3-1-2004

Organizational Structure of Open Source Projects: A Life Cycle Approach

Donald E. Wynn, Jr.
dewynn@uga.edu

ORGANIZATIONAL STRUCTURE OF OPEN SOURCE PROJECTS: A LIFE CYCLE APPROACH

Donald E. Wynn, Jr.
University of Georgia
dewynn@uga.edu

Abstract

The structure of open source project communities is discussed in relation to the organizational life cycle. In lieu of sales figures, the download counts for each project are used to identify the life cycle stage of a random sample of open source projects. A research model is proposed that attempts to measure the fit between the life cycle stage and the specific organizational characteristics of these projects (focus, division of labor, role of the leader, level of commitment, and coordination/control) as an indicator of the success of a project as measured by the satisfaction and involvement of both developers and users.

Keywords: Open Source, Project Life Cycle, Organizational Structure, Fit

Introduction

Researchers have yet to study the managerial aspects behind the typical open source project despite the prolific increase in the number of projects. Many of the projects are effectively self-managed in accordance with the norms of the open source model, but there is a significant amount of variation in the control, structure, planning, and leading of individual projects that varies between projects and can be studied from numerous aspects such as size, time, developer composition, etc. In this research proposal, we will examine the structure of open source projects as they evolve over time. Specifically, we will study this evolution using a project lifecycle framework. The organizational structure concepts we will include are division of labor, hierarchy of authority, decentralization, and management processes (Robey, 1982).

The first section discusses the concept of project life cycles in general. The following section applies this general view to the specific context of open source projects. We then examine a number of hypotheses followed by an illustrative example. Finally, we discuss the proposed research methodology to study these phenomena along with some implications and conclusions.

Project Life Cycles

The typical Project Life Cycle (PLC) is shown in Figure 1 below. Typically, the dependent variables are sales or profits, even though other dependent variables have been substituted for sales, depending upon specific circumstances or purposes, and the independent variable is time (Levitt, 1966).

