Economic Congruence in Open Source Ecologies: Aligning Incentives and Architecture

James Herbsleb, Roberto Weber, Yuanfang Cai, and Thomas Finholt *

April 03, 2008
CMU-ISR-08-109

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*Yuanfang Cai is a Professor at Drexel University, Philadelphia, PA. Thomas Finholt is a professor at University of Michigan, Ann Arbor, MI.

Abstract

This position paper suggests how one could use game theory to describe the incentive structure of open source ecologies, and shows how the structure and payoffs of the game are dependent on the architecture of the software. This incentive structure is critical to open source ecologies which, unlike many individual open source projects, are defined by significant participation by firms. We describe how firm participation in an open source ecology can be modeled as two games. One is a variety of a weak-link coordination game, in which the payoff to a firm depends on other firms also deciding to participate. The second is a public goods game, where all firms benefit from contributions to the public good, but defectors have an incentive to free ride on the efforts of others. The structure of both games is crucially dependent on the interdependence between players, which can be modeled by the software architecture, which determines, for example, the extent to which the open source architecture can support independent extension and evolution of individual components developed by different participants. The incentives for contribution will vary greatly among firms, as will the number of players who care about a particular component, and the payoff from maintaining and enhancing it. Based on prior research, all of these factors are highly likely to influence the behavior of participants, and have a

Keywords: Software development, Congruence, Economic Congruence

major impact on the success of the ecology. We conclude with several research questions raised by our analysis.

This effort is partially funded by the National Science Foundation under grant number IIS-0534775, IIS-0329090, and the Software Industry Center and its sponsors, particularly the Alfred P. Sloan Foundation. Effort also supported by a 2007 Jazz Faculty Grant. The views and conclusions are those of the authors and do not reflect the opinions of any sponsoring organizations/agencies.

1 OPEN SOURCE: FROM PROJECTS TO ECOLOGIES

Open source ecologies provide a novel, increasingly important, complex, and poorly understood mode of production for software. Individuals, firms, and foundations with differing motivations and resources collaborate in the creation of public goods while also serving their individual interests. Such ecologies are fielding very competitive products, even when matched against large, successful firms. While open source software ecologies such as Eclipse, Apache, and Gnome are the most prevalent form of this production mode, there are also well-known and important examples in other digital domains, such as Wikipedia. While we focus on software, we also strive to develop a theoretical framework that can be applied to open production more generally.

Several open source ecologies now have substantial economic importance, but their eventual long-term impact is hard to determine. Are we observing a novel production mode that will continue to occupy a small -- but important -- place among traditional development organizations, or are we watching the embryonic stages of an organizational form that will fundamentally change the industry? Research has made substantial progress in understanding how individual projects function and why individuals volunteer their labor (see next section), but ecologies are dominated by firms, and are sustained by firm-level decisions. In addition to practical concerns that drive our interest in open source ecologies, these structures present an opportunity to understand coordination and incentive structures in a novel context outside the usual bounds of hierarchies and markets [Malone et al 1993; Powell 1990].

Our preliminary work on the Gnome and Eclipse ecologies is beginning to shed some light on the benefits organizations derive from their participation. Interviews with Eclipse participants, for example, indicate that a primary benefit for firms is the opportunity to share the cost of developing commodity software. Vendors developing new features for an Integrated Development Environment (IDE) would prefer to invest engineering resources in marketable new features, not in the tedious and costly task of maintaining and enhancing basic functionality. Collaborating on the basic Eclipse framework allows them to share this cost.

