Economics of Software Product Development Collectives

Michael Weiss

> *"Define very precisely what your competitive differentiators are for your customers or you're going out of business. Focus all possible energies there, and acquire everything else from open source software, or help build it in open source software. Or in other words: pick your niche; co-evolve the platform in collaboration with other actors in the ecosystem."*
>
> Mike Milinkovich
> Executive Director, Eclipse

Where software product development occurs is shifting from single companies to groups or collectives of companies. In this article, we retrace the evolution of how software product development is organized and then offer insights into the economic motivation for collectives, which will be relevant to companies considering joining a software product development collective. Building on the literature on software product line economics, we identify three factors affecting the economics of collectives (level of contribution, number of members, and diversity of use), and develop a model that links those factors to three economic outcomes (time, quality, and cost). This model can be used by potential members when deciding whether or not to join a collective.

Introduction

The traditional view of software development is that it occurs within a single company. While parts of the development may be sourced from outside the company, the final product has been specified, and is owned in full, by the company. When a company develops multiple products in the same domain, it benefits from organizing its software development activity as a product line (http://sei.cmu.edu/productlines).

A product-line provides a platform (also known as a core asset base) shared by a set of related products that are developed by an organization. The shared platform identifies points of commonality and variation. Products are created on top of the platform by reusing its core assets, while reducing the effort that goes towards developing assets that are unique to the product.

The motivation for a product line is reducing the cost of developing new products while increasing their quality and reducing the time to market. By taking a product line approach, a company can manage product diversity and reuse more systematically. In other words, products built using a product line approach will share a common base, which allows a company to manage customer-specific variations more systematically.

This traditional view is being challenged by two recent developments: a transition from software product lines to software ecosystems (Bosch, 2009; http://tinyurl.com/3gfr5lg) and a transition from software ecosystems to collectives. A transition from product lines to software ecosystems takes place when a product-line company makes its platform available to developers outside the company. These include internal developers (as in a product line), strategic partners with long-term relationships, undirected external developers, and independent solution providers.

The transition from software ecosystems to collectives recently has created many new collectives, even though they often go by different names, including "ecosystems". Examples are the open source Eclipse project (http://eclipse.org) and the closed source Artop ecosystem (http://www.artop.org). A collective is set up when

Economics of Software Product Development Collectives

Michael Weiss

a group of organizations wants to achieve a goal they cannot achieve on their own. A collective can address common needs of its members, allowing them to focus on the differentiating features of their products.

It is often observed that somewhere between 50% and 90% of development effort is spent on creating software that does not differentiate a company from its competitors (van der Linden, 2009: http://tinyurl.com/6ef7p22; Milinkovich, 2008: http://tinyurl.com/6aguklw). Only the remainder differentiates a company from its competitors. This observation has motivated companies to acquire the non-differentiating parts of their software stack elsewhere, for example, as COTS (http://wikipedia.org/wiki/Commercial_off-the-shelf) or open source software. When such software is not available, or when a higher degree of control over the software is desired to enable more effective customization, organizations have joined efforts to create their own common software stack in a collaborative effort, making the result available to each other, or even to anyone else who wishes to use it.

This article seeks to identify the factors that affect the economics of collectives and to create a model linking those factors to economic outcomes. It develops propositions from case studies of collectives about how the composition of a collective affects the achievement of the business goals of their members. The propositions link three characteristics of collectives (level of contribution, number of members, and diversity of use) to three variables used to model the economics of product lines (time, quality, and cost).

Collectives

A collective can achieve things that its individual members cannot achieve on their own, as described in the April 2011 issue (http://timreview.ca/issue/2011/april) of this publication. For example, as a collective, a group of startups can deliver a complete solution to a customer, whereas individually they are only able to deliver pieces of the solution, which the customer has to integrate. Joining forces makes the group of startups much more competitive against large system integrators. Collectives can also collaborate to address common needs, allowing their members to focus on the differentiating features of their products. The more members a collective has, the more its members are able to share the load of meeting common needs. However, such collaboration is also fraught with problems, for example, the coordination overhead that results from dependencies between subtasks.