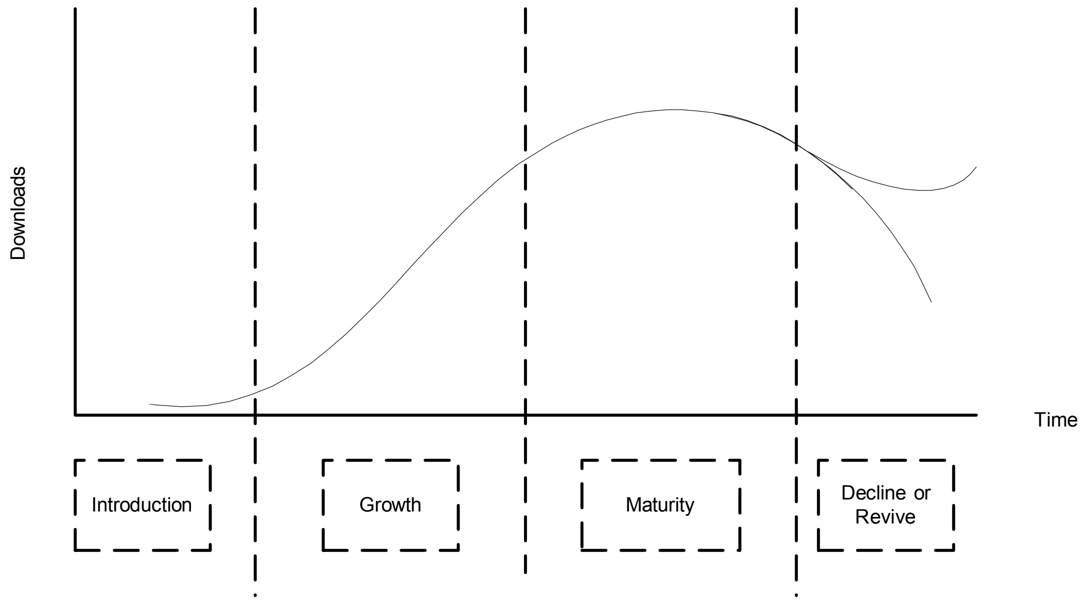

Figure 1: Open Source Project Life Cycle

In the typical life cycle graph, there are four distinct stages. In the introduction stage, an idea is generated and developed. The growth stage is typified by a rapid increase in the sales of a product. In the maturity stage, sales reach a stable maximum point. The fourth stage occurs when sales have begun to decline steadily. Alternatively, sales may rise in response to a change in environmental or market conditions (such as a new fad) or as a result of a change in management policies, hence a rising line instead of a continuous decline. This increase is the distinguishing feature between product decline and revival. In some versions of the model, each phases is broken down into the individual state and the transition into it, for a total of 8 states (Baliga & Hunt, 1988).

These changes are the result of two influences in an organization's life. First, administration of the organization becomes more complex as the size increases and more stakeholders become involved. Second, this increased complexity dictates the increased usage of more sophisticated organizational structure, information processing capabilities, and decision-making style (Miller & Friesen, 1984).

Not all products or industries follow the curve explicitly. Some never attract a critical mass of users and never reach the growth phase. The amount of time spent in a specific stage varies from product to product. Also, the shape of the curve may vary extensively from the ideal case above.

In actuality, there have been numerous attempts to model organizational life cycles as a number of developmental stages. Although previous research has included between three and eight stages, most can be generalized to the typical four-stage model. In nearly all models, these stages are sequential, cumulative, imminent, not easily reversed, and involve a broad range of activities and structures (Quinn & Cameron, 1983; Van De Ven & Poole, 1995).

Open Source Life Cycles

In this section, we will examine each life cycle stage individually in the context of open source projects. We discuss the transitions that lead to each individual stage as well as the managerial/administrative demands and constraints.

Introduction Phase

At the outset, a developer typically encounters a gap between a perceived personal or job related software need and available applications. This results in the initial motivation for a project to develop a software application to fill the gap. An individual developer or small group of developers works to produce an initial version of the application. It is typically the founding developer that is responsible for the creating the initial structure and recruit other team members (Mintzberg, 1984).The founder also is responsible for securing "facilities" such as registration on Sourceforge.net or other sites which gives other users the ability to download and experiment with the application, but the user base is typically very small.

In this phase, the key goal for the developer is not only to produce a working version of the software product and to attract more developers, but to sell the vision for the organization (Ward, 2003). The core group negotiates an informal structure consisting of relatively general roles by each member. The role of the group leader at this stage is similar to that of a quarterback, directing the efforts of the rest of the team (Galbraith, 1982) and communicating the mission and goals of this project (Baliga & Hunt, 1988). Because of the nature of open source organizations, there is no direct assignment of work by the leader; all team members are free to choose their own tasks (Sharma, Sugumaran, & Rajagopalan, 2002). This makes the leader's role more facilitative than directive.

2.2 Growth

The project grows as more users become aware of the existence of an application and it provides them with a solution to a perceived gap in their needs. As a result, the administrative demands of the project increase as well. For instance, there is more feedback from the users regarding feature requests, bugs, support requests, etc. Many of these users are also developers and contribute code to resolve issues they may have with a project. The administrative team has the responsibility of evaluating the contributed code fragments for quality and sufficiency, with the possibility of including the additional code in subsequent releases of the software (and appropriate credit given to the submitter). Because the needs have increased, the admin team adds some of these users to the admin and developer teams. The product itself experiences several releases and numerous supplemental patches and bug fixes throughout this stage.

Because of the increased size and the transient team membership (especially in non-core roles), there is a need for more formalized structure. The Admin assumes the role of an "Accelerator", employing systems and structures to enable the project to grow and to manage the growth (Ward, 2003). Reliance on technological tools such as concurrent versioning systems (CVS), discussion groups, and mailing lists becomes important for coordinating the efforts of the community. As the stage progresses, the increased amount of work allows members to self-select more specialized roles such as code tester, release manager, interface designer, support manager, documentation writer, and bug fixer. The resultant structure is still relatively centralized with the core developers retaining overall project control, but lesser functions are delegated away from the core group.