Beyond this cost-sharing benefit of participation, there are others that vary depending on the company's business model. Many firms sell proprietary software (a "plug-in") that works with Eclipse, providing additional functionality, such as code analysis or visualizations. Other companies sell complementary goods whose value is enhanced by Eclipse. IBM, for example, has a major presence in the web server market, and an IDE that makes it easy to develop web applications for their platform can be expected to improve their sales. Still other participants provide installation, customization, or training services for users of Eclipse. A subset of companies cutting across these categories also use the Eclipse community and Eclipse-sponsored events as a major marketing channel for their Eclipse-related goods and services. Moreover, contributing to the free distribution may help to increase Eclipse's share of the IDE market, at the expense of proprietary products. Increasing the number of Eclipse users increases the potential market for any company that sells Eclipse-related products or services. Participation in Eclipse and contributing to the free distribution can also be a way to expose potential customers to a basic version of their product, with the hope that many customers will purchase the full proprietary version. Thus, the ecology is fuelled by complex, inter-related motives, resulting in both a public good and the creation of markets for complementary goods, proprietary enhancements, and services.

In order to function, an open source ecology must provide two different types of economic incentives. First, there must be adequate incentives for participation in the ecology. There must be ways for firms to earn adequate returns by offering products or services tied to the open source software. In the absence of such incentive, the open source project may continue to exist or even thrive, but the firm-driven economic activity of an ecology will not develop. Second, there must be adequate incentives for producing, maintaining, and enhancing the public good. As with all software, maintenance of the open source version of the software is necessary if it is to continue to be viable. It must be made to run on new operating systems, interact with new hardware, acquire new features to compete in the market place, and have the inevitable bugs fixed quickly. If inadequate resources are devoted to maintenance, the software will lose out to competitors.

Underlying both incentive problems, and giving them their particular shape within an ecology, is the modular structure of the software architecture. For example, the extent to which various components within an architecture are loosely coupled determines how easily the participants can improve each component independently or extend the system with new features without perturbing other parts of the system. Moreover, software architecture determines the extent to which different participants will rely on various parts of the core and the gradient of such reliance, i.e., whether a few firms rely heavily, and others not at all, or whether reliance is a slowly decreasing function. Those with a high reliance on a particular component are more likely to help maintain it. Finally, it dictates *numbers* of participants who have a potential interest in maintaining parts of the core. In some cases, large numbers or even all participants have a substantially equal stake in some central components, while in other cases only one or a small number will have substantial reliance. All of these factors, of course, can play a very large role in shaping the payoffs various actors can achieve, and their willingness to act.

In the next sections, we show how game theory and architectural analysis techniques can be used to understand these incentives for participation in the ecology.

2 INCENTIVES IN OPEN SOURCE ECOLOGIES

First, it is important to point out that incentives in open source ecologies are fundamentally unlike those governing project participation by individual volunteers. Researchers have, for example, considered the role of ideology [Stewart et al 2006], skill development, entertainment and personal use [Lakhani et al 2005, Ghosh 2005], and achieving status or "signaling" their value in the labor market to current or future employers [Lerner & Tirole 2002]. There appears to be an interplay of intrinsic and extrinsic motivations [Roberts et al 2006]. While these results shed considerable light on why *individual* volunteers participate in open source projects, we need different approaches for understanding when, why, and how *firms* participate and contribute to public goods.

Game theory provides analytic tools for understanding the complex incentive structure of open source ecologies. It has provided an important theoretical basis for understanding cooperation and competition as represented by simple games such as Prisoner's Dilemma [Axerlrod 1984], how incentive structures govern contributions to public goods [Groves & Ledyard 1977], and how independent agents coordinate their activity [Van Huyck, Battalio and Beil 1990].

As a first step in modeling firms' incentives using game theory, we note, as described in the previous section, that a firm can be regarded as potentially participating in two games involving the decision to participate in the production and distribution of an open source good. These games model similar, though subtly different, components of firms' decisions, and understanding how both games represent a particular ecology is valuable for understanding what determines success.

The first game involves firms' decisions to invest in products or services that are tied to the open source software. The return on such an investment is crucially linked to the contributions of other members. For example, the firm might build an application that enhances the functionality of the open source product, using a specific Application Programming Interface (API) to communicate with it. Many vendors, for example, build plug-ins for the Eclipse development environment that perform useful operations such as code analyses or report generation. Since they are built for Eclipse, the natural market for such a firm's products or services is the installed base of the open source product.

