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.

These regimes are established through formal rules and procedures, such as explicit development methodologies, work assignments, performance evaluations, and evaluation of project milestones. Kirsch also states that there are informal forms of control: self-control and clan control. Self-control is where "an individual sets his own goals for a particular task, then proceeds to self-monitor, self-reward, and self-sanction."

Kirsch borrows the notion of clan control from Ouchi (1979), who was concerned with the difficulties associated with purely managerial control: "a control mode that depends heavily on monitoring, evaluating, and correcting in an explicit manner is likely to offend people's sense of autonomy and of self-control ... In this state, people will require even more close supervision, having been alienated from the organization as a result of its control mechanism." Clan control involves careful screening and selection of organization members, and a lengthy training and socialization process that allows for the internalization of shared values and norms. Control is achieved because a clan member "wants to identify with and emulate a respected person or group" (Ouchi, 1979). Kirsch recommends that management use a "portfolio" of control modes for the structuring of software teams, and, like Henderson and Lee (1992), indicates that different forms of control are appropriate for different aspects of the development task. Formal modes of control are "essential for coordinating task activities by monitoring specific behaviors and outcomes" while informal controls are "a means of fostering relationships to ensure cooperation."

Although the concepts of behavioral, outcome, and clan control are important insights, they nonetheless are inadequate for understanding and designing software team structures. This stems first from their inability to ground the different modes of control in individual choice. That is, might not similar mechanisms of individual choice operate regardless of the form of external control, whether from past socialization, one's peers, or one's supervisors in a hierarchy? And second, any theory of team "management" should apply to diverse settings, including student software teams and open source development efforts. Given the different goals in these other settings, however, a discourse centered around "control" makes less sense. Rather, conditions that elicit cooperation and incentives for individuals to choose to do so appear to be much more useful directions to pursue, to which we now turn.

2.2. Institutions, choice, and cooperation

2.2.1. Definitional preliminaries concerning institutions

Ostrom (2005) defines institutions in a manner similar to that of North (1990), as "prescriptions that humans use to organize all forms of repetitive and structured interactions including those within families, neighborhoods, markets, firms, sports leagues, churches, private associations, and governments at all scales." One of the virtues of Ostrom's definition, is that it both highlights the diversity of human settings in which rules are used (i.e. any repeated, structured social setting), as well as the fact that rules are often taken for granted by the participants themselves and not explicitly stated or written. We thus can distinguish between those rules that have explicit external representation—*rules-in-form* (what North (1990) calls *formal institutions*)—and the rules that participants take as actually operating within a particular setting, the *rules-in-use*. These rules-in-use can "reinforce, subvert, and sometimes even supersede formal rules, procedures, and organizations" (Helmke and Levitsky, 2006).

Implicit in this perspective is that institutions always exist, whether by default, by trial and error, or by explicit design. What this means for software teams, is that rules-in-form exist to structure virtually all of the human interactions: those rules between management (or, in an academic setting, the instructors) and development teams that determine development methods, standards, and evaluations of projects and personnel, and rules (whether explicitly encoded or implicitly understood) among the software developers themselves that mediate such things as coding standards, code ownership, code testing, and conflict resolution.

Rules prescribe what people may, must, or must not do under particular conditions with particular costs for non-compliance. Ostrom (2005) defines rules, whether tacitly understood or explicitly formalized, as having the following five elements:

Attribute: the set of individuals to whom the rule applies;
Deontic: one of "may", "must", or "must not";
Aim: constraints on actions or outcomes;
Conditions: when and where the rule applies;
Or-Else: the consequences for when the rule is not followed.

An example of a rule from the case study below concerning student teams is: *Group reports are required the first class session of each week*. The *attribute* is contextually understood as all students in the course, the *deontic* is taken as "must", the *aim* is that group reports are to be written and handed in, the *condition* is that this is to occur at the first class session of each week, and the *Or-Else* is taken implicitly as "otherwise your team will receive a score of zero for the week." Costs associated with making rules, with monitoring compliance, and with sanctioning non-compliance are termed *transaction costs* (Levi, 1988; Miller, 1990). Ostrom distinguishes rules from *norms*, in that norms have all of the same elements except for the *Or-Else*, i.e. norms do not include a sanction for non-compliance.

