

# Apache and Eclipse: Comparing Open Source Project Incubators

**Juan C. Dueñas, Hugo A. Parada G., Félix Cuadrado, Manuel Santillán,  
and José L. Ruiz, *Universidad Politécnica de Madrid***

A two-stage incubation model based on analysis of Apache and Eclipse processes might help other organizations build their own incubation process and better manage risks.

**T**he popularity of free and open source software has boosted the creation of several kinds of communities around F/OSS projects. Successful communities such as GNU, Mozilla, the Apache Software Foundation, and the Eclipse Foundation drive innovation while delivering high-quality, enterprise-class software. Yet, because communities are composed of loosely coordinated contributors mainly guided by practice, no shared agreement exists on a single set of methods and processes.<sup>1-3</sup>

Moreover, the degree of success varies greatly among F/OSS communities, and many projects have died before delivering a first release. They usually failed to create a supporting community—only one or two developers<sup>4,5</sup> had tried to sustain them. These projects tend to go inactive and often vanish completely.<sup>6</sup> In fact, there is some evidence that relates the evolution of F/OSS systems to that of their communities.<sup>7,8</sup> As a general rule, F/OSS projects don't produce viable systems if their teams don't achieve a critical mass of 5 to 15 developers.<sup>3</sup> These facts suggest that kicking off a F/OSS project is one of its most critical phases<sup>9-11</sup>: it requires experience in creating and engaging the community, as well as basic infrastructure and management skills. At the same time, the participants must define the technical scope and domain. Since meritocracy and volunteer effort are two cornerstones of the F/OSS movement, the communi-

ties of recently launched projects grow mainly by subscription.

Recent studies propose the emergence of a new organizational model for F/OSS, called OSS 2.0, based on aggregating communities into ecosystems at some point between the “cathedral” and the “bazaar.”<sup>12</sup> Under this approach, communities adopt some practices of traditional software development processes and organizations, focusing on their long-term survival and better use of resources. However, no one has looked at projects' early phases, when the communities supporting them are born and thus are in a weaker position to face high risks.

To nurture healthy communities, deliver stable releases, and manage initial risks, organizations follow various incubation processes and approaches. We analyzed such processes in two well-known communities; here, we present our results and describe some underlying princi-

ples. We also propose a set of best practices for applying this kick-off approach in newborn F/OSS projects and define a path to future research about the life cycle of open source projects and communities.

## Context

To date, most of F/OSS case study research has focused on low-structure communities with many small projects. SourceForge, combining the notions of community, repository, and project management tool, is a good example of this kind of community. It comprises more than 116,000 projects, with a rough average of two developers per project (see data at the FLOSSmole Project, <http://ossmole.sourceforge.net>). However, consolidation is also taking place in F/OSS, and stronger communities are becoming more relevant. They run fewer but bigger projects and deal with risk more effectively. Two leading F/OSS communities, Eclipse and Apache, are successfully applying incubation processes. Compared to others, their models can be considered intermediate between the cathedral and bazaar paradigms because their processes' degree of formalization is higher.

F/OSS development activities are open, so project information is made public over the Web.<sup>4</sup> However, the artifacts generated in different communities and projects vary significantly, both in quantity and quality. In popular projects, active members of the community continuously update and improve the artifacts; in other projects, information is scarce and outdated.

In our study, we hypothesized that the level of F/OSS community activity is a determining factor in project success. We followed a mixed approach that combined assessing the work from a qualitative perspective and analyzing the generated outputs quantitatively. In this way, we evaluated aspects such as roles, process formalization, Web sites, minutes, relevant email messages, project proposals, and milestone schedules. Concerning the quantitative analysis, we considered variables such as the number of incubated projects, the number of projects that graduated, the remaining projects, incubation start and graduation dates, and the number of committers per project. The statistical population being small, we intentionally avoided analysis techniques aside from basic statistical tools such as average and linear regression to obtain trends that confirm or deny apparent causal relationships; we cal-

culated statistics and project status as of April 2006. Because these are young, dynamic communities, the amount of data will increase, helping to further refine our conclusions.

## Incubator exploration

Concerning the selected communities, Eclipse is an industry-driven initiative in which more than 100 companies, universities, and contributors deliver open source applications around the Eclipse environment. The Apache Software Foundation is a highly successful initiative, made up of individuals, that provides popular, high-quality, open source software. (According to a recent Netcraft survey, more than half of the world's Web sites use its Web server; see <http://survey.netcraft.com/Reports/200708/byserver>). Initially founded around the Apache Web server, the ASF currently provides many other libraries and software infrastructure elements. Both communities follow a mature approach to F/OSS, having established incubation processes to face the problems inherent in project launches.