The primary risk is that this market will fail to provide sufficient revenue to justify the investment. This can happen, for example, because not enough others contribute to the production. Situations such as these are modeled by "weak-link" coordination games, in which contributing to production is efficient and profitable, but only if enough others do so [Hirschleifer 1983]. If there is insufficient participation by other vendors, there will not be a rich selection of applications providing the range and variety of functionality that large populations of users require. The ecology would risk being overwhelmed by competitors. In order for a vendor firm to participate, it must conclude that other vendors will also participate.

There are at least two scenarios that make success in such situations more likely. One is if the project has one very large player that is visibly committed to the community. It is likely that this was the case with Eclipse, for example, which began with a very large and visible commitment of resources from IBM. Another plausible scenario, based on our prior work, is when a community starts out very small, building up an expectation of contribution, which continues as the project grows, outpacing the level of contribution that can be achieved when a community starts out large [Weber 2006]. This is arguably the path taken by Apache, which began with a small group of web administrators who needed to share fixes and enhancements to the suddenly-unsupported NCSA httpd server, and grew into a large ecology over a period of years [Mockus et al, 2002]. A third scenario -- motivated by research on "weak-link" games -- involves regular and repeated communication among community members, stating their willingness to continue their involvement with the project and commitment to production [Blume and Ortmann 2007].

The second game governs a firm's decision to "give away" some effort or work products to the community in order to maintain, update, and enhance the public good, i.e., the "free" version of the product. This is the product on which everyone in the ecology depends, both for things built on top of it, and as a "loss leader" to attract new users and grow the market. If the "free" version is highly attractive and useful, it will be more successful in growing the market. This situation creates a classic public goods game, which can be thought of as a variation on the Prisoner's Dilemma (PD). If all players contribute by sharing their effort, everyone makes a (relatively small) profit. If a firm does not contribute, but other players do, that firm makes a larger profit since it has lower costs. But if an insufficient number of players contribute, everyone loses as the software gets out of date and bugs go unfixed. The game becomes complex, as, for example, the distribution of benefits from contributions differs among the firms. One firm may need a particular update more badly than do other firms, and hence may be more willing to contribute. Thus, symmetry in needs and potential contributions might be an important factor determining the success or failure of a particular open source ecology. The role of a central player, such as foundation that owns the software and exerts some control over contributions, may also have an effect due to its influence on the contributions of other players [Hamman et al 2007]. The size of the aggregate contribution necessary to sustain the software functionality may also play a crucial role in determining the success or failure of the ecology (Cadsby and Maynes 1999).

As mentioned in the previous section, both of these games – the moves, the payoffs, and the number of participants – are influenced by the architecture of the software. Understanding the relationship between game-theoretic predictions and laboratory behavior in these games and architectural features and product qualities in open source ecologies would be a valuable contribution.

3 ARCHITECTURAL DESIGN: INCENTIVES, COLLABORATION, AND GOVERNANCE

In order to investigate the games that we believe provide the incentives for firm participation in open source ecologies, we need to be able to analyze software products to determine the specific structure of constraints among components, determine how different parts of the system interact with each other, and what components will be affected by a change to a given part of the software.

The ability to analyze architectures is essential for the following reasons: First, it will allow us to characterize the overall level of modularity of the software, which is a critical factor in its value [Baldwin & Clark 2000, Sullivan et al. 2001]. Second, analysis will help identify the boundaries and payoffs of games played by the various participants. For example, we will be able to identify the extent to which various firms have software that depends on particular open source components, hence motivation to enhance and maintain them. This will give us a way to bound the participants in the various public goods games, and to identify differences in potential payoffs among players.

[insert examples of 2 plugins]

Examining documents such as architecture diagrams may be helpful. However, prevailing box-and-line style design modeling techniques are not designed to generally and comprehensively capture design decisions, their relations, and the modular structure of software components [Sullivan et al 2001; Cai 2006; Huynh et al 2008]. We need a more general design representation that can reveal the modular structure among design decisions, as well as the relation among decision-makers. We propose to extend the prior work of *design structure matrices* (DSMs) [Steward 1981, Baldwin & Clark 2000] for the purpose of architecture modeling.

