available</td>
<td>• Potential high support and maintenance costs</td>
</tr>
<tr>
<td>• Constantly being updated</td>
<td>• Need right level of expertise to manage</td>
</tr>
<tr>
<td>• Problems/bugs are quickly rectified</td>
<td>• Environment/platform incompatible</td>
</tr>
</tbody>
</table>
Unhealthy/unsupportive OSS community: 97%
Licensing issues: 92%
Lack of /low quality documentation: 90%
Function and non-function mismatches: 72%
Interoperability issues: 69%
Poor quality of OSS software: 61%
High maintenance efforts after integrating: 53%
Lack of internal expertise: 48%
High learning efforts and cost: 43%
Lack of vendor support: 30%

Data source: Literature reviews on 32 papers on the topics of OSS component integration.

Top 10 OSS integration challenges found in literature reviews
Trade-offs

Cost
- Support Cost
- Learning Cost
- Maintenance Cost
How to estimate the costs?
COCOTS

- **CO**nstructive **COTS** integration cost model
- Extensive for COCOMO
- Focuses on COTS integration into in-house applications
- Has 4 levels of effort source
<table>
<thead>
<tr>
<th>COTS</th>
<th>OSS</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/50" alt="x" /></td>
<td><img src="https://via.placeholder.com/50" alt="✓" /></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/50" alt="?" /></td>
<td><img src="https://via.placeholder.com/50" alt="✓" /></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/50" alt="✓" /></td>
<td><img src="https://via.placeholder.com/50" alt="?" /></td>
</tr>
</tbody>
</table>
COOSS Overview

- Constructive OSS integration cost model
- 4 effort submodels
  - Assessment Effort, Customizing Effort, Glue Code Effort, Volatility Effort
- Total Effort = Sum of 4 sub efforts
## COCOTS v.s. COOSS

<table>
<thead>
<tr>
<th>Submodels</th>
<th>COCOTS</th>
<th>COOSS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Assessment Effort</strong></td>
<td>Rigorous: Payment Average Effort for two passes</td>
<td>Mild: Free license Average Effort</td>
</tr>
<tr>
<td><strong>Tailoring/Customizing Effort</strong></td>
<td>Black box: Configure Average Effort</td>
<td>White box: Modify and Customize; COCOMO II Reuse Model</td>
</tr>
<tr>
<td><strong>Glue Code Effort</strong></td>
<td>COTS effort multipliers</td>
<td>OSS Cost Drivers</td>
</tr>
<tr>
<td><strong>Volatility Effort</strong></td>
<td>COTS effort multipliers</td>
<td>OSS Cost Drivers</td>
</tr>
</tbody>
</table>
Assessment Effort

- Initial evaluation
- Selection aspects:
  - Functional requirements - Fitness/Capability offered
  - Non-functional requirements – Performance/Reliability/Usability/Maintainability
  - Service Quality – Community support
- Effort model:

\[
\text{Assessment Effort} = (\# \text{ Candidate OSS componets}) \sum_{\text{class}} (\text{Average assessment effort for attribute in class}) \times (\#\text{attributes})
\]
<table>
<thead>
<tr>
<th>Assessment Attributes</th>
<th>Openness</th>
<th>Completeness</th>
<th>Clarity</th>
<th>Validity</th>
<th>Precedent</th>
<th>Scale</th>
<th>Reliability</th>
<th>Safety</th>
<th>Security</th>
<th>Understandability</th>
<th>Performance</th>
<th>Functionality</th>
<th>Environment</th>
<th>Project History</th>
<th>Community Activities</th>
<th>Internal Expertise</th>
<th>Time and Schedule</th>
<th>Licensing</th>
</tr>
</thead>
</table>
## Assessment Checklist Example

<table>
<thead>
<tr>
<th>Community Environment</th>
<th>Project History</th>
</tr>
</thead>
<tbody>
<tr>
<td>· How long does the OSS project exist?</td>
<td></td>
</tr>
<tr>
<td>· How often do new release come out?</td>
<td></td>
</tr>
<tr>
<td>· How many stable releases are there?</td>
<td></td>
</tr>
<tr>
<td>· How recent is the last one release?</td>
<td></td>
</tr>
<tr>
<td>· Does the project offer a separate cutting-edge and stable release cycle?</td>
<td></td>
</tr>
<tr>
<td>· A project that appears dormant for years is a bad sign: developers might have lost interest and abandoned it?!</td>
<td></td>
</tr>
<tr>
<td>· Do developers fix existing bugs, or just piling one new flashy features?</td>
<td></td>
</tr>
<tr>
<td>· Do they respect their user base, or do they break backward compatibility with each new release?</td>
<td></td>
</tr>
<tr>
<td>· Is the project’s direction compatible with yours?</td>
<td></td>
</tr>
<tr>
<td>· Is there a real community behind the project, or will you tie the knot with a one-man show?</td>
<td></td>
</tr>
<tr>
<td>· Is the community working together as a team or constantly fighting?</td>
<td></td>
</tr>
<tr>
<td>· Do developers cooperate under a well-defined democratic process, or will you depend on the whims of an autocrat?</td>
<td></td>
</tr>
<tr>
<td>· Are the users supportive, answering questions, and going out of their way to make newcomers feel welcome, or are they insular, arrogant, and rude?</td>
<td></td>
</tr>
</tbody>
</table>
Customizing Effort