### The Apache Incubator

The Incubator is the entry path into the ASF for projects and code bases wishing to become part of the Foundation's efforts. As the Apache Web site explains ([www.apache.org/foundation/how-it-works.html](http://www.apache.org/foundation/how-it-works.html)), the Incubator's main responsibilities are to

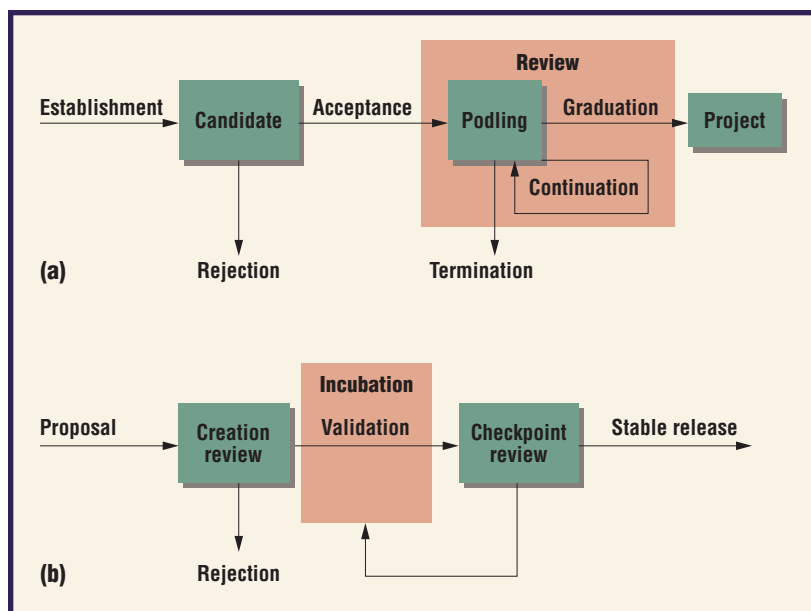
- filter proposals,
- help create projects,
- evaluate an incubated project's maturity, and
- promote the creation of a community that shares the ASF's principles, including meritocracy as one of its central elements.

Two types of projects can graduate: subprojects that have finalized the incubation period and are added into an already existing top-level project, and top-level projects that shape the main lines of the foundation's evolution and can sponsor other incubated subprojects.

**Roles.** The Incubator Project Management Committee supervises the incubation processes. The PMC accepts projects and provides the technical and administrative support required; it regularly reviews incubated projects, proposing them for termination, continuation, or escalation.

In addition to the PMC, some roles are defined per project. Candidates put forward

**The OSS 2.0  
organizational  
model  
aggregates  
F/OSS  
communities  
into ecosystems  
at some point  
between the  
“cathedral” and  
the “bazaar.”**



**Figure 1. Incubation processes: (a) the three Apache phases of establishment, acceptance, and review; and (b) the Eclipse incubation/validation process.**

project proposals. The champion is a foundation officer or member who helps candidates make their initial submission to a sponsor. The sponsor is the ASF entity that defends the project candidate as a worthy contributor and agrees to supervise the candidate in question. Upon the Incubator PMC's acceptance, the candidate becomes a podling (a project in the review phase, before graduation) under its care. The mentor is a permanent ASF member who has specific responsibilities toward the Incubator PMC, the sponsor, and the members of the assigned podling project. For example, the mentor helps, guides, and protects the project on the basis of his or her experience with the process and familiarity with incubation policy and procedures. A mentor keeps timely communication between the PMC and podling members (the community) about the decisions that affect the project; in other words, the mentor is the contact point between the community and the PMC Incubator.

**Process phases.** The process shown in figure 1a has three main phases: establishment, acceptance, and review ([http://incubator.apache.org/incubation/Process\\_Description.html](http://incubator.apache.org/incubation/Process_Description.html)). The establishment phase consists of finding a champion, preparing a proposal, and presenting it. Then, the champion assigns candidate status to the project.

In the acceptance phase, the sponsor assesses the proposal and either accepts or rejects it. If it's accepted, the sponsor proposes

to the Incubator PMC the escalation of the candidate project to podling status, and the sponsor assigns a mentor to the project.

During the iterative review phase, the Incubator PMC assesses project status. Three outcomes are feasible: the project can be terminated, continued (during which the project must take the development recommendations into account), or engaged (the project is accepted into the ASF as either a subproject or a top-level project). The project can deliver releases while it's in the review phase, but the ASF won't endorse the releases because it hasn't fully accepted the podling project as part of the foundation.

### The Eclipse incubator process

The Eclipse development process is formally defined in governance documents ([www.eclipse.org/org/documents](http://www.eclipse.org/org/documents)) and updated on the Web site ([www.eclipse.org/projects/dev\\_process](http://www.eclipse.org/projects/dev_process)). Eclipse views incubation only as the first validation phase within the project life cycle. During this phase, the development process starts and the project's community is born. Thus, the project is firmly established after incubation.