Design structure matrices (DSMs) [Steward 1981, Baldwin & Clark 2000] are a powerful mechanism for analyzing both the modular structure of software architectures and how these dependencies cause change to propagate through the software. A DSM is a square matrix where

the rows and columns are labeled with design dimensions in which decisions have to be made. A marked cell models the fact that the decision made on the row depends on the decisions made on the column. DSMs are a straightforward but powerful tool for analyzing relations among components, and have been applied to software modular structure analysis, capturing the decomposition of a system into independent modules [Sullivan et al 2001, MacCormack et al 2006, Cai 2006, LaMantia et al 2008].
Using DSM as the core model, prior work has shown that the degree of modularity – i.e, the extent to which components can be developed, changed, and even replaced independently – has a major impact on the value of the software [Baldwin & Clark 2000; Sullivan et al 2001; LaMantia et al 2008]. According to Baldwin and Clark's (2000) theory, modularity adds value in the form of *real options*: a module creates an option to invest in a search for a superior replacement and to replace the currently selected module with the best alternative discovered, or to keep the current one if it is still the best choice. Thus, more modular products should, other things being equal, provide more value to ecology participants [Sullivan et al 2001; MacCormack et al 2006].
Researchers currently build large-scale DSM models from source code, using direct syntactic references as the only source of dependencies [MacCormack et al 2006, LaMantia et al 2008]. However, source code DSMs do not explicitly model implicit design-level decisions and do not express complex logical constraints among the components [Sullivan et al 2001; Huynh et al 2008]. We have shown that manually constructing design level DSMs with design decisions are time consuming and error-prone [Cai and Sullivan 2005, Cai 2006].
We plan to extend current DSM modeling and value-based analysis techniques to account for the relation among incentives, collaboration and governance. Thus, we will first develop the formal foundation of DSM modeling, building on our prior work of *augmented constraint network* (ACN) [Cai and Sullivan 2005, Cai 2006] to logically represent software design decisions and their constraints, enable automatic change impact analysis, and automatically derive DSMs from this. We also plan to extend current DSM/ACN research to explicitly explore the relation between architectural design decisions and the dynamics of communication between the participants. Based on the extended DSM/ACN model, we will identify an interesting variety of games, i.e., clusters of closely-related components and the participants who have contributed to it. We will then use the contribution and communication data -- who has actually contributed what code and what discussions -- to test hypotheses about who will make contributions, and how they differ with differing game size, history, modularity and incentive structure.

4 RESEARCH QUESTIONS

This analysis naturally leads to several fundamental questions about incentives in open source ecologies, and how they are structured by software architecture:

- How does software architecture shape the scope and payoffs of games creating incentives for participation and for public good creation in open source ecologies?

- What characteristics of ecologies (e.g., one large, visibly committed player; game that starts small and grows) lead to games supporting successful evolution?

- How can we measure, and eventually predict, the degree to which an architecture can effectively support an open source ecology?

- How can we predict what kinds of contributions will and will not be made to the public good and by what kind of ecology member?

Addressing these questions would require a combination of qualitative and quantitative empirical approaches, as well as advances in design analysis techniques.

5 ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed.

6 REFERENCES

1. Axelrod, R. (1984). *The Evolution of Cooperation*, New York: Basic Books.

2. Baldwin, C. Y. & Clark, K. B. (2000). *Design Rules, Vol. 1: The Power of Modularity*. The MIT Press.

3. Blume, A. & Ortmann, A. (2007). The effects of costless pre-play communication: Experimental evidence from games with Pareto-ranked equilibria. *Journal of Economic Theory*, 127, 1, 274-290.

4. Cadsby, C. B. & Maynes E. (1999). Voluntary provision of threshold public goods with continuous contributions: experimental evidence. *Journal of Public Economics* 71, 1, 53-73.