Maturity

In the third stage, the project approaches critical mass. The number of users and developers grows to a maximum size. Because of the even larger size of the community, the admins are involved in a significant amount of time enforcing policies, evaluating others' code, and other non-development functions. This leads to increasing levels of delegation to the community members. In some cases, the code becomes large enough to warrant multiple versions and releases of the project. Although the user base grows, an increasing percentage of the users are more passive than others.

The central focus of the administrative core group in this stage is to sustain the project (Ward, 2003). Because of the larger project size, there is an organizational need to coordinate and control the diverging structure (Robey, 1982). In traditional organizations, this leads to bureaucratic control. However, open source projects are highly resistant to bureaucracy by nature, so this is an unacceptable response. There is an increase in the level of formality and rigidity, but the overall structure remains quite flexible and organic throughout the life of the project. The admins must be careful to focus on the entire community and not just one set of roles. They must also focus on the long-term effects and not just short-term solutions, especially for projects with a longer time horizon (Smith, Mitchell, & Summer, 1985).

Although most decisions are reached by consensus (Sharma et al., 2002), sometimes conflicts arise among parties within the community, which the admins (who have sole responsibility for code and releases) are well-positioned to resolve or mediate. Rules and norms are established by the core group (in agreement with the norms of open source) to assist in this conflict resolution and community control.

The larger size, high degree of delegation, and self-management by the workers leads to very high levels of task specialization. It is also at this point that attrition and turnover becomes more significant as developers and other active members lose interest in the project. For that reason, another underrated management task is maintaining individual motivation and morale (Baliga & Hunt, 1988). Admittedly the self-management aspects make this a more difficult yet attainable goal.

Decline (or Revival)

As users find other solutions to their products and developers lose interest in continuing to grow the product, the project enters the final stage marked by decreasing users and developers. There are fewer downloads of the product as users are less interested in the project. There is a smaller group of admins and developers remaining, which may not include the founding developer(s). The primary focus of the remaining community is the support and maintenance of existing functionality.

In some cases, there is a revival of the project community in response to a new release, changing environmental or market conditions (release of new operating systems), or simply a late discovery by a group of motivated developers. This revival can cause the project to enter a new growth or maturity stage depending on magnitude.

Leader succession becomes an issue as the original developer may have fulfilled his/her original requirements (technical or personal) and loses interest. At this point, an orderly succession occurs if another member is interested in assuming the leadership role (Raymond, 2001). If not, the project either becomes undirected or the original founder simply ceases to innovate. There are cases where a developer will secede from the original project to form a new development project (or fork) based on the source code. This is generally frowned upon unless there is a conflict between coalitions within the original community that cannot be resolved (Raymond, 2001). In some cases, projects are simply declared inactive and disbanded.

Graphical Examples

By substituting downloads for sales in Figure 1 above, we can identify the development stage for existing open source projects. Download counts can be obtained for each of the 70,000+ hosted projects to identify the current stage. For the examples in this section, monthly download counts were smoothed with a moving-average to reduce noise and establish trending. Note that the scales are not consistent across all six graphs.

The three graphs in Figure 2 below were taken from smoothed download counts for existing open source projects on Sourceforge.net. These graphs indicate projects currently in stages 1, 2, and 3 respectively.

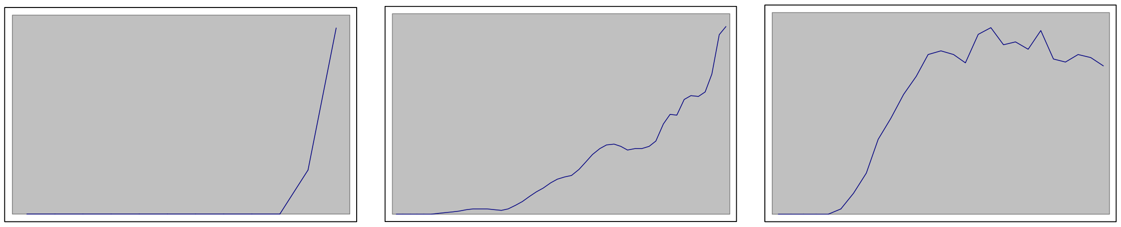

Figure 2: Open Source Project life cycles (Data: Sourceforge.net)

Not all graphs are easily distinguishable, just as not all products, industries, and projects follow the life cycle graph precisely. The typical S-shaped curve is not the only possible life cycle pattern, only the most typical (Wind, 1982). The projects in the Figure 3 each appear to have been in stage four at some point. The first project progressed into a near complete decline. The second project revived and arguably has reverted to a growth stage. The third project had a brief revival followed by the current state of decline.