In our research, we assume that both the proposal and the creation review are key stages in this process because, as figure 1b shows, they are the entry point to the incubation/validation phase. This process, in contrast with the Apache Incubator, doesn't use a separate place for the incubation phases. In Eclipse, projects are incubated within the top-level Technology project, concentrating in this area all the project creation experience.

**Roles.** Like the ASF, the Eclipse high-level structure differentiates between standard and top-level projects. Each top-level project has several projects and one PMC with executive authority. Each standard project organization consists of the project lead and the development team (committers and developers), who are respectively responsible for planning the project (through a milestones schedule) and defining the technical architecture. Once the PMC approves the proposal, this structure is established—that is, the roles and responsibilities are assigned in agreement with Eclipse bylaws. Decisions are made by voting; the PMC handles decisions for standard projects, and the Eclipse Management Organization (EMO) does so for top-level projects.

**Process phases.** Figure 1b illustrates the process. The preproposal phase starts with a short declaration of an individual's or company's interest in establishing a project. The first step is to decide whether to graduate the new idea as a top-level or regular project or as part of another project. The next step is to contact the EMO and extend the project proposal.

The creation review consists of a short presentation containing a brief proposal summary, a report about the community response to the proposal, current project participants, initial implementation focus, confirmation that the project members have read and understood the Eclipse development process, and the guidelines and future directions. As a result of the creation review, the project can be accepted, rejected, or conditionally accepted (the EMO can give comments). If it's approved, the project establishes an infrastructure and the PMC nominates the initial committers for EMO approval; if comments are made, the proposal must take them into account before presenting the updated creation review. Once the project passes creation review, it's formally created and the project structure is adopted.

The validation phase deals with issues such as identifying critical use cases, producing a high-level design (architecture), acquiring all the necessary intellectual-property rights, and establishing the committer community around the project. This phase ends with a checkpoint review that evaluates the proposal in detail to check that

- a working and demonstrable code base exists,
- the community is active,
- the project is operating fully in the open using open source rules of engagement,
- the project adopts Eclipse's philosophy and principles, and
- an in-depth review of the project's technical architecture takes into account its dependencies and interactions with other projects.

The project leader asks the EMO for a checkpoint review when the project leader believes the project meets the exit criteria. If the EMO approves the checkpoint review, the project delivers a stable release and the incubation phase ends.

## Quantitative analysis

Once we identified each community's principles, roles, and phases, we studied their incubators' behavior by applying simple statistic analysis on the available data.

### Apache

Since the Incubator's creation, the ASF has incubated 55 projects, of which 22 have successfully graduated, 31 are still waiting for graduation, and two were terminated. To assess the incubation process, we separated the projects into the graduated ones and those still in the Incubator.

