Topic 5: Evaluating OSS projects

Even Wiik Thomassen

Norwegian University of Science and Technology
Department of Computer and Information Science

September 29, 2011

Papers

1. Evaluation criteria for free/open source software products based on project analysis – 2006
2. Assessing the Health of Open Source Communities – 2006
3. The Evolution of Open Source Software using Eclipse Metrics – 2009
4. Apache and Eclipse: Comparing open source project incubators – 2006

Paper 1: Introduction

Paper

"Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis" by David Cruz, Thomas Wieland and Alexander Ziegler.

Paper 1: Introduction

Paper

"Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis" by David Cruz, Thomas Wieland and Alexander Ziegler.

Why Make the product decision process more transparent and deterministic

Paper 1: Introduction

Paper

"Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis" by David Cruz, Thomas Wieland and Alexander Ziegler.

Why Make the product decision process more transparent and deterministic

How Provides a systematic approach for evaluating and interpreting F/OSS products

Paper 1: Introduction

Paper

"Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis" by David Cruz, Thomas Wieland and Alexander Ziegler.

Why	Make the product decision process more transparent and deterministic
How	Provides a systematic approach for evaluating and interpreting F/OSS products
Who	Both product manager/evaluator and F/OSS developer/manager

Various Usage Scenarios for F/OSS

1. As a platform for a mission critical process.
2. With a long-term consideration.
3. As a cost reduction model.
4. As exploration object, for example, for technology.
5. As an exhibition prototype
6. As a base line for further development and business model.
7. To bridge a temporary bottleneck.
8. For becoming independent of proprietary solutions and providers.
9. To gain transparency concerning safety and security.
10. For research purposes.
11. As a CASE tool.

Requirements Categories

Categories

Paper defines 6 categories, their criteria and fulfilment evaluation.

Requirements Categories

Categories

Paper defines 6 categories, their criteria and fulfilment evaluation.

1. Functional requirements
2. Technical requirements
3. Organisational requirements
4. Legal requirements
5. Economical requirements
6. Political requirements

Functional Requirements

1. Required functionality covered
2. Clear direction of product evolution

Functional Requirements

1. Required functionality covered
2. Clear direction of product evolution

- Project description often has a feature list.
- Which can be incomplete, or wrong.
- Run demos, test suites, analyze reference installations, source code.
- For confirmed missing features, estimate expenses for adding one's own implementation.

Technical Requirements

1. Target platforms supported
2. Reliability
3. Maintainability

Technical Requirements

1. Target platforms supported
2. Reliability
3. Maintainability

- Actual number of bugs – Public bug tracker. Hard to evaluate.
- Number of open feature requests – Public bug tracker.
- Code metrics. – Automatic tools. Relationship between code lines and comments. Average code lines per function.
- Frequency of changes – Source repository logs.
- Dependencies on other software – Often found in README file.

Organisational Requirements

1. Community exists
2. Product evolution
3. Sufficient support
4. Long life existence
5. Compatible development process

Organisational Requirements

1. Community exists
2. Product evolution
3. Sufficient support
4. Long life existence
5. Compatible development process

- Number of developers – Project page often list developers.
- Number of testers – Public Bug Tracker.
- Number of users – Can be estimated from download numbers.
- Product evolution – Monitor changes to source repository.
- Support – Documentation and the climate in discussion forums or mailing lists.

Legal Requirements

1. No copyleft effect
2. No liability for third party code
3. No patent infringements

Legal Requirements

1. No copyleft effect
2. No liability for third party code
3. No patent infringements

- Check license, websites which help evaluate known licenses.
- Liabilities depends on laws in relevant countries.
- Patents almost impossible to evaluate, check out other users.

Economical Requirements

1. Sustainability of the usage of the F/OSS
2. Protection of investment of migration
3. Increase productivity by usage
4. Quick availability, easy to download
5. Cost reduction by using the F/OSS product
6. Division of development costs

Economical Requirements

1. Sustainability of the usage of the F/OSS
2. Protection of investment of migration
3. Increase productivity by usage
4. Quick availability, easy to download
5. Cost reduction by using the F/OSS product
6. Division of development costs

- Estimate resources contributed by other users.
- If large, reputable companies contribute to the project, sustainability is likely.
- Estimate cost of migrate to, train the staff for, and roll out the F/OSS product.
- Estimated monthly costs.

Political Requirements

1. Possibility for influencing further development
2. Decrease of proprietary dependencies
3. Transparency over security
4. Publicity and marketing effects

Political Requirements

1. Possibility for influencing further development
2. Decrease of proprietary dependencies
3. Transparency over security
4. Publicity and marketing effects

- Discover which well-known companies support the project.
- How is the user documentation, and in which languages.
- Investigate the climate on project forums and mailing lists.
- Investigate political changes in large contributing companies.