A key characteristic of collectives is that they are voluntary organizations. Membership in a collective is a function of how well the collective helps its members meet their business goals.

As contributors to the collective, members gain access to the total value generated by the collective. Previous research has shown that, as long as the total value received is higher than the cost of contribution, members benefit from joining (Baldwin and Clark, 2006; http://tinyurl.com/3qygf9y). Conversely, existing members of a collective are not interested in members who do not add value to the collective. Thus, collectives often impose conditions on membership such as asking members to commit resources.

Figure 1 summarizes the transitions from a single company to a collective model of developing software products. The transitions occur along two dimensions. The first transition is from an internal to an external activity, as the platform is made available to external developers. The second transition is from a hierarchical to a network type of governance. The locus of creation and evolution of the platform shifts from a single platform owner to a network that collectively creates and owns the platform.

Case Study: Eclipse

In the research underlying this article, we studied several cases both from firsthand observation and the literature. From these cases, we identified factors that affect the economics of collectives and created a model that links those factors to economic outcomes. The model is described as a set of propositions or statements that suggest causal links between the factors and the economic outcomes. A summary of each case was prepared that described its purpose, governance structure, and software architecture. Factors and economic outcomes were identified in an iterative manner.

In this section, we describe one of our case studies in detail: the Eclipse project. Eclipse is an open source community focused on building an open software development platform (Smith and Milinkovich, 2007; http://timreview.ca/article/94). The Eclipse project was founded in 2001 as a spin-out of technology that IBM had acquired from Object Technology International. Initially, the Eclipse community was primarily driven by IBM and other software vendors. In 2004, with the creation of an independent, non-profit governance body – the Eclipse Foundation – IBM relinquished its control over the project and allowed other players, in-

Economics of Software Product Development Collectives

Michael Weiss

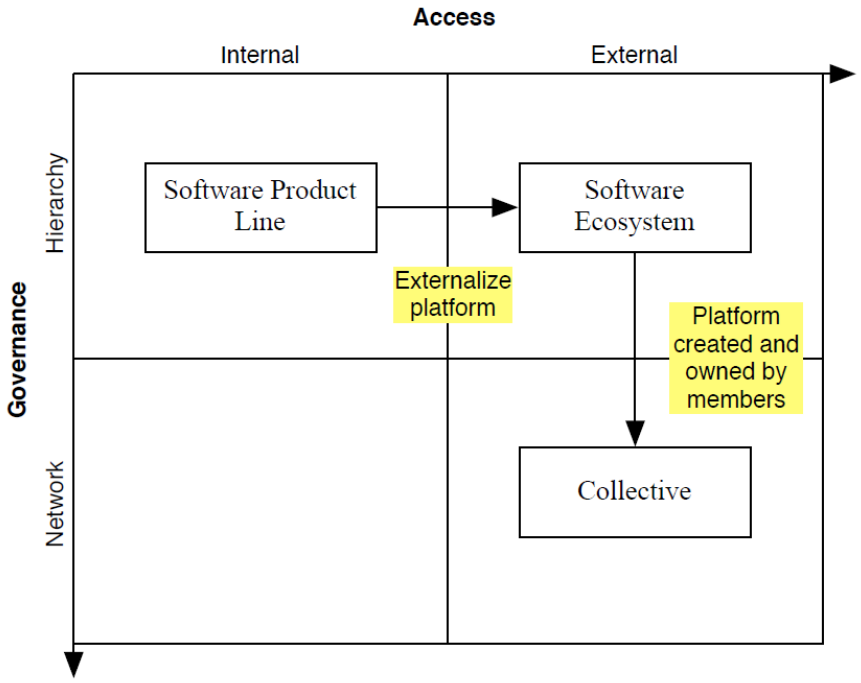

Figure 1. Evolution of software product development models

cluding IBM's competitors, to become equal members of the community.

The Eclipse Foundation is responsible for the technical infrastructure, coordinating the development process, handling the intellectual property rights, and promoting Eclipse and its wider ecosystem. The role of the Eclipse Foundation is administrative; it does not set the direction of the project or develop code. The direction of the project is set by strategic members of the collective. To become a strategic member, a company has to pay a membership fee and commit resources to the development of the platform. The Eclipse project is organized as a set of top-level projects with subprojects.