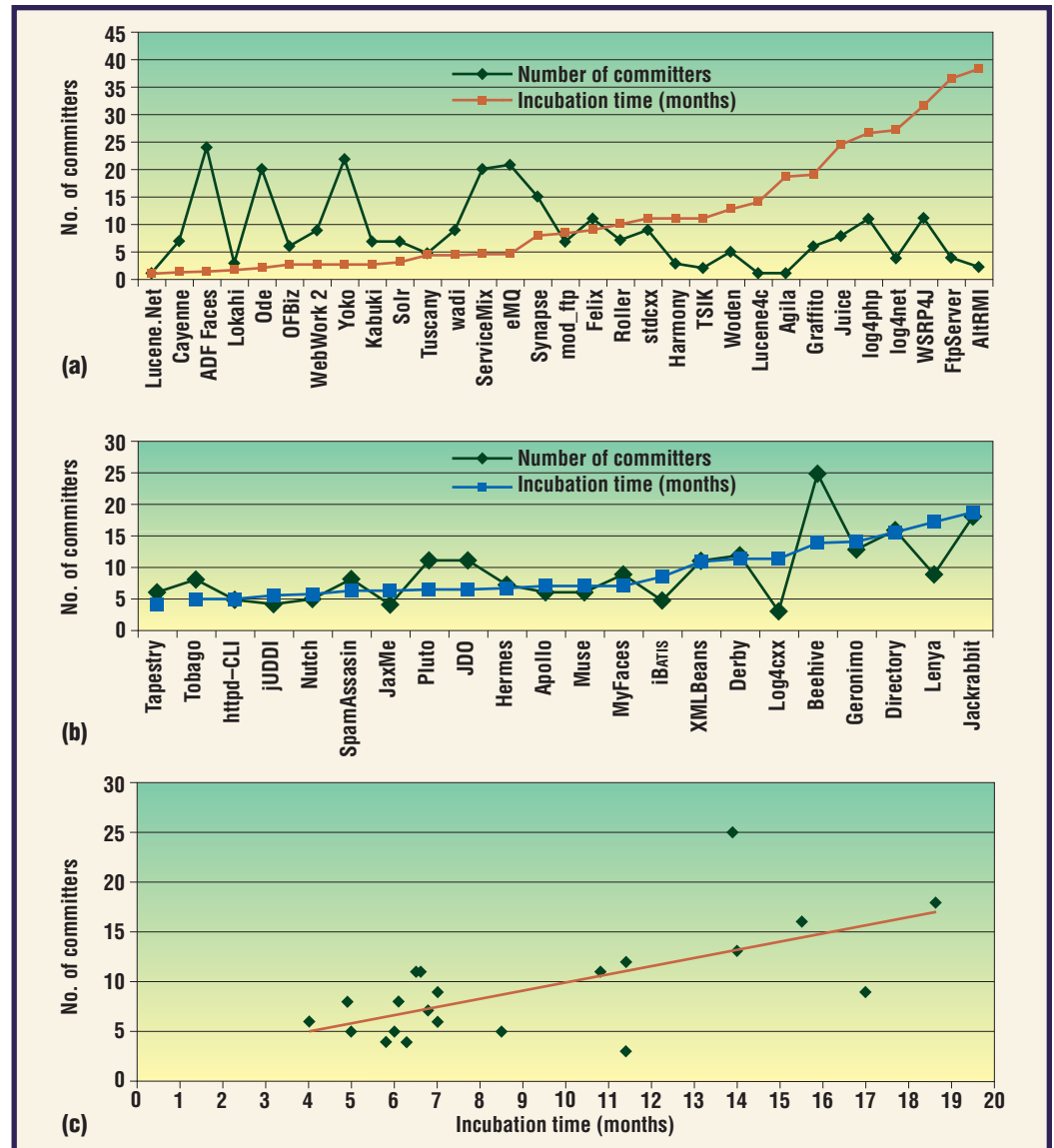
**Projects remaining in incubation.** The left-hand side of figure 2a shows that after five months in incubation, 11 out of 18 projects (with more than five committers per project) achieved a consolidated community. This period corresponds to the Incubator's fourth year and therefore to a major level of experience among the mentors and sponsor who were responsible for selecting and supporting candidate projects. On the other hand, of the projects in incubation longer than 11 months, seven had fewer than five committers and eight had more than 18 months of incubation. This hints at the potential risk of getting inactive. The April 2006 Incubator report showed evidence of some of these projects' inactivity. The right-hand side of figure 2a highlights a diminishing trend in the number of committers when the incubation time exceeds 14 months.

**Graduated projects.** Figure 2b depicts some interesting issues. For the 12 projects with shorter incubation periods, the average incubation time was six months. Except for the SpamAssassin project, all of them graduated as subprojects within other top-level ASF projects. The experience and knowledge of their parent projects (their Apache sponsors) helped to shorten their incubation time. Another advantage for these subprojects was the relationship they could establish with the parent projects' developers, some of whom got involved in the process, bringing their know-how and contributing to the critical mass of developers that keeps the community alive.

The remaining 10 projects had a longer incubation period—12.8 months on average. While two of them (Derby, Log4cxx) graduated as subprojects, the rest graduated as top-level ones, five of them having the Incubator as spon-



**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.**



sor. This suggests that top-level projects require a longer incubation time because they don't have strong relationships with existing projects within the ASF. Moreover, projects with a larger scope require a larger effort and more committers. In the Log4cxx project's case, the low number of committers (only three) might have affected the incubation period.

Figure 2c shows a linear regression curve relating the number of committers with the incubation time for graduated projects (degrees of freedom = 20,  $r = 0.661119977$ ,  $p = 0.001$ ). The correlation between the number of committers and the incubation time is strongly significant, and there's a clear indication that for larger projects (those graduated as top-level ones), obtaining the critical mass requires more time.

We believe incubation time is a period of major risk for projects: they need to achieve a stable release with a sufficient technical impact while engaging a growing community. Regarding this critical time period, the Apache incubator doesn't provide estimates for either the optimal period of incubation or the phase duration. Estimating or defining reasonable incubation duration is hard because the community that supports these projects consists of volunteer developers. However, on the basis of observed data, we consider six months, plus or minus one month, a reasonable estimation of incubation time for subprojects. For top-level projects, a feasible incubation time might be 13 months, plus or minus one month. However, activity levels during incubation are



as important as the duration itself. In this sense, long inactivity periods not only increase the incubation time but can also indicate a higher risk of project failure.

In summary, the most critical variables that impact incubation time and increase project risks are

- the period of project inactivity,
- the engagement of new committers to achieve critical mass, and
- the delivery of an early release with enough technical impact.