2.2.2. Rational choice and cooperation

Institutions shape human behavior in that they "structure incentives in human exchange" (North, 1990). North states that these incentives reduce the uncertainties associated with not knowing the actions that others will take in the absence of these rules. Under certain rule configurations, individuals face social dilemmas that pit

individual self-interest against collective interest. If people were purely self-interested utility maximizers, they would always pursue their own interests at the expense of the collective in any social dilemma. This is what Hardin (1968) referred to as the "tragedy of the commons": each animal herder using a grazing commons agrees to limit their use of the commons so as to induce others to likewise limit their use, but then to privately overgraze their own animals so as to reap additional individual benefit.

> Therein is the tragedy. Each man is locked into a system that compels him to increase his herd without limit—in a world that is limited. Ruin is the destination toward which all men rush, each pursuing his own best interest in a society that believes in the freedom of the commons. Freedom in a commons brings ruin to all (Hardin, 1968).

Despite Hardin's pessimistic prediction, there is considerable evidence that people do not always follow a selfish strategy in social dilemmas, and that people engage in behavior to alter the rules under which they operate to try to reduce incentives for free riding. Not all commons are overgrazed; not everyone free rides on the efforts of others; ruin is only one of several possible futures. One set of evidence stems from experimental economics, where individuals interact within a controlled, experimental setting using game-theoretic rules that create collective action problems, such as the Prisoner's Dilemma (Rapoport and Chammah, 1965). This work is supplemented by computer simulations of alternative strategies under various game-theoretic rules. The second set of evidence stems from field studies in a variety of communities across the globe where community members jointly manage shared resources, such as pasturelands, forests, fisheries, and groundwater. Citations to some of these literatures are provided in the balance of this section centered around some of the key insights that this research has revealed.

2.2.2.1. Face-to-face communication. In lab experiments where players are engaged in repeated social dilemma games under conditions of anonymity and independent action, the "tragedy of the commons" (i.e. players use equilibrium allocation strategies that are collectively inefficient) has been repeatedly observed (Ostrom et al., 1994; Cardenas et al., 2000; Casar and Plott, 2003). However, simply changing one condition—allowing players to engage in face-to-face conversation—has a dramatic effect on outcomes, with players approaching optimal allocations (Ostrom et al., 1994). Ostrom et al. (1994) conjecture that there are several reasons for this. First, "[s]imply promising to cut back [i.e. be more cooperative] on their investments in the common-pool resource led most subjects to change their investment pattern." Second, participants freely expressed their anger when others were found to have broken promises made in previous rounds of play (e.g. "some scumbucket is investing more than we agreed upon" (Ostrom et al., 1994)). And third, promise breakers reverted to cooperative behavior after receiving verbal sanctions from the other players, even under conditions where behavior of the previous round of play is only reported in the aggregate (i.e. promise breakers cannot be individually identified).

What is not clear from this research is whether face-to-face communication might be replaced by electronically mediated communication to achieve the same effects. Perhaps with sufficient information bandwidth, real-time interaction, and effective interface design, similar such effects could be obtained from geographically dispersed but electronically mediated groups.

2.2.2.2. Repeatability and reciprocity. In perhaps the most well-known computer simulations in social science, Axelrod (1984) ran a computer tournament of alternative strategies to a repeated Prisoner's Dilemma game, where strategies were solicited from game-theorists, mathematicians, and social scientists, encoded as computer programs, and paired against all others in a series of computer simulations. One strategy, known as TIT-FOR-TAT (cooperate on the first play and on each subsequent play take the action that the other player did on the previous play), performed best in the tournament. Even after publishing these results, resoliciting new strategies, and rerunning the tournament, TIT-FOR-TAT was still the most successful strategy. Axelrod's account for these results highlight the importance of reciprocity and trust-building.

> What makes it possible for cooperation to emerge is the fact that the players might meet again. This possibility means that the choice made today not only determines the outcome of this move, but can also influence the later choices of the players. The future can therefore cast a shadow back upon the present and thereby affect the current strategic situation (Axelrod, 1984).