Eclipse has a well-defined process for member engagement, and project guidance is provided by three councils. The requirements council collects, reviews, and prioritizes incoming requirements. The planning council manages the release train. The architecture council defines and evolves the architecture of the Eclipse platform. Individual projects are overseen by project management committees. The councils are composed of strategic members and representatives of the project management committees.

Eclipse is designed to be highly extensible. At its core is a minimal runtime that provides tools for extension management. All functionality of Eclipse (even "core" functionality such as basic user interface elements) is implemented in the form of plug-ins. Plug-ins are the basic distribution unit of functionality in Eclipse. A plug-in can declare extension points, which are points where the behavior of the plug-in can be extended by others. It also implements extensions to the extension points of existing plug-ins. Those extension points are not predefined by the Eclipse platform, but can be defined by each plug-in author.

Findings

From the analysis of the cases examined in this research, three factors were identified as characteristics of collectives: level of contribution, number of members, and diversity of use. Level of contribution refers to the amount of work contributed to the core asset base by a member of the collective. Contributions are not limited to code, but can include requirements, designs, test cases, and feedback. The number of members is the size of the collective. Diversity of use measures the range and variety of contexts of use for the platform.

Economics of Software Product Development Collectives

Michael Weiss

Figure 2 shows a model that links these factors to economic outcomes that we developed as a result of examining the case studies. Traditional cost-benefit models of product lines only model the impact on cost, not other benefits such as time to market or quality. The three economic outcomes considered in this model are time, quality, and cost. Time is either time to market or the coordination overhead. Quality refers to the quality of the core asset base or the quality of the product. Cost is either the cost for the organization of the collective, the cost to create the core asset base, the cost to reuse assets, or the cost to create a unique asset not based on the platform.

The level of contribution is not evenly distributed among members of a collective. Instead, as studies of open source projects show, a small number of members account for a majority of the contributions (Crowston et al., 2011; http://tinyurl.com/3nrntty). Some members may be in a better position to create a specific core asset, because the skills required are not generally available, or they may have a more urgent need than other members for a specific asset to be available in the asset base. Most Eclipse subprojects receive their primary input from a single company. This company has greater influence over which core assets a contained in the platform than companies that contribute less.

Proposition 1: *Time to market decreases with the level of contribution as a result of better alignment between contributed assets and the contributor's needs.*

In the literature on small groups, trust has been noted as a determinant of effective team collaboration (Crowston et al., 2011; http://tinyurl.com/3nrntty). Successful leaders make a strong contribution and hold a central position in the community. Projects run by leaders who have demonstrated their technical skills and who have a record of past successes are generally more likely to succeed. Trust can be increased by developing key functionality early in a project to demonstrate that the project is doable and has merit. With the initial release of the Eclipse source code in 2001, IBM triggered contributions from other companies.

