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

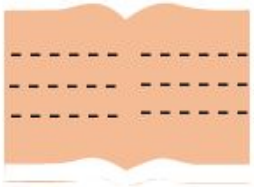
Lin Shi, Celia Chen, Qing Wang, Barry Boehm
Center of Systems and Software Engineering, University of Southern California
Institute of Software Chinese Academy of Sciences

Background

Motivation

COOSS Overview

Agenda



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

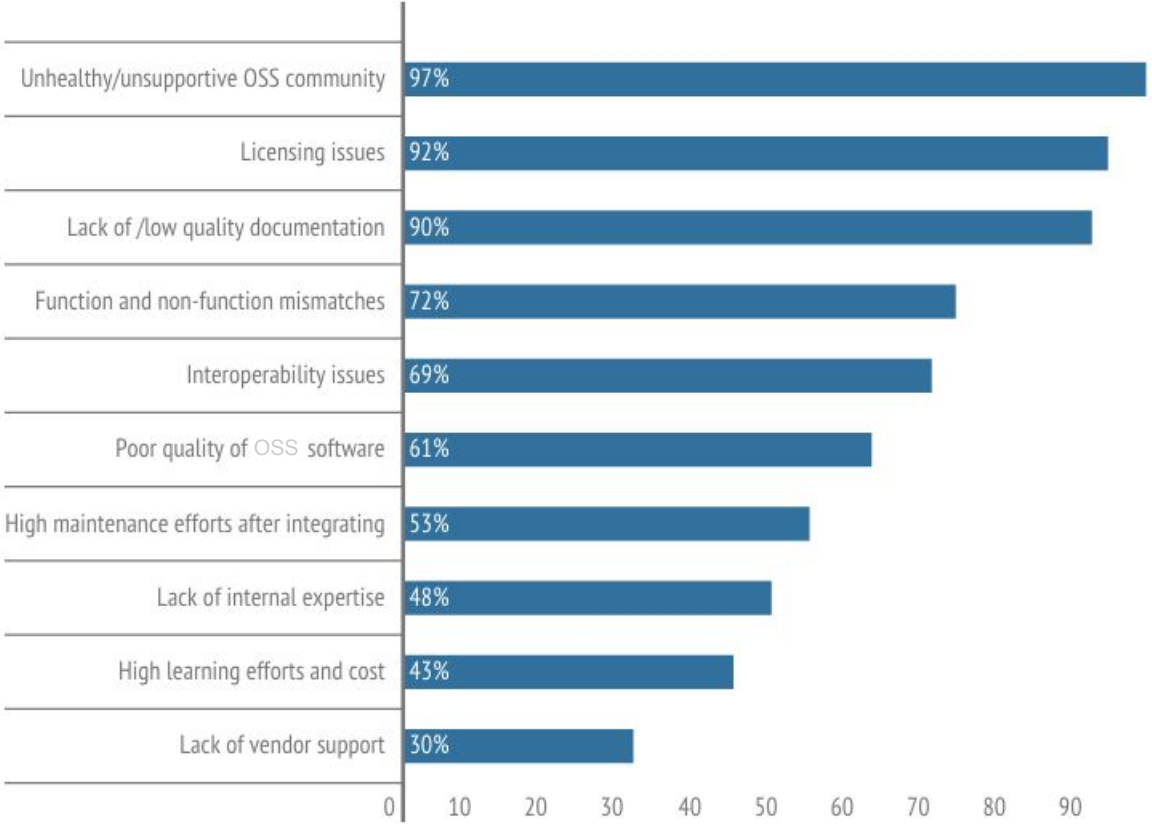
by Black Duck Software and North Bridge Partner's survey (2015)

Advantages

- No cost
- Extensive community of developers involved
- Source code is readily available
- Constantly being updated
- Problems/bugs are quickly rectified

Disadvantages

- Need very experienced staff to integrate
- Vulnerable to threats
- Potential high support and maintenance costs
- Need right level of expertise to manage
- Environment/platform incompatible