Summary and Conclusion

- The paper presented a systematic and deterministic approach for evaluating F/OSS products.
- Important to notice that the approach is not an automated decision system.
- It is only a guideline. Decision makers has to adapt to the circumstances and needs of his company.

Paper 2: Introduction

Paper

"Assessing the Health of Open Source Communities" by Kevin Crowston and James Howison

Paper 2: Introduction

Paper

"Assessing the Health of Open Source Communities" by Kevin Crowston and James Howison

Why Its difficult to understand FLOSS communities.

Paper 2: Introduction

Paper

"Assessing the Health of Open Source Communities" by Kevin Crowston and James Howison

Why Its difficult to understand FLOSS communities.

How Approach and tools to research the community of developers, leaders, and active users behind FLOSS.

Paper 2: Introduction

Paper

"Assessing the Health of Open Source Communities" by Kevin Crowston and James Howison

Why	Its difficult to understand FLOSS communities.
How	Approach and tools to research the community of developers, leaders, and active users behind FLOSS.
Who	Those who rely, recommend or want to contribute to a FLOSS project.

Life cycle and motivations

FLOSS projects life cycle

Often start alone or a small group (Cathedral). Then creative explosion where it develops quickly, gathers features and capabilities that in turn attract additional developers and users.

Life cycle and motivations

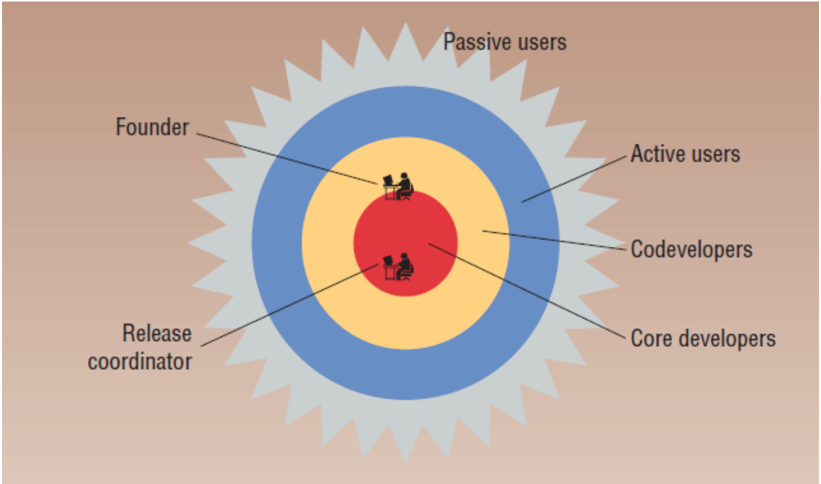
FLOSS projects life cycle

Often start alone or a small group (Cathedral). Then creative explosion where it develops quickly, gathers features and capabilities that in turn attract additional developers and users.

Motivations for efforts in FLOSS projects:

1. Intellectual engagement
2. Knowledge sharing
3. The product itself
4. Ideology, reputation, and community obligation

A FLOSS project community



Community health and development processes

Community health:

- Less than 1% of projects exceeded 10 developers.
- Projects with hundreds of developers are the exception rather than the rule.

Community health and development processes

Community health:

- Less than 1% of projects exceeded 10 developers.
- Projects with hundreds of developers are the exception rather than the rule.

Development processes:

- Rarely formally documented.
- Often lack detailed roadmap.
- Organizing for fun can be more important than organizing for efficiency.

Summary and Conclusion

- Try to understand the community in addition to the code.
- Examine project's homepage, mailing lists, IRC channel.
- Use tools to inspect communities evolution. FLOSSmole, CVSAnalY and OpenBRR.
- Trying to change an existing community is likely to end in frustration.

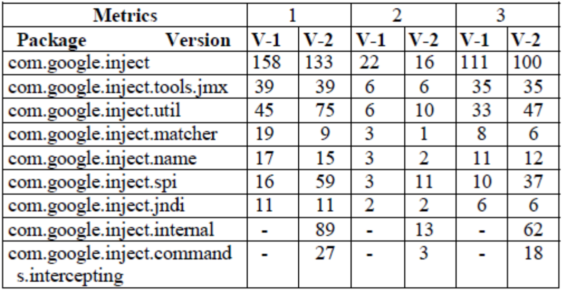
Paper 3: Introduction

Paper

"The Evolution of Open Source Software using Eclipse Metrics" by Ajlan Al-Ajlan

Why Effective management of software evolution is crucial for organizations success and to stay competetive.