Repeatability, however, is a two-edged sword, since it can both allow cooperation to emerge, as well as give rise to increased non-cooperation in the presence of persistent free riders. Marwell and Ames (1979) report on a set of one-shot experiments where people contribute substantial amounts toward public goods, i.e. goods that are usable by all parties simultaneously (*non-subtractible*) and where it is difficult to exclude any parties from their use (*non-excludable*). "[I]n replication after replication, regardless of changes in a score of situational variables or subject characteristics ... [p]eople contribute substantial portions of their resources—usually an average of between 40 to 60 percent—to the provision of a public good". Yet when these experiments are altered with only the addition of repeated interactions among the same players "after five trials, the contributions to the public good were only 16 percent of optimum" (Thaler, 1991, p. 11). The lab experiments indicate that not everyone free rides, but that there are few who will persist in contributing to public goods in the absence of similar such cooperative efforts by

other users of the good; no one wants to be a sucker. Cooperation in this collective action settings is contingent on the cooperation of others. Repetition can thus lead to either virtuous circles or vicious circles by bringing the horizon of the future into focus when making current choices.

2.2.2.3. Monitoring. Strong evidence for the importance of monitoring comes from Elinor Ostrom and her colleagues at the Workshop on Political Theory and Policy Analysis at Indiana University (Ostrom, 1990). In this research, they examined case histories of several hundred communities who had jointly managed common resources over many years, such as pasturelands, forests, fisheries, and groundwater, all situations in which cooperation is contingent and free riding is an ever-present danger. Their analysis yielded a set of "design principles" that characterized those groups who had successfully managed their resources. Key among these was not only monitoring, but *mutual* monitoring by the community members of one another. For example, in research that they cite by Maass and Anderson (1986), farmers in the semi-arid Marcia and Orihuela region of Spain evolved institutions where the method of water allocation from diverted river water is that each farmer receives a particular time-slice of water from the canal. This simplifies and reduces the cost of monitoring, in that the farmer allocated to receive the next time-slice has an incentive to monitor the recipient of the current time-slice of water. In addition, guards are also hired by the farmers to monitor compliance, thus providing two levels of monitoring. Monitoring is thus low-cost because it is distributed across the resource users, and it occurs at multiple levels.

Making the results of monitoring *public* also serves further to induce rule compliance. This is what Wade (1988) calls *transparency* of institutional arrangements. By an analysis of historical documents among a large group of cooperative water distributors in two groundwater systems in the Los Angeles area, Blomquist (1994) points out how institutional transparency can achieve high rates of rule compliance over long periods of time. "Although the institutional arrangements have been in effect in Raymond Basin for nearly 50 years and in Orange County for 40 years, sanctions have never been applied for non-compliance. When instances of non-compliance with rules requiring meter installations, meter repairs, payment of contributions, or restrictions on water withdrawals have occurred, *reporting of the violation has sufficed to bring about compliance in the next time period without the application of sanctions* [emphasis added]."

The fact that group members themselves do most of the monitoring has less the character of surveillance by a superior with its attendant loss of autonomy and alienation, and seems closely related to direct protection of ones own interests among peers. This suggests that "clan control" might require fewer shared norms and values, but require more mutual oversight so that individuals have higher degrees of information about one another's compliance.

2.2.2.4. Sanctions. The fact that sanctions were not used in the above groundwater case does not mean that the *threat* of sanctions is unimportant. Sefton et al. (in press) report on an experiment in which players are allowed both to monitor one another's actions in previous rounds of a game that involves contributing to a public good and to levy sanctions to specific individuals based on their level of contribution to the public good in previous rounds of play. The authors conclude that "we find that sanctioning sustains public goods provision at a level above that observed in the absence of sanctioning opportunities", even though there was a cost to both the giver and receiver of sanctions, and even though levels of sanction were reduced over time.

Ostrom (1990) as well documents the importance of mutual and multi-level sanctioning among the successful resource managing communities studied. In these communities, rules for sanctioning start with light sanctions for the first infraction and increase in severity with additional infractions.

Levi (1988) also underscores the importance of sanctions in a common situation of contingent cooperation: that of taxpayers within a polity. Because of the high transaction costs associated with surveillance by the state, systems of taxation are unsustainable unless taxpayers themselves are primarily responsible for complying with tax laws. She calls taxpayer behavior in these situations *quasi-voluntary* compliance: "It is *voluntary* because taxpayers choose to pay. It is *quasi*-voluntary because the non-compliant are subject to coercion—if they are caught" (Levi, 1988). Sanctions play an important role in ensuring that everyone is playing by the same rules, thus creating conditions that give rise to quasi-voluntary compliance. As Levi states, "[t]he importance of deterrence [i.e. sanctions] is that it persuades taxpayers that *others* are being compelled to pay their share."