## Eclipse

We analyzed 23 projects from the Eclipse Technology project (the incubator). The concept of graduation doesn't exist in Eclipse; projects are assessed in the checkpoint review to decide if they pass to the following phase.

Figure 3a shows the incubation time and the number of committers registered in each project's proposal presentation (creation review). We considered that the incubation period started with the creation review; we didn't include the proposal phase because the preproposal date wasn't registered. Because projects must have at least three committers to be approved, a well-conformed community often already exists around a project: in our study, 13 out of 23 projects had more than five committers, three projects had five committers, and only five projects had fewer than five committers. Figure 3b shows a linear regression curve relating the number of committers with the time under incubation/validation (degrees of freedom = 21,  $r = 0.410672402$ ,  $p = 0.10$ ). The correlation between these factors is significant but not strong. However, these data are enough to indicate that the incubation/validation phase encourages community growth. In addition, this enforces the idea of an initial community size being an important criterion for project establishment because it has a positive impact in the early delivery of a first stable release.

## Qualitative analysis

From the information obtained about the incubation processes, we selected the practices that showed higher impact on community creation, growth, and strengthening (see table 1), especially those aimed at risk mitigation. Note the following facts:

- Both approaches differentiate between top-level projects and subprojects. This is quite relevant, because it acknowledges that many F/OSS initiatives are born inside an already formed community. Moreover, successful projects often promote subprojects that might eventually acquire the status of top-level projects.
- Both incubators require candidates to have a focused set of objectives that must be formally stated in a proposal. This helps create healthy communities that aim at solving specific problems, promoting effectiveness.
- Both emphasize the importance of the community for a project's health by using the number of committers as a criterion for approval. This results in long-living projects that continue to create value even if the initial set of committers leaves.
- Both follow an iterative approach during the incubation phase, reducing the associated risk to delivering a stable first release.

Despite their similarities, Apache and Eclipse address the incubation process in slightly different ways: Apache has created a specific organizational structure—the Incubator—so its PMC is dedicated to guaranteeing the candidate projects' success (<http://incubator.apache.org/official/resolution.html>). Eclipse prefers to incubate new projects under the structure of an existing top-level project. The top-level PMC must make a significant investment in time and energy.