5. Cai, Y. (2006). *Modularity in Design: Formal Modeling and Automated Analysis*. PhD thesis, University of Virginia.

6. Cai, Y. & Sullivan, K. (2005). Simon: A tool for logical design space modeling and analysis. Proceedings of the 20th IEEE/ACM *International Conference on Automated Software Engineering*, 329–332.

7. Cataldo, M., Wagstrom, P., Herbsleb, J.D., Carley, K. (2006). Identification of coordination requirements: Implications for the design of collaboration and awareness tools. *Proceedings of ACM Conference on Computer-Supported Cooperative Work,* 353-362.

8. Ghosh, R.A. (2005). Understanding Free Software Developers: Findings from the FLOSS Study. *Making Sense of the Bazaar: Perspectives on Open Source and Free Software*, Joseph Feller, Brian Fitzgerald, Scott Hissam and Karim Lakhani (eds.). Cambridge: MIT Press.

9. Groves T. & Ledyard, J. O. (1977). Optimal Allocation of Public Goods: A Solution to the "Free Rider" Problem. *Econometrica*, Econometric Society, vol. 45(4), 783-809.

10. Hamman, J., Weber, R. and Woon, J. (2007). "Endogenous leadership in public goods." Working paper.

11. Herbsleb, J.D., Mockus, A., Roberts, J.A. (2006). Collaboration in Software Engineering Projects: A Theory of Coordination. *Proceedings of International Conference on Information Systems.*

12. Hirshleifer, J. (1983). From weakest-link to best-shot: The voluntary provision of public goods. *Public Choice*, 41, 371-386.

13. Huynh, S., Cai, Y., Song, Y. & Sullivan, K. (2008). Automatic Modularity Conformance Checking. Proceedings of the 30th *International Conference on Software Engineering* (To appear).

14. Lakhani, K. R. & Wolf, R. G. (2005). Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. *Perspectives on Free and Open Source Software*, edited by Joe Feller, Brian Fitzgerald, Scott Hissam and Karim Lakhani. Cambridge, Mass: MIT Press, 2005.

15. LaMantia, M. J., Cai, Y., MacCormack, A. D., & Rusnak, J. (2008). Analyzing the evolution of largescale software systems using design structure matrices and design rule theory: Two exploratory cases. Proceedings of the 7th *Working IEEE/IFIP International Conference on Software Architectures* (To appear).

16. Lerner, J. & Tirole, J. (2002). Some simple economics of open source. *J. of Industrial Economics*, 50, 2, 197-234.

17. MacCormack, A., Rusnak, J. & Baldwin, C. Y. (2006). Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. *Management Science* 52, no. 7.

18. Malone, T.W., Yates, J. & Benjamin, R.I. (1993). Electronic markets and electronic hierarchies. In T.J. Allen & M.S.S. Morton (Eds.). *Information Technology and the Corporation of the 1990s: Research Studies.* New York, NY: Oxford University Press, pp. 61-83.

19. Mockus, A., Fielding, R., & Herbsleb, J.D. (2002). Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11 (3), 309-346.

20. Powell, W. W. (1990). Neither Market nor Hierarchy: Network Forms of Organization. *Research in Organizational Behavior,* edited by B. Staw and L. L. Cummings. Greenwich, CT: JAI Press. 295-336.

21. Roberts, J. A., Hann, I. & Slaughter, S. (2006). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study. *Management Science*, 52 (7), 984-999.

22. Steward, D. V. (1981). The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, 28(3), 71–84.

23. Stewart, K. J. & Gosain, Sanjay (2006). The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *MIS Quarterly* (30:2), 291-314.

Sullivan, K., Griswold, W. J., Cai, Y., & Hallen, B. (2001). The structure and value of modularity in software design.

24. Steward, D. V. (1981). The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, 28(3), 71–84.

25. Stewart, K. J. & Gosain, Sanjay (2006). The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *MIS Quarterly* (30:2), 291-314.

Sullivan, K., Griswold, W. J., Cai, Y., & Hallen, B. (2001). The structure and value of modularity in software design.