- **White box**
- **Effort Model**: COCOMO II Reuse model
  - Software Understanding Increment SU
  - Cost drivers

\[
\text{Equivalent KSLOC} = \text{Adapted KLOC} \times \left(1 - \frac{AT}{100}\right) \times AAM
\]

\[
AAM = \left\{
\begin{array}{ll}
\frac{AA + AAF(1 + 0.02 \times SU \times UNFM)}{100}, & \text{for } AAF \leq 50 \\
\frac{AA + AAF + SU \times UNFM}{100}, & \text{for } AAF > 50
\end{array}
\right.
\]

\[
PM = A \times \text{Size}^E \times \prod_{i} EM_i
\]
## OSS Software Understanding (OSU)

<table>
<thead>
<tr>
<th></th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure</strong></td>
<td>Very low cohesion, high coupling, spaghetti code</td>
<td>Moderately low cohesion, high coupling</td>
<td>Reasonably well-structured; some weak areas</td>
<td>High cohesion, low coupling</td>
<td>Strong modularity, information hiding in data / control structures</td>
</tr>
<tr>
<td><strong>Application Clarity</strong></td>
<td>No match between program and application world-views</td>
<td>Some correlation between program and application</td>
<td>Moderate correlation between program and application</td>
<td>Good correlation between program and application</td>
<td>Clear match between program and application world-views</td>
</tr>
<tr>
<td><strong>Code commentary</strong></td>
<td>Obscure code</td>
<td>Some code commentary</td>
<td>Moderate level of code commentary</td>
<td>Good code commentary</td>
<td>Self-descriptive code; Useful examples/samples</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
<td>Documentation missing, obscure or obsolete</td>
<td>Some useful documentation</td>
<td>Moderate level of documentation</td>
<td>Useful documentation</td>
<td>Complete, readable, and well-organized documentation; User manual; Technical documentation that help building and modifying support</td>
</tr>
<tr>
<td><strong>Community Support</strong></td>
<td>One-man show/No real community behind</td>
<td>A few active members contribute for the community, slow response</td>
<td>Moderate community with some active members</td>
<td>Good community</td>
<td>Highly active community behind, quick response on technical questions</td>
</tr>
<tr>
<td><strong>SU Increment</strong></td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCCQ</td>
<td>OSS Component Code Quality</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODCQ</td>
<td>OSS Documentation Quality</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCFC</td>
<td>OSS Components Functional Complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OREL</td>
<td>OSS Components Reliability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCPF</td>
<td>OSS Components Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORMA</td>
<td>OSS Release Maturity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCMA</td>
<td>OSS Community Maturity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCIC</td>
<td>OSS Components Integrator Capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCCP</td>
<td>OSS Components Compliance with platform</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OLDP</td>
<td>OSS library Dependency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OICP</td>
<td>OSS Components Interface Complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Any new written code that link OSS components to the in-house applications.
- Two situations:
  - to facilitate data or information exchange
  - to connect components
Glue Code Effort

Glue Code Effort = \(A \times [(\text{Size}) (1 + \text{OREVOL})]^B \times \prod \text{Effort multipliers}\)

- \(A\) = linear scaling constant
- \(\text{Size}\) = of the glue code in lines of code or function points
- \(\text{OREVOL}\) = Percentage of rework of the glue code due to requirements change or volatility in the OSS components
- \(B\) = an architectural nonlinear scaling factor
- \(\text{Effort multipliers}\) = 11 multiplicative effort adjustment factors with ratings from very low to very high
System Volatility Effort = (application effort) *\{[1+(\text{SOREVOL}/1+\text{REVL})]^E - 1\}*(Effort multipliers)

- application effort = new coding effort
- SOREVOL = Percentage of rework of the glue code due to OSS components volatility
- REVL = Percentage of rework in the system independent of OSS components
- E = 1.01+(COCOMO Scale Factors)
- B = an architectural nonlinear scaling factor
- Effort multipliers = 11 multiplicative effort adjustment factors with ratings from very low to very high
We surveyed a set of 32 papers on the topic of OSS integration and the results provided us with a list of top OSS integration challenges. The top 10 integration challenges we found served as a starting point to come up with COOSS model.

Contributions
- Assessment Effort Submodel: OSS components assessment attributes/checklists
- Software Understanding for OSS components
- 11 cost drivers
Future Work

- Construct rating criteria for the eleven effort multipliers
- Conduct Survey to collect multiplier values
- Improve the cost driver and scale factors
- Collecting OSS components integration data to evaluate the estimation
  - OSS projects
  - Students projects


Reference