In summary, the social science research cited provides a number of insights into determinants of cooperative behavior, all of which can be applied to the structuring of team-based software development. These stress the importance of face-to-face interaction among the participants, repeated interactions, mutual, multi-level, and public monitoring, and sanctioning mechanisms to create disincentives for free riding. Framing these insights within an institutional analytic framework amounts to understanding and designing rules-in-use that create these conditions under which cooperation is known to emerge. In the next section, we look at one such framework for institutional analysis, and apply this framework to an empirical case study in the section following.

3. The Institutional Analysis and Development framework

The Institutional Analysis and Development (IAD) framework developed by Ostrom and her colleagues

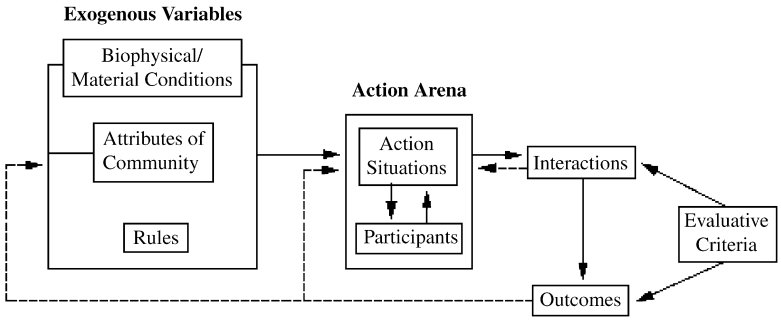

Fig. 1. IAD framework. *Source*: Ostrom (2005).

provides an analytical tool for unpacking the institutional structure of collective action settings (Ostrom, 2005). This section provides an overview of the IAD framework, summarizing the description in Ostrom (2005).

Fig. 1 provides a graphical overview of the main elements of the IAD framework (in boxes), the causal relationships between the elements (solid arrows) and the feedback relation between the elements (dotted arrows). An *action situation* exists "[w]henever two or more individuals are faced with a set of potential actions that jointly produce outcomes" (Ostrom, 2005). The action situation and the individuals (or *participants*) constitute the *action arena*. In the case study to follow, the action arena is student software teams within a software engineering course, with the participants being the students and the instructor. There are interactions between the participants and their external world, which generate outcomes that can be evaluated using a variety of criteria (e.g. self-reported learning, quality of software produced, etc.). The exogenous variables consist of the biophysical conditions, community attributes, and rules-in-use, and are taken as fixed for purposes of analysis of a particular action arena.

Rules can be thought of as the main "degrees of freedom" that groups can use to structure their interactions, since biophysical conditions and cultural characteristics tend to be much more resistant to deliberate and predictable control. For purposes of analysis, the IAD framework distinguishes between three nested levels of rules:

operational rules determine the day-to-day operations associated with resource extraction and contributions to public goods;

collective choice rules concern how the operational rules will be monitored, the sanctions for non-compliance, the rules used for changing operational rules, and the set of people eligible to change the rules at the operational level; and

constitutional rules determine how the collective choice rules will be monitored, the sanctions for non-compliance, the rules for changing collective choice rules, and the set of people eligible to change the rules at the collective choice level.

The internal structure of action situations consists of the following seven elements (italicized): *participants* who hold *positions* who can select with some amount of *control* using a given amount of *information* over a set of *actions* that yield *outcomes* with particular *costs and benefits*. Rules can be grouped semantically according to the element of the action situation that the rule primarily impacts:

Position rules define the different positions (or roles) within the action situation and the associated actions.

Boundary rules indicate which participants can take on which positions, and how such positions are entered and exited.

Choice rules specify the conditions under which actions that participants in particular positions must, must not, or may take.

Aggregation rules determine who has control, in what combinations (e.g. by a designated individual, by majority vote, by consensus) over a decision.

Information rules govern what information needs to be provided to what participants by whom.

Payoff rules specify the rewards and punishments associated with actions or outcomes.

Scope rules: constrain outcome variables that may, must, or must not be affected by actions within the arena.

4. Applying the IAD: a case study of student software teams

4.1. Case rationale

In this section, I present an empirical case study to illustrate the application of the IAD in an actual setting, to connect theoretical understandings to actual software development behavior, and to provide an example of a particular rule configuration that leads to high levels of cooperative behavior.

