Available online at www.sciencedirect.com
ScienceDirect
Int. J. Human-Computer Studies 66 (2008) 484–494
International Journal of Human-Computer Studies
www.elsevier.com/locate/ijhcs

An institutional analysis of software teams

Josh Tenenberg*

School of Informatics, Indiana University, 901 East 10th Street, Bloomington, IN 47408, USA

Available online 29 August 2007

Abstract

Modern software is constructed by teams of software developers. The central question that this paper addresses is what policies should be enacted for structuring software teams to enhance cooperative as opposed to self-serving behavior? The contribution of this paper is in viewing software teams as being subject to a set of well-understood *collective action* problems: there are individual incentives to receive the joint rewards for a team-developed software project without contributing a fair share to its development. In this paper, an *institutional analysis* perspective is used in presenting a set of theoretical principles and an analytical framework recently developed in game theory, political economy, experimental economics, and natural resource governance for the understanding and resolution of these collective action problems. The principles and analysis framework are applied to an empirical case study of software teamwork within an academic setting. This case study shows, first, how to apply the analytic framework on an actual collective action situation. Second, it demonstrates how the theoretical understandings can be used as a basis to account for outcomes within this setting. And third, it provides an example of a particular institutional arrangement that elicits high levels of cooperation and low levels of free riding within a real-world setting. Understanding the importance of institutions for shaping individual and social behavior within software development teams makes these institutions more amenable to intentional human design.

Keywords: Free riding; Cooperation; Software management; Teamwork; Social dilemma; Collective action problem

1. The social nature of software development

Modern software is constructed by teams of software developers and used within social settings. Cain et al. (1996) capture the inherently social nature of software development:

> Software development is a predominantly social activity. It is important to view software development groups, departments, and corporations as social bodies. ... The essentially human nature of customer interactions, programmer creativity, and programming team dynamics demand that we deal with the social side of software production enterprises (Cain et al., 1996).

The central question that this paper addresses is what policies should be enacted for structuring software teams to enhance cooperative as opposed to self-serving behavior? The contribution of this paper is in viewing software teams as being subject to a set of well-understood *collective action* problems: "[a]ll efforts to organize collective action, whether by an external ruler, an entrepreneur, or a set of principals who wish to gain collective benefits, must address a common set of problems. These have to do with coping with free riding, solving commitment problems, arranging for the supply of new institutions, and monitoring individual compliance with sets of rules" (Ostrom, 1990, p. 27). Software teams are as subject to these collective action problems as other settings in which the institutional approach has been used, such as governance of the nation-state (Helmke and Levitsky, 2006) or shared natural resources (Ostrom, 1990). Individual software developers make choices about the extent to which they contribute to the joint production of digital artifacts and the extent to which they free ride on the effort of others so as to reap the benefits without paying the costs. In this

*Corresponding author at: Department of Computing and Software Systems, Institute of Technology, University of Washington, Tacoma, 1900 Commerce Street, Tacoma, WA 98402, USA. Tel.: +1 253 6925860; fax: +1 253 6925862.
E-mail address: jtenenbg@u.washington.edu
URL: http://faculty.washington.edu/jtenenbg/

1071-5819/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ijhcs.2007.08.002

paper, an *institutional analysis* perspective is used in presenting a set of theoretical results and an analytical framework recently developed in game theory, political economy, experimental economics, and natural resource governance for the understanding and resolution of these collective action problems. The novelty is in bringing these insights to the enterprise of software development in both commercial and academic settings.

Key to this social science research is its focus on the *institutions* that people develop to organize their collective action. North (1990) defines institutions as "the rules of the game in a society or, more formally, ... the humanly devised constraints that shape human interaction." These institutions indicate what people are permitted, required, and prohibited from doing, under what circumstances, and under what costs if they fail to do so. "Institutions" and "rules" will be used synonymously throughout the balance of this paper. Understanding the importance of institutions for shaping individual and social behavior within software development teams makes these institutions more amenable to intentional human design.

The paper will be structured as follows. I will begin by discussing existing accounts of forms of organization for the management of commercial software teams. These are centered around relationships of *control* between managers and developers to increase the likelihood that developer effort will be directed toward achieving organizational goals. These accounts, however, do not provide a sufficiently fine-grained analytic understanding of how to resolve these problems of cooperation. I then draw on research from experimental economics, computer simulation in social science, and natural resource governance to highlight a set of key theoretical principles associated with institutions that shape human behavior in collective action settings. These highlight the importance of face-to-face communication, repeated interactions, monitoring of rule compliance, and sanctions for non-compliance. Following this will be a discussion of an analytic framework for understanding institutions in existing collective action settings. This framework is useful for developing a fine-grained understanding of particular institutional forms in existing collective action settings.

I then provide an empirical case study of software teamwork within an academic setting. This case study shows, first, how to apply the analytic framework on an actual collective action situation. Second, it demonstrates how the theoretical understandings can be used as a basis to account for outcomes within this setting. And third, it provides an example of a particular institutional arrangement that elicits high levels of cooperation and low levels of free riding within a real-world setting. Finally, I reiterate the argument for the value of the institutional analytic approach, and summarize implications for both research and practice. For research, these include the use of the analytic tools and theoretical understandings in the design of subsequent studies to examine the relationship between different institutions and their effectiveness in eliciting cooperation. For practice, these include establishing conditions that enable self-governance among developers to emerge, and seeking to reduce the costs of monitoring and sanctioning.

2. Background literature

2.1. Forms of control in commercial software teams

Miller (1990) states the central collective action problem associated with teamwork:

> In [a simple team setting] ... individuals would be better off working hard than shirking. ... If the other works hard, each person is better off shirking. If the other one shirks, each is better off shirking. Each person has a dominant strategy to shirk, despite the fact that [they] are worse off when each chooses his or her dominant strategy.

Within teams, including software development teams, the pursuit of individual self-interest can lead to social inefficiency, what we have been calling a *collective action problem* (Ostrom, 1990) (or, what is sometimes termed a *social dilemma* (Miller, 1990)). There are always incentives to obtain the benefits associated with team-based production without carrying out a fair share of the work, what we have been calling *shirking*, or *free riding*. How then do individuals organize so as to get the benefits of collective action when they face social dilemmas? How do they constrain their own and one another's selfish impulses for greater individual and collective benefit?

Much of the management science literature on software team organization is centered on the issue of *control* of software development labor. According to Kirsch et al. (2002), "[c]ontrol is defined as the set of mechanisms designed to motivate individuals to work in such a way that desired objectives are achieved". Borrowing from Ouchi (1979), Henderson and Lee (1992) take control to require monitoring and evaluation of both software developer *behavior* and *outcomes*. They consider that there are two main sources of control within a software development organization: managerial and team-member control. They argue that both forms of control are necessary for effective software teamwork, but that they are differentially suited to different kinds of monitoring and evaluation.

> [M]anagerial control appears more effective when it is behavior oriented while team member control is more effective when it is outcome oriented. This suggests that effective teams have a manager with the skills and capabilities to influence how work is accomplished while the pressure to meet deadlines and commitments arises from one's peers (Henderson and Lee, 1992).

Kirsch (1997) develops an alternative control taxonomy for software teams. She takes behavioral and outcome control as basic control regimes that are primarily the responsibility of hierarchical management structures.