COMPANY-LED OPEN SOURCE COLLECTIVES

MICHAEL WEISS

1. Introduction

Open source software (OSS) has become an integral part of commercial software. There has been significant amount of research on how companies engage with open source projects [12, 3, 4, 9, 1]. However, previous research has primarily focused on how single companies can benefit from initiating a new open source project or contributing to an existing project. In this article, our focus is on networks of companies (or collectives) within the same application domain to develop shared assets in the form of open source projects.

Companies use open source to reduce their development and maintenance cost, and to improve their time to market [13]. Building on open source allows them to focus their development effort on the points of difference over their competitors. For most products, only a small portion (less than 20%) of the software differentiates a company. The non-differentiating portion of the software can be obtained from external sources, either as commercial off-the-shelf components or as open source software [10].

Another line of research relevant to our study is product line engineering. Companies use product lines to manage product diversity and reuse [10]. Product line engineering separates the development of a common platform from the development of applications. Platforms identify points of commonality and variability. Applications are created by selecting the appropriate variants [8]. However, most research on product line engineering focuses, once again, on how single companies can structure their product development process as a product line, or how they can offer a platform to third parties.

Recently, research on product line engineering has started to examine the relationship between OSS development and product line management [2, 10, 11]. The research differentiates between using OSS in a product line, and the adoption of product line practices in OSS. Many large open source projects are, de facto, structured as product lines, that is, there are clearly identifiable platform and application components [11].

The objective of our research is to examine how collectives of small companies lever open source platforms to improve their ability to deliver products. A collective of companies shares the cost and risk of developing a shared code base, but also reams the benefits that come from being able to reuse code. A good example of collective software development is the Eclipse open source project. Over 80% of contributions to the Eclipse platform are company contributions made by a group of more than 50 companies [5].

2. Evolution of OSS projects

Many open source projects start out with a single developer or company with a need. The need is narrowly defined and focuses on resolving an immediate technical challenge faced by the project initiator. An example of a project started by an individual is Linux; the project started out as a personal project by Linus Thorvalds to build a freely available Unix operating system. An example of a company-initiated project is Eclipse; the project started with IBM donating the codebase for its VisualAge product as open source.

At this point, the project initiator is in full charge of the direction of the open source project. The next stage of evolution occurs when a community forms around the project. Typically, the project initiator is still in charge of the technical roadmap of the project, and the community members (individuals or other companies) create products or services complementary in nature to the project. Growth of the open source project is limited beyond this point, unless it moves from a model where a single entity controls the direction of the project to a model where all community members collectively decide on its course.

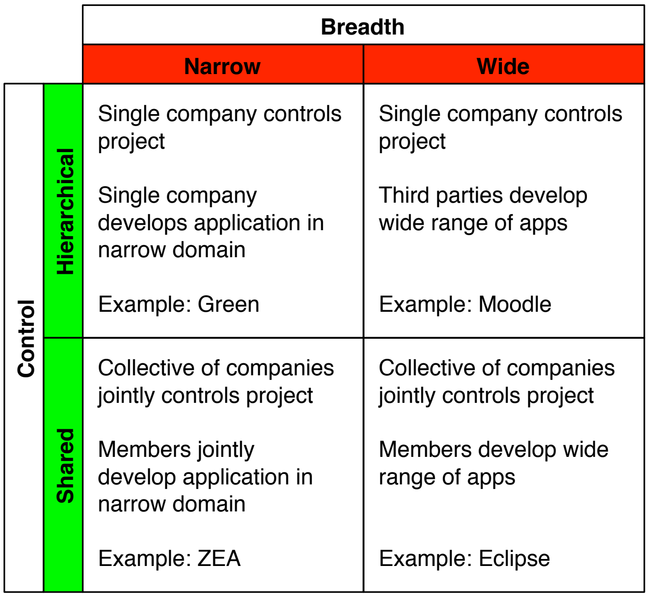
Evolution of the project to this model requires that the project initiator is removed at arms length from the project (as documented by [15] for a range of open source projects), and joins the community as just another member. The direction of the project is now set by the member organizations. Often, the relationship between the members and the project is also formalized through a neutral organization or foundation, which acts as the legal representative of the project and facilitates between the community members [2]. For example, the Eclipse project is coordinated by the Eclipse Foundation.

Take, for example, project Green. Green is a project in the education space that was started at a university by a single developer and was then spun out into a company. The project initially had a small group of core contributors, and control of the direction of the project was with the spin-off company. A small community has formed around the project consisting of companies and individuals that develop custom features and offer complementary services to the project. But, at this point, a significant shift occurs.

More companies want to join the community, however, they do not feel that their needs are met under the current project structure. These companies differentiate themselves from each other through their specific application domains, not in terms of the platform they share. This changes the nature of the project, and to reflect this change, a foundation is created to manage the project and the project is renamed into Red. In the Red project, the other companies take a more active role. The new project is ready to grow in size and diversity in ways that the Green project could never have done.

3. How companies participate

Company-led open source projects differ in significant ways in terms of who controls the project, and the breadth of applications derived from the project [14]. Control refers to decision making, and includes control over the project direction, its architecture, commits and releases, and who captures the majority of the value. Control can be hierarchical or shared. In a hierarchically controlled project, a single company makes all the decisions. In a project with shared control, decisions are made jointly by the project members.