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

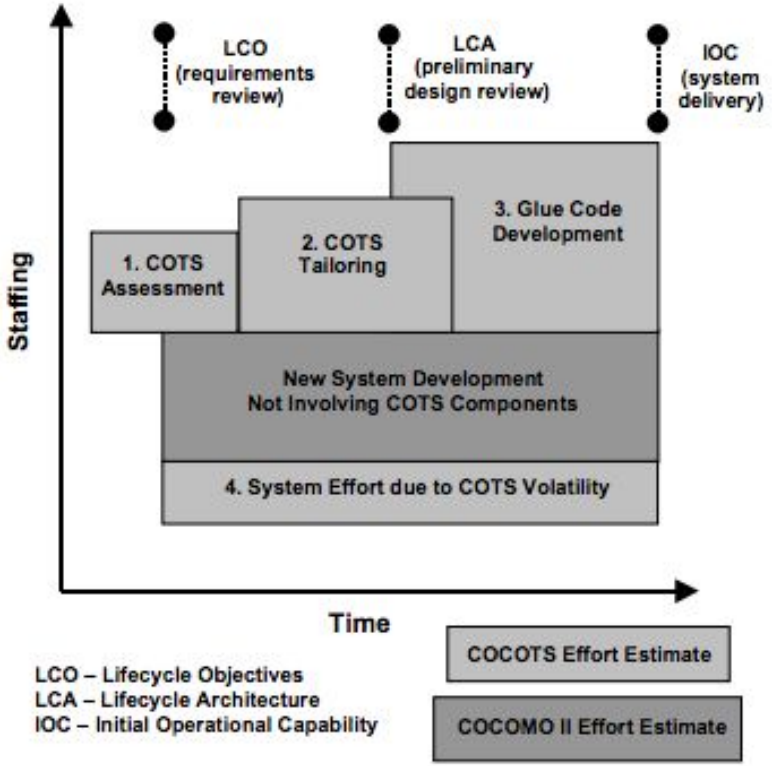
How to estimate the costs?

Inspiration

COCOTS

- **CO***nstructive* **COTS** *integration cost model*
- Extensive for COCOMO

- Focuses on COTS integration into in-house applications
- Has 4 levels of effort source



COTS		OSS
	Source Code	
	Future Evolution	
	Extensive Support	

COOSS Overview

- **CO***nstructive* **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

Submodels	COCOTS	COOSS
Assessment Effort	Rigorous: Payment Average Effort for two passes	Mild: Free license Average Effort
Tailoring/Customiz-ing Effort	Black box: Configure Average Effort	White box: Modify and Customize; COCOMO II Reuse Model
Glue Code Effort	COTS effort multipliers	OSS Cost Drivers
Volatility Effort	COTS effort multipliers	OSS Cost Drivers

Assessment Effort

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

$$\text{Assessment Effort} = (\#\ Candidate\ OSS\ componets) \sum_{class} (Average\ assessment\ effort\ for\ attribute\ in\ class)(\#attributes)$$

Assessment Attributes

Product Engineering	Requirements/Features	Openness
		Completeness
		Clarity
		Validity
		Precedent
		Scale
		Reliability
		Safety
		Security
	Design and Integration	Understandability
		Performance
		Functionality
		Environment
Community Environment	Project History	
	Community Activities	
Program Constraint	Resource	Internal Expertise
		Time and Schedule
		Licensing

Assessment Checklist Example

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

Customizing Effort

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

$$Equivalent\ KSLOC = Adapted\ KLOC * (1 - \frac{AT}{100}) * AAM$$

$$AAF = 0.4 * DM + 0.3 * CM + 0.3 * IM$$

$$AAM = \begin{cases} \frac{AA + AAF(1 + 0.02 * SU * UNFM)}{100}, & for\ AAF \leq 50 \\ \frac{AA + AAF + SU * UNFM}{100}, & for\ AAF > 50 \end{cases}$$

$$PM = A * Size^{E} * \prod_{i}^{p} EM_i$$

OSS Software Understanding (OSU)

	Very Low	Low	Nominal	High	Very High
Structure	Very low cohesion, high coupling, spaghetti code	Moderately low cohesion, high coupling	Reasonably well-structured; some weak areas	High cohesion, low coupling	Strong modularity, information hiding in data / control structures
Application Clarity	No match between program and application world-views	Some correlation between program and application	Moderate correlation between program and application	Good correlation between program and application	Clear match between program and application world-views
Code commentary	Obscure code	Some code commentary	Moderate level of code commentary	Good code commentary	Self-descriptive code; Useful examples/samples
Documentation	Documentation missing, obscure or obsolete	Some useful documentation	Moderate level of documentation	Useful documentation	Complete, readable, and well-organized documentation; User manual; Technical documentation that help building and modifying support
Community Support	One-man show/No real community behind	A few active members contribute for the community, slow response	Moderate community with some active members	Good community	Highly active community behind, quick response on technical questions
SU Increment	50	40	30	20	10

Cost Drivers

OCCQ	OSS Component Code Quality
ODCQ	OSS Documentation Quality
OCFC	OSS Components Functional Complexity
OREL	OSS Components Reliability
OCPF	OSS Components Performance
ORMA	OSS Release Maturity
OCMA	OSS Community Maturity
OCIC	OSS Components Integrator Capability
OCCP	OSS Components Compliance with platform
OLDP	OSS library Dependency
OICP	OSS Components Interface Complexity

Glue Code Effort

- 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

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

- A = linear scaling constant
- Size = of the glue code in lines of code or function points
- OREVOL = Percentage of rework of the glue code due to requirements change or volatility in the OSS components
- B = an architectural nonlinear scaling factor
- Effort multipliers = 11 multiplicative effort adjustment factors with ratings from very low to very high

System Volatility

*System Volatility Effort = (application effort) *{[1+ (SOREVOL/1+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