The case study is used as the main method for achieving these goals, in contrast to, for example, the experimental or quasi-experimental study (Campbell and Stanley, 1963). Experimental designs rely upon high levels of control over individual variables. It is less useful when there are multiple independent variables (such as degree and type of monitoring, sanctioning, and interaction among participants) that interact in complex ways and where the final outcome depends upon the conjunctive (or *configurational* (Blomquist, 1994)) interaction among the variables, as is

the case with field studies on social dilemmas outside the behavioral lab. As Yin (2003) points out, the case study is particularly appropriate in settings where there is a strong connection to theory, where the background contextual factors and the phenomena of interest are tightly coupled, and where multiple sources of data can be triangulated in the analysis, as is the case here.

It is important to underscore that the point of this case study is not to argue that the particular rule set used here is optimal or that it is appropriate to apply outside of its specific setting of use. Rather, it is to show that the theoretical principles discussed—concerning face-to-face interaction, repeatability, monitoring, and sanctioning—can be instantiated within actual settings so as to elicit high levels of cooperation. These theoretical principles are not specific to particularities of *student* software teams, but are based on the structural characteristics of social dilemmas, universals of human behavior explored in the behavioral lab, and characteristics of successful social behavior uncovered by analyzing collective action in a variety of field settings.

This case study is within the academic (as opposed to commercial or open source) setting for four reasons. The first is the fact that this is the setting for software development that I know best, both as an educator and as an educational researcher. Second, and as a consequence of the first reason, the courses that I teach have provided the opportunity to capture opportunistic data about the rules under which students work and the outcomes associated with their actions. Third, academic settings have characteristics that make application of the literature from management science on clan and hierarchical controls problematic, since student teams cannot rely upon shared norms and values nor on a strict hierarchical control regime. And finally, there is a dearth of literature within education broadly and computing education specifically on how to structure teamwork so that it serves the learning goals.

4.2. Case specifics

This case study examines both a graduate-level and undergraduate-level *Introduction to Software Engineering* course at the University of Washington, Tacoma. Data were collected from all students in the undergraduate course during Autumn 2004 and Winter 2004, and in the graduate course in Spring 2004 and Spring 2005, and all documents handed out to students by the instructor were analyzed using the IAD. Of these, 17 of 30 undergraduate students and 21 of 34 graduate students consented to the use of their data for analysis. I discuss the rules used for structuring student teamwork, present evidence for the effectiveness of these rules in eliciting cooperation and discouraging free riding, and conclude with an account that relates theoretical principles for eliciting cooperation to the rule structures used and the associated outcomes.

In teams of 3 or 4, during the course of the academic term, students develop a piece of software of several thousand lines of code, comprising approximately 15–20 computational modules. The project has three distinct *phases*, each lasting approximately three weeks: the *requirements*, the *design*, and the *code and test* phases. In this regard, the project follows a standard *waterfall* model of software development (Boehm, 1997). Associated with the end of each phase is a set of documents, called a *milestone*, which is handed in by the entire team.

The great majority of undergraduates in this course took their first two years of college courses at a regional community college. Their average age is in the mid-20s and all but a small handful of students work at least part-time. All students commute to the university, as there are currently no residential facilities, and commute anywhere from 10 min to 1 h, though commute times are frequently longer due to traffic delays. The majority of undergraduates work at least part-time. The undergraduate version of this course is between 75% and 85% male, while graduate courses average 65% male. The majority of students are Caucasian, though there are significant numbers of first- and second-generation immigrants as well. The graduate version of the course is for students who do not have an undergraduate software-related degree (such as Computer Science). Approximately 2/3 of the graduate students are male, and all work at least part-time.

4.3. Rules

A key feature of the rules under which teams worked is that all *operational* rules concerning the day-to-day allocations of work are determined completely by the students, while *collective choice* rules that concern payoffs (i.e. criteria for grading/marking), information sharing, and face-to-face meetings are instructor-specified within the syllabus.

4.3.1. Operational rules

Based on an analysis of weekly reports from each student team (described below), the operational rules that students developed primarily concern the determination of a task decomposition for the following week, allocating these tasks among the team members, monitoring previous individual task completion and monitoring collective progress toward the next milestone. The student-developed operational rules change daily to reflect the students' changing understanding of the project, individual needs for varying workloads (e.g. when children are ill or when there are extra demands at their workplaces), differences in expertise, and different understandings of the need for coordination and integration of work undertaken individually. Some groups carry out virtually all of their work together in the computer labs, while other groups carry out virtually all of their work at their individual homes, meeting together for the minimal weekly group meeting

to "check in" their previous work and determine new work allocations.