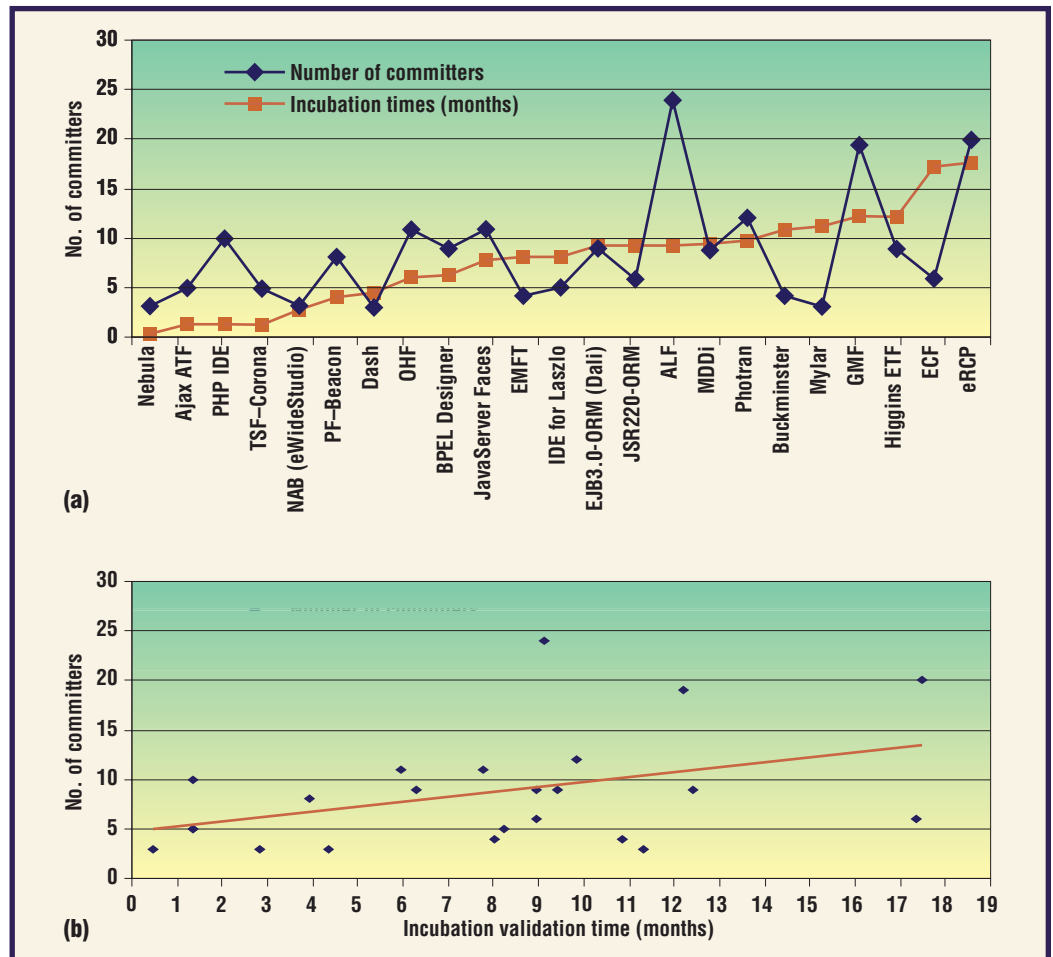
## Our proposal

Both communities could improve some of these processes. Although the Apache incubator specification defines the concept of project inactivity, which the incubator PMC uses, no clear criteria exist for declaring a project inactive. In the Eclipse community, methods for measuring project activities aren't clear, so the PMC has sole responsibility for evaluating periodic reviews. In this sense, although the EMO acknowledges that the project community's size is a key aspect for community health and the project's technical impact, the minimum number of committers to pass the creation review isn't clearly established. Not specifying this number has the advantage of avoiding a rigid decision-making process, but some guidelines would help new projects identify their weak points.

In both cases, the process presents two



**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.**



main states: the assessment proposal and delivery of the first system release. To enable the adoption of the incubation process in other communities and organizations, we propose a high-level process that takes into account the similarities of the Apache and Eclipse incubation processes. It consists of two main stages:

- the launch stage (preproposal and proposal phases), which addresses the definition of the project scope, an outline of a high-level architecture, specification of the project's main features, identification of other related projects, and engagement of the initial committers; and
- the establishment stage, in which the committers iteratively add new functionality to achieve a stable release as soon as possible. On the basis of parameters such as the number of committers, reports of activity on the project-supporting tools, and the amount and quality of public information, the project mentor and the PMC should

perform periodic checks on the community's technical advancement and growth.

**S**ome practices applied in F/OSS communities can be an excellent example of how to coordinate loosely coupled teams that are distributed across the globe.

Not all the results of our analysis on the incubators apply to other contexts. For example, the average number of committers and the average time of incubation of new projects are specific values for the communities under study. Other F/OSS communities handling a smaller code base, dealing with a smaller set of potential committers, or targeting a specific technical domain might show significant differences for these values—perhaps even failing to conform to our initial hypothesis (the community's growth correlates with technical success). But even in these other communities, applying incubation mechanisms might help select and launch new projects successfully.

**Table 1****Similarities in Apache and Eclipse incubation processes**

	<b>The Apache Incubator</b>	<b>The Eclipse incubation/validation phase</b>
Incubator process definition	<ul style="list-style-type: none"> <li>■ The process was formally established in October 2002.</li> <li>■ It's formally described, with specific stages and activities.</li> <li>■ The statement for incubation policy, principles, and philosophy are public (<a href="http://incubator.apache.org/incubation/Incubation_Policy.html">http://incubator.apache.org/incubation/Incubation_Policy.html</a>).</li> </ul>	<ul style="list-style-type: none"> <li>■ The process was formally established in November 2003.</li> <li>■ It's formally defined in a legal document at <a href="http://www.eclipse.org/org/documents">www.eclipse.org/org/documents</a> and at the Eclipse Web site.</li> <li>■ The statement of principles is public (<a href="http://www.eclipse.org/projects/dev_process">www.eclipse.org/projects/dev_process</a>).</li> </ul>
Organizational structure	<ul style="list-style-type: none"> <li>■ Roles and responsibilities are clearly defined.</li> <li>■ The structure is centralized: the Project Management Committee is responsible for administering the Incubator.</li> </ul>	<ul style="list-style-type: none"> <li>■ Roles and responsibilities are clearly defined.</li> <li>■ The structure is centralized: the Technology top-level project PMC is responsible for incubated projects.</li> </ul>
Risk mitigation measures	<ul style="list-style-type: none"> <li>■ To graduate from the Incubator, projects require at least four committers.</li> <li>■ Projects can graduate as subprojects or top-level projects.</li> <li>■ Projects follow an iterative approach during the review phase to achieve a first stable release.</li> </ul>	<ul style="list-style-type: none"> <li>■ Early in the process during creation review, the Eclipse Management Organization assesses the community's size. In the proposal phase, the PMC defines ways to participate as a top-level project or subproject.</li> <li>■ Projects follow an iterative approach during incubation/validation to achieve a first stable release.</li> </ul>
Process information made public	<ul style="list-style-type: none"> <li>■ Project proposals and board reports are published on the Incubator wiki (<a href="http://wiki.apache.org/incubator">http://wiki.apache.org/incubator</a>).</li> <li>■ Intellectual-property clearance is at <a href="http://incubator.apache.org/ip-clearance/index.html">http://incubator.apache.org/ip-clearance/index.html</a>.</li> <li>■ Status information of projects under incubation, graduated from incubation, and retired from incubation are at <a href="http://incubator.apache.org/projects/index.html">http://incubator.apache.org/projects/index.html</a>.</li> </ul>	<ul style="list-style-type: none"> <li>■ The proposal and creation review are archived and are accessible from the Eclipse proposal Web site (<a href="http://www.eclipse.org/proposals">www.eclipse.org/proposals</a>).</li> <li>■ Checkpoint reviews are archived and accessible (<a href="http://www.eclipse.org/projects/previous-release-reviews.php">www.eclipse.org/projects/previous-release-reviews.php</a>).</li> <li>■ Reports of periodical project reviews and minutes are published (<a href="http://www.eclipse.org/technology/pmc-minutes.php">www.eclipse.org/technology/pmc-minutes.php</a>).</li> </ul>
Infrastructure to support project development activities	<ul style="list-style-type: none"> <li>■ Incubator Web site and wikis</li> <li>■ Code repositories</li> <li>■ Download sites and distribution-mirroring system</li> <li>■ Mail management environment</li> <li>■ Issue/bug tracking</li> </ul>	<ul style="list-style-type: none"> <li>■ Project Web site</li> <li>■ Code repositories</li> <li>■ Download sites and distribution-mirroring system</li> <li>■ Mail management environment</li> <li>■ Issue/bug tracking</li> </ul>