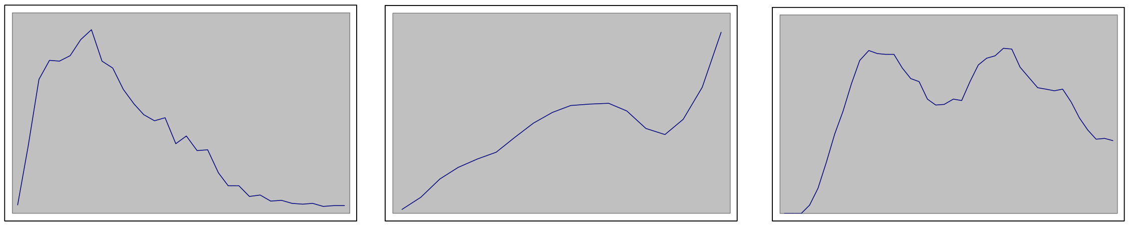

Figure 3: Additional Open Source Project life cycles (Data: Sourceforge.net)

Research Model and Methodology

The research model is shown in Figure 4 below.

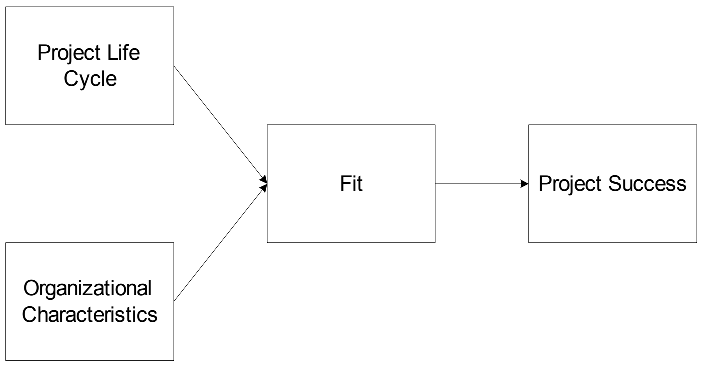

Figure 4: Research Model

We propose (a) that a fit exists between the characteristics of an Open Source Project Team and the stage of the life cycle, and (b) that the level of fit has a positive impact on the success of the project. The level of fit is measured by comparing the deviation of the profile of existing open source organizations to the hypothesized profiles (Venkatraman, 1989; Venkatraman & Prescott, 1990). Table 1 lists several characteristics of open source projects at each stage of growth.

Table 1: Theoretical OS Project characteristics at each stage (Galbraith, 1982; Quinn & Cameron, 1983)

Stage	**Introduction**	**Growth**	**Maturity**	**Decline or Revival**
Primary Focus	Idea Generation	Expansion	Stability	Adaptation
Structure	Completely informal	More formal; Centralized	Somewhat formal; Decentralized	Slightly formal but less adherence
Division of Labor	Generalists	Some specialization	Highly specialized	Less specialized
Leader Role	Quarterback	Player/Coach	Manager	Strategist
Coordination	Informal; one-on-one	Technology introduction	Formal; Technology-intensive	Formal but less adherence
Level of Commitment	High (mostly founders)	Maximum	Waning	Decline: low; Revival: growing

The level of coalignment between life cycle stage and characteristics affects the project success. For the purposes of this study, success is measured by the satisfaction and involvement of both users and developers (Crowston, Annabi, & Howison, 2003).