4.3.2. Collective-choice rules

In contrast to the quickly changing operational rules, the collective-choice rules specified in the syllabus (and other instructor-provided documents) are constant across the academic term. These rules primarily specify key positions, information sharing, and payoffs.

Position rules: There are two explicitly stated team roles: the *scribe* who records the minutes for that week's meeting (described below) and the *facilitator* who is responsible for chairing the weekly meeting.

Boundary rules: At the first team meeting, participants determine a rotation so that every person takes on each role in sequence.

Choice rules: All students are required to meet weekly, face to face with their group for a minimum of 1.5 h, to read and respond to email daily, and to not share the team's intellectual work products with other teams, except during designated times in the classroom (e.g. during cross-team design reviews).

Aggregation rules: No aggregation rules were specified.

Information rules: After each regular weekly meeting, each team writes a report that includes minutes of the meeting (including duration, location, and who attended), a *task matrix* that states for each team member what tasks they have committed to complete by the next week and what tasks they completed in the previous week, and an *equity estimator* that estimates the percentage of team effort each team member is undertaking during the upcoming week and undertook during the previous week. The task matrix and equity estimator are collectively constructed as a part of each weekly meeting. Scribes disseminate typed weekly reports to teammates within 24 h of each meeting.

One additional information rule is that at the end of each milestone, students are required to provide to the instructor an evaluation of their own effort and that of their teammates in carrying out the milestone along a number of different dimensions (e.g. "Ability to meet deadlines", "Attendance at meetings"). These evaluations are kept confidential from the other team members.

Payoff rules: Ten percentage of a student's final grade stems from their group's weekly reports, and 70% of their grade stems from their team project. Team projects are assigned a score, and each individual is awarded that score times an *individual multiplier*, a real number between 0 and 2, with a default value of 1. This multiplier reflects contribution toward team effort as documented in weekly reports and self- and mutual-evaluations.

Scope rules: No scope rules were specified.

4.4. Evidence for cooperation and free riding

The data used to evaluate outcomes consist of weekly reports, project milestone documents, individual student reports at the end of the milestones (including self- and mutual-evaluations), a pre-survey and post-survey that students filled out concerning attitudes toward teamwork in academic settings, and grades assigned to students. These data were examined for evidence of free riding and for changes to student attitude concerning teamwork.

4.4.1. Student mutual evaluations

Students answered the following question in their self- and mutual-evaluations following each milestone: "Suppose you have 100 units of something desirable to distribute across your team in proportion to their overall contribution and effort on this project. How would you distribute it?" Questions such as this are not uncommon in evaluating individual effort in student team projects (Fincher et al., 2001). I define a "fair share" as being an equal distribution across all students in the team. Out of 378 total responses to this question for all students in all courses, 82% of these distributions were within 10% of the fair share and 93% were within 20% of the fair share. Students also rated each team member (including themselves) on a 5-point scale from Best (5) to Worst (1) along eight different dimensions, including attendance at meetings, quality of work contributed, communication, meeting deadlines, and contribution to the team effort. In total, students made 2889 individual judgments about themselves and teammates. In 82% of these judgments, the highest rating was awarded, and in 97% of these judgments one of the two highest ratings was awarded. Thus, almost always, students provided judgments under conditions of confidentiality that indicated that they and their groupmates were carrying out a fair share of the teamwork.

4.4.2. Meeting work commitments

Based on an analysis of all task matrices produced by students during these terms, each team member averaged 2.4 tasks committed to per week. Aggregating over all weekly reports of all teams, students completed 94% of the tasks that they committed to on time. Of the remaining 6% of the tasks, over 90% of these were committed within two days of their due date.