In terms of software practitioners' work, we've concluded that launching technically successful, long-lived projects requires focused effort performed by a large enough set of aligned committers in a short time. Joining a stable community lets users share code bases, infrastructures, tools, and processes, and it's also a good place to find other developers interested in the same topics. If a developer is presenting a new project proposal, it should be done through the incubator.

But if joining isn't an option, the project's initial community should set as a high priority getting a critical mass following, for example, the two-phase process we've proposed. In this case, launching the project ideas and code quickly is important, as is promoting the project in public forums to gain attention and committers.

In the industrial arena, on the other hand, the decision process follows different channels. Typically, many parts of the organization are involved in the decision, and it usually enforces formal risk management policies. In some sense, industrial organizations already have their own processes for incubation.

However, in certain situations, some of the identified F/OSS incubation practices might contribute to producing better software elements while reducing the associated risks. These might include

- defining project scope early,
- launching projects after gathering a certain number of stable committers,
- exploring similar and related projects before delivering the first release, and
- classifying projects on the basis of estimated risk (depending on the differences in scope with respect to previous ones).

Moreover, adaptations of the incubation process can be used by companies to promote bottom-up innovation inside the corporation. Incubation might provide a framework in which individuals can contribute their ideas and gather help and feedback in an informal way, while at the same time providing a means for selecting which ideas have enough potential to be further developed.

The open source communities are in con-



## About the Authors



**Juan C. Dueñas** is a professor in the Telecommunications School at Universidad Politécnica de Madrid. His research interests are in Internet services, service-oriented architectures, software architecture, and software engineering and evolution. He is a committer of the Apache Felix project, leader of the OS4OS Spanish community, and promoter of the OSGi Spanish user group. He's also deputy director of the Dept. of Telematics Engineering at UPM and a member of the IEEE. He received his PhD in telecommunications engineering from UPM. Contact him at Escuela Técnica Superior de Ingenieros de Telecomunicación, Univ. Politécnica de Madrid, Ciudad Universitaria, E-28040, Madrid, Spain; [jcdueñas@dit.upm.es](mailto:jcdueñas@dit.upm.es); [www.dit.upm.es/jcdueñas](http://www.dit.upm.es/jcdueñas).

**Hugo A. Parada G.** is a PhD candidate in the telematics engineering program at UPM. His research interests include software evolution, distributed software development, and open source software development. He received his engineering degree in systems from Francisco de Paula Santander University in Colombia. Contact him at Escuela Técnica Superior de Ingenieros de Telecomunicación, Univ. Politécnica de Madrid, Ciudad Universitaria, E-28040, Madrid, Spain; [hparada@dit.upm.es](mailto:hparada@dit.upm.es); [www.dit.upm.es/hparada](http://www.dit.upm.es/hparada).




**Félix Cuadrado** is a PhD candidate and researcher in the telematics engineering program at UPM. He's contributed to several European Eureka-ITEA projects and several Eclipse projects. His research interests include open source software development and distributed services. He received his master of engineering degree in telecommunication from UPM. Contact him at Escuela Técnica Superior de Ingenieros de Telecomunicación, Univ. Politécnica de Madrid, Ciudad Universitaria, E-28040, Madrid, Spain; [fcuadrado@dit.upm.es](mailto:fcuadrado@dit.upm.es); [www.dit.upm.es/fcuadrado](http://www.dit.upm.es/fcuadrado).