A random sample of 150 open source projects will be taken from data provided by Sourceforge.net. Each project will be evaluated to determine their current life cycle stage (where possible) using download counts. Next, the project admins, developers, and several identifiable users for each evaluated project will be contacted via email to request completing a brief questionnaire to measure the current focus of the project, formal structure, division of labor, leader role, coordination, level of commitment, user success, and developer success. The resultant data will be grouped to develop profiles and success measures for each individual project. As mentioned above, these profiles are compared to t to the theoretically stage profiles. This deviation can be conceptualized as a Euclidean distance between sample and theoretical profiles, which is calculated as follows (Venkatraman & Prescott, 1990):

$$\text{MISALIGN} = \Sigma_j (b_j (X_{sj} - X_{cj}))^2$$

Where Xsj = score for jth variable of the sample profile
Xcj = score for jth variable of the theoretical profile
Bj = weight of the jth variable in the environment (constant = 1.0 in this study)
J = 1,n where n is the number of characteristics in each profile.

Lower values indicate a better fit between profiles, which is expected to be negatively correlated to success measures for each open source project.

Conclusions

Open source projects offer a promise of inexpensive, efficient, high-quality software development for both corporate and individual purposes. By studying the life cycle of open source projects, we hope to be able to prescribe an effective set of structural features for the creation and evolution of successful new projects. This study is a first step in that direction.

Subsequent research should include more in-depth analysis of the control functions within an open source project, reward structures, the impact of user participation, and reasons for the attrition of key developers and users.

References

Baliga, B. R., & Hunt, J. G. (1988). An Organizational Life Cycle Approach to Leadership. In J. G. Hunt & B. R. Baliga & H. P. Dachler & C. A. Schriesheim (Eds.), *Emerging Leadership Vistas* (pp. 129-149). Lexington, MA: Lexington Books.

Crowston, K., Annabi, H., & Howison, J. (2003, Dec 14-17, 2003). *Defining Open Source Project Success.* Paper presented at the International Conference on Information Systems, Seattle, WA.

Galbraith, J. (1982). New Venture Planning - The Stages of Growth. *Journal of Business Strategy, 3*(1), 70-79.

Levitt, T. (1966). Putting the Product Life Cycle to Work. *Harvard Business Review*, 19-23.

Miller, D., & Friesen, P. (1984). A Longitudinal Study of the Corporate Life Cycle. *Management Science, 30*(10), 1161-1183.

Mintzberg, H. (1984). Power and Organization Life Cycles. *Academy of Management Review, 9*(2), 207-224.

Quinn, R. E., & Cameron, K. (1983). Organizational Life Cycles And Shifting Criteria of Effectiveness: Some Preliminary Evidence. *Management Science, 29*(1), 33-51.

Raymond, E. (2001). *The Cathedral and the Bazaar, Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, CA: O'Reilly & Associates.

Robey, D. (1982). *Designing Organizations: A Macro Perspective*. Homewood, IL: Richard D. Irwin, Inc.

Sharma, S., Sugumaran, V., & Rajagopalan, B. (2002). A Framework for creating hybrid-open source software communities. *Information Systems Journal, 12*(1), 7-25.

Smith, K. G., Mitchell, T., & Summer, C. (1985). Top Level Management Priorities in Different Stages of the Organizational Life Cycle. *Academy of Management Journal, 28*(4), 799-826.

Van De Ven, A. H., & Poole, M. S. (1995). Explaining Development and Change in Organizations. *Academy of Management Review, 20*(3), 510-540.

Venkatraman, N. (1989). The Concept of Fit in Strategy Research - toward Verbal and Statistical Correspondence. *Academy of Management Review, 14*(3), 423-444.

Venkatraman, N., & Prescott, J. E. (1990). Environment Strategy Coalignment - an Empirical-Test of Its Performance Implications. *Strategic Management Journal, 11*(1), 1-23.

Ward, A. (2003). *The Leadership Lifecycle*. Houndsmill, Basingstoke, Hampshire: Palgrave MacMillan.

Wind, Y. J. (1982). *Product Policy: Concepts, Methods, and Strategies*. Reading, MA: Addison-Wesley.