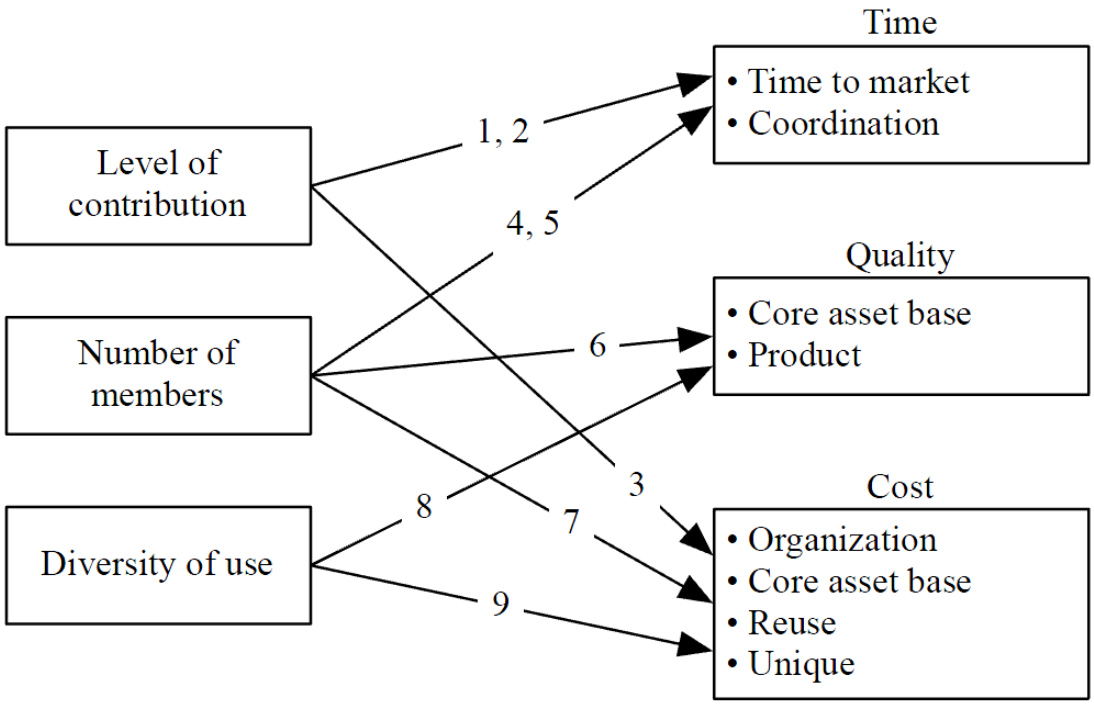

Figure 2. Linking factors to economic outputs. The arrows between factors and economic outcomes are the propositions that suggest causal relationships between them. The numbers on the arrow refer to the propositions. For example, the level of contribution influences time and cost.

Economics of Software Product Development Collectives

Michael Weiss

Proposition 2: *Coordination overhead decreases with the level of contribution as a result of the increase in trust it creates between the members.*

Through their level of contribution, a member can ensure the core assets fit with their business goals. Members who contribute the most to a specific asset can expect to benefit when reusing the asset. A study of open source development found that contributors obtain private benefits from the development of shared assets that are not available to "free riders" who only use the assets (von Hippel and von Krogh, 2003; http://tinyurl.com/6e39qa3). These include learning, sense of ownership and control, and feedback from others on the contributed code. Contributors are also in a better position to tailor their code to their individual needs, because the code that they contributed for general use may not be a good fit with someone else's needs. Many commercial products (such as IBM's WebSphere product) are built on top of the Eclipse platform. When IBM released the initial version of Eclipse, they had a significant lead over others in using the platform even though the code was open to anyone.

Proposition 3: *The cost to reuse assets in the core asset base and the cost to develop unique assets both decrease with the level of contribution.*

When members of a collective contribute to a core asset base, they develop a shared platform. The purpose of the shared platform is to provide non-differentiating functionality to members of the collective so that each member can focus on its differentiating features. A decision on whether to include a contribution in the shared platform is made on the basis of how well the contribution is aligned with the goals of the other members of the collective. If a contribution were only to benefit a single member, then it would not be included in the platform. For example, the Eclipse Modeling Framework provides modeling and code generation capabilities that are leveraged by tools such as Rational Rose. Tools based on the framework can interoperate because they share common representations.

Proposition 4: *The time to market decreases with the number of members. Members can focus on the development of value-added features.*

Each member added to a team introduces coordination overhead, which is time not spent productively towards achieving the task of the team. The capacity of team members to interact with one other in meaningful ways is also limited. Conversely, a smaller number of collaborators allows members to interact more frequently with each other. This creates stronger ties among the members and increases commitment and identification with the collective and its goals. The effort to coordinate activities can be controlled by restricting access, that is, strategically selecting members for specific interactions. In open source software projects, restricting access to core members reduces the amount of coordination required when members collaborate on a section of the project. The Eclipse project is organized into top-level projects, each of which has multiple subprojects. Only a subset of project members is active in any specific subproject.