		Breadth	
		Narrow	Wide
Control	Hierarchical	Single company controls project Single company develops application in narrow domain Example: Green	Single company controls project Third parties develop wide range of apps Example: Moodle
	Shared	Collective of companies jointly controls project Members jointly develop application in narrow domain Example: ZEA	Collective of companies jointly controls project Members develop wide range of apps Example: Eclipse

Figure 1. How companies participate in a company-led open source project

Applications can be either in a narrow domain (such as education), or spread across a variety of domains (such as language training and business intelligence). If applications are in a narrow domain and the project is controlled by a single company, the project often has an integral architecture. The reason is that the company has little incentives to divide the architecture into modules, as it requires additional effort. However, when other companies are involved in the project, the architecture needs to be modular to some degree.

There are four basic ways for companies to participate in a company-led open source project [14], as shown in Figure 1. This categorization is based on first-hand experience with open source start-ups and an examination of extensible open source platforms conducted by the author in [7]. As should be apparent from the earlier discussion, the Green project belongs into the top-left quadrant. In the top-right quadrant, a single company exposes an interface to attract third-parties to create applications, for example, the Moodle learning management system. As an example of a company in the bottom-left quadrant, the Zope Europe Association (ZEA) coordinates a group of open source companies, allowing them to compete for large government contracts [6]. The bottom-right quadrant is reserved for collectives of companies that jointly control a platform, which provides the basis for a diverse range of applications. The Eclipse project is an example of such a collective.

4. Discussion

This paper develops a model of how companies can participate in company-led open source projects. The differences between the four quadrants of the model can be interpreted in terms of the product line concepts of commonality (platforms) and variability (applications). Hierarchical-wide OSS projects and OSS projects with shared control are organized like product lines: a platform and applications that extend it.

Hierarchical-wide and shared-wide open source projects like Moodle and Eclipse have a plug-in architecture that provides variability through extension points and extensions. As observed in [2], the products in this product line are new plug-ins and products using existing plug-ins. Both Moodle and Eclipse support a wide breadth of applications. However, the amount of variation supported by Eclipse is much higher than for Moodle.

The key difference between hierarchical-wide and shared-wide projects is that in the former, a single company is in charge of the project (the traditional model of product development), whereas in the latter a collective of member companies jointly decides on the course of the project and shares the cost and risk of implementation.

Collectives also allow small companies to to compete for much larger contracts than they could do individually by providing their members with a common brand, pooling their assets, and creating a reliable delivery process. Examples of variation are localization and geographic coverage: member companies of ZEA are distributed across Europe.

The collective model allows small companies to develop reusable assets and focus on the differentiating aspects of their products. It also allows them to deliver whole products instead of pieces of a solution, thus making them more competitive. This new organizational form was precipitated by the demise of large companies, and the need of the new small companies that have arisen from the ashes to deliver products more effectively. By necessity, these companies had to adopt a more open approach to product development.

References

[1] Capra, E., Francalanci, C., Merlo, F., and Rossi-Lamastra, C. (2011), Firms' involvement in open source projects: A trade-off between software structural quality and popularity, The Journal of Systems and Software, 84, 144-161.

[2] Chastek, G., McGregor, J., and Northrop, L. (2007), Observations from viewing Eclipse as a product line, Workshop on Open Source Software and Product Lines, colocated with the International Software Product Line Conference (SPLC).

[3] Dahlander, L. (2007), Penguin in a new suit: a tale of how de novo entrants emerged to harness free and open source software communities, Industrial and Corporate Change, 16(5), 913-943.

[4] Dahlander, L. (2008), How do firms make use of open source communities?, Long Range Planning, 41, 629-649.

[5] Eclipse Foundation (2011), Company/project commit details, `http://dash.eclipse.org/dash/commits/web-app/commit-count-loc.php`, last accessed Feb 26, 2011.

[6] Feller, J., Finnegan, P., and Hayes, J. (2006), Open source networks: An exploration of business model and agility issues, European Conference on Information Systems, `http://is2.lse.ac.uk/asp/aspecis/20060179.pdf`.

[7] Noori, N., and Weiss, M. (2009), Managing the quality of platform extensions, FLOSS International Workshop on Free/Libre Open Source Software, `http://www.decon.unipd.it/personale/curri/manenti/floss/weiss.pdf`.

[8] Pohl, K., Böckle, G., and van der Linden, F. (2005), Software Product Line Engineering: Foundations, Principles, and Techniques, Springer.

[9] Stuermer, M., Sebastian S., and von Krogh, G. (2009), Extending private-collective innovation: A case study, R&D Management, 39(2), 170-191.

[10] van der Linden, F. (2009), Applying open source softeware principles in product lines, Upgrade, X(2), April, 32-40.

[11] van Gurp, J., OSS product family engineering (2006), International Workshop on Open Source Software and Product Lines, colocated with the International Software Product Line Conference (SPLC).

[12] von Hippel, E., and von Krogh, G. (2003), Open source software and the private-collective innovation model: issues for organization science, Organization Science, 14(2), 209-223.

[13] Weiss, M. (2010), Profiting from open source, European Conference on Pattern Languages of Programs (EuroPLoP).

[14] Weiss, M. (2011), Control and diversity in company-led open source projects, Open Source Business Resource (OSBR), April, 29-32, `www.osbr.ca`.

[15] West, J., and Gallagher, S. (2006), Challenges of open innovation: The paradox of firm investment in open-source software, R&D Management, 36(3), 319-331.

Technology Innovation Management Program, Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, K1S 5B6, Canada

E-mail address: `weiss@sce.carleton.ca`