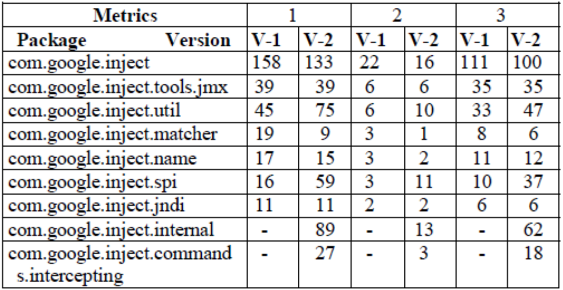
Metrics in Google Guice



Metrics		1		2		3	
Package	Version	V-1	V-2	V-1	V-2	V-1	V-2
com.google.inject		158	133	22	16	111	100
com.google.inject.tools.jmx		39	39	6	6	35	35
com.google.inject.util		45	75	6	10	33	47
com.google.inject.matcher		19	9	3	1	8	6
com.google.inject.name		17	15	3	2	11	12
com.google.inject.spi		16	59	3	11	10	37
com.google.inject.jndi		11	11	2	2	6	6
com.google.inject.internal		-	89	-	13	-	62
com.google.inject.command s.intercepting		-	27	-	3	-	18

1. Standard Lines of Code in Method
2. Standard Cyclomatic Complexity Metric
3. Standard Number of Statements Metric

Metrics in Google Guice



Metrics	1		2		3	
Package Version	V-1	V-2	V-1	V-2	V-1	V-2
com.google.inject	158	133	22	16	111	100
com.google.inject.tools.jmx	39	39	6	6	35	35
com.google.inject.util	45	75	6	10	33	47
com.google.inject.matcher	19	9	3	1	8	6
com.google.inject.name	17	15	3	2	11	12
com.google.inject.spi	16	59	3	11	10	37
com.google.inject.jndi	11	11	2	2	6	6
com.google.inject.internal	-	89	-	13	-	62
com.google.inject.commands.intercepting	-	27	-	3	-	18

1. Standard Lines of Code in Method
2. Standard Cyclomatic Complexity Metric
3. Standard Number of Statements Metric

He never explains what actual versions V-1 and V-2 refers to, or the time between their release!

Excerpts from the paper

This report has discussed and addressed the key subjects relating to OSS and OSSE, including their types, tools, requirements, definitions, advantages and limitations, and the reusability of code.

Excerpts from the paper

This report has discussed and addressed the key subjects relating to OSS and OSSE, including their types, tools, requirements, definitions, advantages and limitations, and the reusability of code.

measuring software evolution . . . will disclose the errors and assist developers in correcting and avoiding them in the future.

Excerpts from the paper

This report has discussed and addressed the key subjects relating to OSS and OSSE, including their types, tools, requirements, definitions, advantages and limitations, and the reusability of code.

measuring software evolution . . . will disclose the errors and assist developers in correcting and avoiding them in the future.

At the moment, the most popular OSS databases are available on the Internet, and they are: 1) GNU, 2) SourceForge, and 3) Freshmeat. The testing projects, listed in Table 1, present some idea of how these OSS database are assessed at present [10]

Summary and Conclusion

- Paper is badly written.
- Repeats himself a lot.
- Makes grand claims but don't back them up.
- Misleading.
- Contains very little actual knowledge.

"I pity the fool who's asked about this paper on the exam!"

Paper 4: Introduction

Paper

"Apache and Eclipse: Comparing Open Source Project Incubators" by Juan C. Dueñas, Hugo A. Parada G., Félix Cuadrado, Manuel Santillán, and José L. Ruiz

Paper 4: Introduction

Paper

"Apache and Eclipse: Comparing Open Source Project Incubators" by Juan C. Dueñas, Hugo A. Parada G., Félix Cuadrado, Manuel Santillán, and José L. Ruiz