CONCLUSION

- 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

1. Abts, Chris, Barry W. Boehm, and Elizabeth Bailey Clark. "COCOTS: A COTS software integration lifecycle cost model-model overview and preliminary data collection findings." ESCOM-SCOPE Conference. 2000.
2. Majchrowski, Annick, and Jean-Christophe Deprez. "An operational approach for selecting open source components in a software development project."Software Process Improvement. Springer Berlin Heidelberg, 2008. 176-188.
3. Boehm, Barry W., Ray Madachy, and Bert Steece. Software cost estimation with Cocomo II with Cdrom. Prentice Hall PTR, 2000.
4. Golden, Bernard. "Open Source Maturity Model." *The Open Source Business Resource* (2008): 4.
5. Ven, Kris, and Jan Verelst. "The importance of external support in the adoption of open source server software." *Open Source Ecosystems: Diverse Communities Interacting*. Springer Berlin Heidelberg, 2009. 116-128.
6. Merilinna, Janne, and Mari Matinlassi. "State of the art and practice of opensource component integration." *Software Engineering and Advanced Applications, 2006. SEAA'06. 32nd EUROMICRO Conference on*. IEEE, 2006.
7. Chen, Weibing, et al. "An empirical study on software development with open source components in the chinese software industry." *Software Process: Improvement and Practice* 13.1 (2008): 89-100.
8. Ruffin, Michel, and Christof Ebert. "Using open source software in product development: A primer." *Software, IEEE* 21.1 (2004): 82-86.
9. Hauge, Øyvind, Carl-Fredrik Sørensen, and Andreas Røsdal. "Surveying industrial roles in open source software development." *Open source development, adoption and innovation*. Springer US, 2007. 259-264.
10. Ayala, Claudia, et al. "Challenges of the open source component marketplace in the industry." *Open Source Ecosystems: Diverse Communities Interacting*. Springer Berlin Heidelberg, 2009. 213-224.
11. Krivoruchko, Jacob. "The use of open source software in enterprise distributed computing environments." *Open Source Development, Adoption and Innovation*. Springer US, 2007. 277-282.
12. Tiangco, Francis, et al. "Open-source software in an occupational health application: the case of Heales Medical Ltd." *Procs* (2005).Agerfalk, Par J., et al. "Assessing the role of open source software in the European secondary software sector: a voice from industry." (2005).
13. Jaaksi, Ari. "Experiences on product development with open source software."*Open source development, adoption and innovation*. Springer US, 2007. 85-96.
14. Bac, Christian, et al. "Why and how to contribute to libre software when you integrate them into an in-house application." *Proceedings of the First International Conference on Open Source Systems*. 2005.

Reference

15. Bac, Christian, et al. "Why and how to contribute to libre software when you integrate them into an in-house application." *Proceedings of the First International Conference on Open Source Systems*. 2005.
16. Mannaert, Herwig, and Kris Ven. "The use of open source software platforms by Independent Software Vendors: issues and opportunities." *ACM SIGSOFT Software Engineering Notes*. Vol. 30. No. 4. ACM, 2005.
17. Ven, Kris, and Herwig Mannaert. "Challenges and strategies in the use of open source software by independent software vendors." *Information and Software Technology* 50.9 (2008): 991-1002.Kotonya, Gerald, and Awais Rashid. "A strategy for managing risk in component-based software development." *Euromicro Conference, 2001. Proceedings. 27th*. IEEE, 2001.
18. Garlan, David, Robert Allen, and John Ockerbloom. "Architectural mismatch: Why reuse is so hard." *IEEE software* 6 (1995): 17-26.
19. Anh, Nguyen Duc, et al. "Collaborative resolution of requirements mismatches when adopting open source components." *Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2012. 77-93.
20. Spinellis, Diomidis. "Choosing and using open source components." *IEEE Software* 3 (2011): 96.
21. Stol, Klaas-Jan, et al. "A comparative study of challenges in integrating Open Source Software and Inner Source Software." *Information and Software Technology* 53.12 (2011): 1319-1336.
22. Sung, Won Jun, Ji Hyeok Kim, and Sung Yul Rhew. "A quality model for open source software selection." *Advanced Language Processing and Web Information Technology, 2007. ALPIT 2007. Sixth International Conference on*. IEEE, 2007.
23. Adewumi, Adewole, Sanjay Misra, and Nicholas Omoregbe. "A review of models for evaluating quality in open source software." *IERI Procedia* 4 (2013): 88-92.
24. Sarrab, Mohamed, and Osama M. Hussain Rehman. "Empirical study of open source software selection for adoption, based on software quality characteristics." *Advances in Engineering Software* 69 (2014): 1-11.
25. Majchrowski, Annick, and Jean-Christophe Deprez. "An operational approach for selecting open source components in a software development project."*Software Process Improvement*. Springer Berlin Heidelberg, 2008. 176-188.