**Manuel Santillán** is a PhD candidate and researcher in the telematics engineering program at UPM, a consultant for the Everis Consulting Company, and a committer in the Apache Felix project. His research interests include open source software engineering and services middleware. He received his master of engineering degree in telecommunication from UPM. Contact him at Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria, E-28040, Madrid, Spain; [santillan@dit.upm.es](mailto:santillan@dit.upm.es).



**Jose L. Ruiz** recently received his PhD in telecommunications engineering at UPM. His research interests include open source software engineering and services middleware. He's a contributor to several European Eureka-ITEA projects and a committer in the Apache Felix project, a leader of the OS4OS community, and a promoter of the OSGi Spanish user group. Contact him at Escuela Técnica Superior de Ingenieros de Telecomunicación, Univ. Politécnica de Madrid, Ciudad Universitaria, E-28040, Madrid, Spain; [jlrui@dit.upm.es](mailto:jlrui@dit.upm.es); [www.dit.upm.es/jlrui](http://www.dit.upm.es/jlrui).

stant evolution: at the time of this writing, the Eclipse community is reviewing its incubation process to make it more agile, Apache is introducing new initiatives trying to foster innovation, and incubation is mandatory in both communities.

In the future, researchers need to track graduated projects after their first release to analyze aspects related to evolution, effort, community evolution, and evolution of the incubation process itself. 

## Acknowledgments

We thank the anonymous reviewers for their insightful comments. Also, thanks to Jesús Bermejo

from Telvent Interactiva for his support in this work, in the context of the Eureka ITEA COSI project, under a grant from the Spanish Ministerio de Turismo Industria y Comercio-PROFIT.

## References

1. J. Lonchamp, "Open Source Software Development Process Modeling," *Software Process Modeling*, S.T. Acuña and N. Juristo, eds., Series 10, Springer, 2005, pp. 29–64.
2. J.O. Gilliam, "Improving the Open Source Software Model with UML Case Tools," *Linux Gazette*, vol. 67, June 2001.
3. W. Scacchi, "Socio-Technical Interaction Networks in Free/Open Source Software Development Processes," *Software Process Modeling*, S.T. Acuña and N. Juristo, eds., Springer, 2005, pp. 1–27.
4. S. Koch, "Evolution of Open Source Software Systems—A Large-Scale Investigation," *Proc. 1st Int'l Conf. Open Source Systems*, 2005, pp. 148–153; <http://oss2005.case.unibz.it/Resources/Proceedings/OSS2005Proceedings.pdf>.
5. A. Capiluppi, P. Lago, and M. Morisio, "Characteristics of Open Source Projects," *7th European Conf. Software Maintenance and Reengineering*, IEEE CS Press, 2003, pp. 317–327.
6. D. Weiss, "Quantitative Analysis of Open Source Projects on SourceForge," *Proc. 1st Int'l Conf. Open Source Systems*, 2005, pp. 140–147; <http://oss2005.case.unibz.it/Resources/Proceedings/OSS2005Proceedings.pdf>.
7. Y. Ye and K. Kishida, "Towards an Understanding of the Motivation of Open Source Software Developers," *Proc. 25th Int'l Conf. Software Eng.*, IEEE CS Press, 2003, pp. 419–429.
8. W. Scacchi, "Understanding Open Source Software Evolution," *Software Evolution and Feedback: Theory and Practice*, N.H. Madhavji, J. Fernandez-Ramil, and D. Perry, eds., John Wiley & Sons, 2006, pp. 181–206.
9. C. Jensen and W. Scacchi, "Modeling Recruitment and Role Migration Processes in OSSD Projects," *Proc. 5th Workshop Software Process Simulation and Modeling (ICSE-ProSim)*, 2005; [www.ics.uci.edu/~7Ewscacchi/Papers/New/Jensen-Scacchi-ProSim05.pdf](http://www.ics.uci.edu/~7Ewscacchi/Papers/New/Jensen-Scacchi-ProSim05.pdf).
10. Y. Liu, E. Stroulia, and H. Erdogmus, "Understanding the Open-Source Software Development Process: A Case Study with CVSChecker," *Proc. 1st Int'l Conf. Open Source Systems*, 2005, pp. 154–161; <http://oss2005.case.unibz.it/Resources/Proceedings/OSS2005Proceedings.pdf>.
11. C. Gacek and B. Arief, "The Many Meanings of Open Source," *IEEE Software*, vol. 21, no. 1, 2004, pp. 34–40.
12. B. Fitzgerald, "The Transformation of Open Source Software," *MIS Quarterly*, vol. 30, no. 3, 2006, pp. 587–598.

For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).