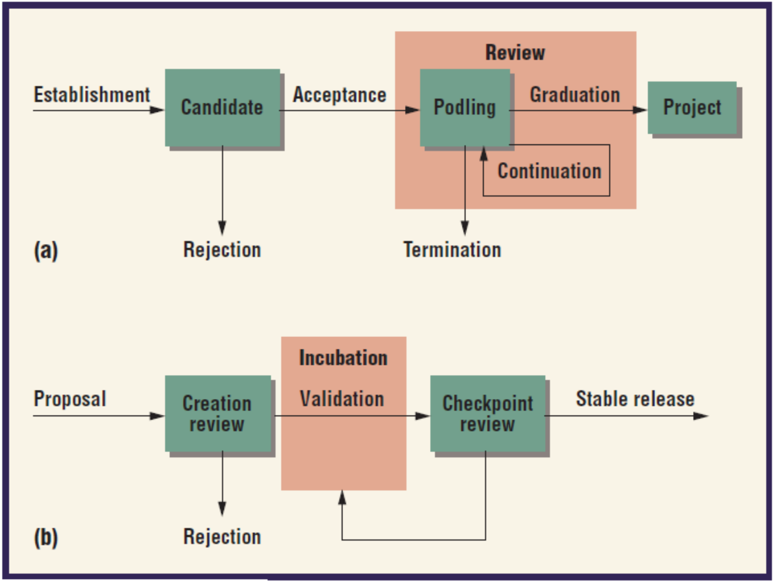
Problem

Open source projects are dying out due to lack of developers.

Incubation

- F/OSS projects need a critical mass of 5 to 15 developers.
- The early phase (kick off phase) is the most critical.
- Incubation helps nurture healthy communities, deliver stable releases and manage initial risks.
- Paper looks at incubation processes at Apache and Eclipse communities.
- Paper proposes best practices for applying this kick-off approach in new F/OSS projects.

Incubation processes



Quantitative analysis

- Number of incubation projects
- Number of projects which graduated
- Incubation start and graduation dates
- Number of committers per project

Quantitative analysis

- Number of incubation projects
- Number of projects which graduated
- Incubation start and graduation dates
- Number of committers per project

- Both differentiate between top-level projects and subprojects.
- Both require candidates to have formally stated objectives.
- Both emphasize projects health by number of committers.
- Both follow an iterative approach which reduces risk with delivering a stable first release.

Apache incubation data

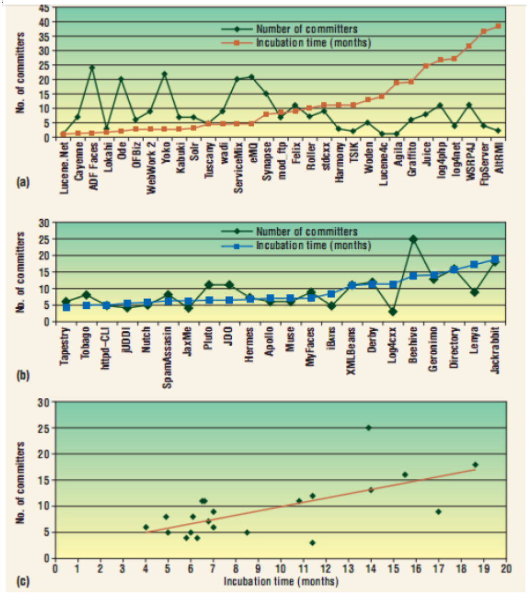


Figure 2. Apache Incubator data:
(a) remaining projects in incubation,
(b) projects that graduated from incubation, and
(c) the regression curve for committers versus time in graduated projects.

Eclipse incubation data

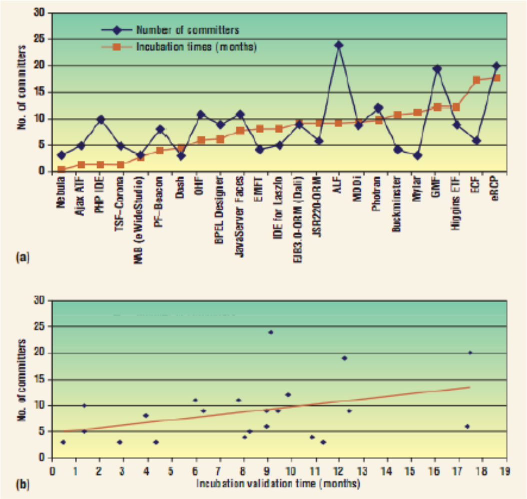


Figure 3. Data from the Eclipse incubation/validation phase: (a) the number of committers and incubation time (months) in projects in the incubation/validation phase, and (b) the regression curve for committers vs. time in projects under incubation/validation.

Summary and Conclusion

- Top level projects require longer incubation time.
- Projects with larger scope require more committers.
- Estimate 6 months for sub-level projects, 13 months for top-level.

Summary and Conclusion

- Top level projects require longer incubation time.
- Projects with larger scope require more committers.
- Estimate 6 months for sub-level projects, 13 months for top-level.

- Paper suggest other incubations have two stages, launch stage and establishment stage.
- Paper concludes successful projects require focused effort by large enough set of aligned committers in a short time.
- Several benefits by joining a community: infrastructure, tools, share code base, get more developers.
- Reduce risk by: define project scope early, launch projects after gathering enough stable committers.
- Incubation might promote bottom-up innovation inside corporations.

The end