Proposition 5: *Coordination overhead increases with the number of members working on the same section of the core asset base.*

A high level of quality in the core asset base attracts new members to the collective. Products built on top of a high quality base will also be of higher quality. In a collective of small companies, individual members do not have the resources to build a system to the level of quality provided by the platform. From proposition 2, it is also apparent that a collective needs to receive enough initial contributions in order to reach an acceptable level of quality that will attract more new members. A study of embedded systems companies using Linux showed that these companies were motivated to reveal their changes to Linux to receive technical support from other companies (Henkel, 2006; http://tinyurl.com/3dbfl7v).

Proposition 6: *The quality of the core asset base increases with the number of members who provide feedback on the assets in the core asset base.*

A collective approach to developing a core asset base is more efficient than for each member of the collective to develop a full software stack in isolation. Instead of creating their own versions of commodity features, members can focus on developing features that differentiate them from each other. The effort for maintaining the software stack as it evolves is also significantly reduced. Changes in underlying technologies can be spread among members. If members have existing investments in their own software stacks, switching to a platform developed by a collective may be expensive at first, but will pay off in the long term. Companies that build on the Eclipse Modeling Framework differentiate themselves through the value they offer to end users.

Economics of Software Product Development Collectives

Michael Weiss

Proposition 7: *The cost of contributing to the core asset base decreases with the number of members who provide resources.*

Each time the core asset base is put to use in a new context, new aspects of the base will be exercised. Each new context of use may uncover errors or omissions that had not been identified before. This increases the chance of correcting errors, thus increasing the quality of all products that depend on the asset base. For example, each Eclipse subproject exposes the shared core components to new uses.

Proposition 8: *The quality of the core asset base increases with diversity of use. Each new context of use will further harden the asset base.*

Diversity of use is driven by the diversity of needs of the members of the collective. At early stages of growth, the availability of multiple perspectives that come with diversity of use benefits a collective. Decisions about what functionality to include in the core asset base will be made from a broad understanding of product needs. At later stages, too much diversity may, in fact, hinder the evolution of the core asset base in a cohesive manner. When initially released, the Eclipse project provided core components for a Java-centric development environment. It subsequently grew in diversity to include components for tool integration, modeling, and web applications that could be applied across a range of domains. Today, Eclipse can perhaps be best characterized as a collection of vertical solutions for specific domains. About one half of the Eclipse project pool today is technology specific. The diversity of Eclipse projects has increased significantly, and as a group, the projects are far less cohesive now.

Proposition 9: *The cost of creating the core asset base first decreases, then increases with diversity of use. At low diversity of use, the collective benefits from a broader range of perspectives. When diversity of use is high, the collective will appear less cohesive.*

Conclusion

The focus of this article was on the shift in software product development from single companies to collectives. The analysis revealed motivations for companies to join a collective by examining the economics of collectives. The article also argued that development in collectives effectively amounts to the creation of a shared platform or product line. Different from traditional software products lines, which are managed by a single platform owner, these product lines are collectively owned. Another important difference is that the members of a collective are typically small and do not have extensive experience in product line engineering. In a future article we will explore the notion of a minimal viable product line, asking how a company can obtain some of the benefits of a product line approach without a full implementation of the approach.

Even though we used the open source collective Eclipse as our example in this article, we have also found the same patterns with closed source collectives (i.e., those that do not share the results of their work with nonmembers). Closed source collectives obtain the same types of benefits from collaboration as their open source cousins. Forming a collective is not a question of open or closed sourcing; it is a question of development models.

Acknowledgement
A version of this article was presented at the International Workshop on Quantitative Methods in Software Product Line Engineering (QMSPLE 2011; http://users.dsic.upv.es/workshops/qmsple2011/).

About the Author

Michael Weiss holds a faculty appointment in the Department of Systems and Computer Engineering at Carleton University, and he is a member of the Technology Innovation Management program. His research interests include open source ecosystems, mashups/Web 2.0, business process modeling, social network analysis, and product architecture and design. Michael has published on the evolution of open source communities, licensing of open services, and innovation in the mashup ecosystem.

Citation: Weiss, M. 2011. Economics of Software Product Development Collectives. *Technology Innovation Management Review.* October 2011: 13-18.