4.4.3. Changes in attitude toward teamwork

Students completed a pre-course survey and a post-course survey concerning their attitudes and beliefs concerning groupwork. This survey indicated that students had previously taken an average of 1.9 courses within the department that involved significant teamwork. Students answered the following questions on the pre-course survey related to commitment and free riding on the scale from strongly agree (5) to strongly disagree (1): "In my groups from previous terms, my groupmates lived up to their group commitments" (what I call the *commitment question*) and "My groups from previous terms had one or more students who did not do their fair share of the work" (what I call the *free rider question*). The post-course survey asked identical questions with the words "this term" replacing

"previous terms." There were statistically and substantively significant differences in the mean responses on both of these questions between the pre- and post-surveys. For the commitment question, the pre-survey mean was 3.11 while the post-survey mean was 4.28, $p < 0.001$ using a one-tailed, paired sample t-test. In short, students believed that teammates from the current term met group commitments significantly more often than teammates in the past. For the free rider question, the pre-survey mean was 3.44 and the post-survey mean was 1.64, $p < 0.001$ using a one-tailed, paired sample t-test. That is, free riding was dramatically reduced in the current academic term as compared to teamwork in the past.

4.4.4. Individual sanctions

Individual sanctions for not contributing a fair share to the software project were represented as multipliers of less than 1.0 for project milestones. Of the 114 individual multipliers assigned (three milestones each for 38 students), sanctions were applied on only three occasions.

4.5. Discussion

There is considerable evidence that the rule structures under which students operated led to high levels of commitment and low levels of free riding. It is worthwhile to review those aspects of the rule structures that, according to the collective action literature, give rise to these high levels of commitment toward carrying a fair share of the work.

Face-to-face communication. Regular face-to-face meetings were required and provided an opportunity for updating one another on project progress, for integrating software modules, and for realigning work commitments. Students reported that they frequently supplemented these meetings with a variety of communication media, including instant messaging, cell phones, email, and synchronous online chat.

Repeatability and reciprocity. Requiring weekly reports provided *repeatability*, as did decomposing the final deliverable into three milestones due three weeks apart. As indicated above, sustained cooperation relies upon the development of trust and norms of reciprocity. A team-based project with only a single deliverable at the end of the course may not provide the necessary scaffolding for trust to develop and reciprocity to be practiced. When effort by any team member is contingent on the effort of others, the risk of being played the sucker in a one-shot situation may outweigh the perceived gains from cooperating. Having weekly "deliverables" by each team member provides a succession of "games" that team members play in order to determine whether to continue their effort toward working toward common goals. In addition, weekly meetings and the accompanying report provided a mechanism for individuals to make public promises to undertake work, which the literature above indicates increases the likelihood that these commitments will be carried out.

Monitoring. Weekly meetings provided a natural setting in which low-cost mutual monitoring could occur. Monitoring was strengthened at multiple levels at low cost by coupling the meetings with the weekly report and the requirement to complete a task matrix that documents individual compliance with past commitments. Requiring scribes to post commitments to all teammates within 24 h enabled individual self-monitoring, made subsequent monitoring by teammates low cost, and provided the information necessary for instructor monitoring without exacting the high costs and low accuracy that external, managerial surveillance of student compliance would require. The task matrices also provide the institutional transparency that significantly reduce the need for sanctioning. Multi-level, mutual, and public monitoring emerge as one of the key elements that allow "clan control" prior to the adoption of shared norms and values.

Sanctions. Although sanctions in the form of grade penalties were rarely used, the threat of sanctions and its occasional use, create conditions for quasi-voluntary compliance. Further, it is likely that as students develop trust, norms of reciprocity, and mutual concern, they start replacing quasi-voluntary compliance with norm-based compliance, i.e. they comply because they have internalized cooperative values in working with this group of people.

5. Conclusion

The primary contribution of this paper is in its framing software development teamwork as subject to a set of well-known collective action problems. This paper draws on research from the behavioral laboratory of experimental economics and on field studies within the natural resource governance literature to highlight a set of key theoretical principles for fostering cooperative behavior: face-to-face communication, repeatability of interactions, mutual, open, and multi-level monitoring, and explicit use of sanctions and the threat of sanctions. Taking an institutional analytic perspective provides an explicit focus on the rules—both formal and informal—that people use to constrain their actions so as to increase their cooperation in settings in which such cooperation is contingent on the cooperation of others. This perspective contrasts with theories of managerial control from the management science literature, which, while introducing important concepts concerning behavioral, outcome, self, and clan control, fail to provide tools for subsequent design and analysis of teamwork. These control based theories are inadequate not only for commercial settings but are inappropriate for other software development settings in which hierarchical control is not a key feature, such as open source projects and student software teams in academics.

An implication of this work for research is in the use of the institutional analytic approach for subsequent studies of software teamwork, particularly in a variety of different settings. With additional case studies available, comparisons can be made among these cases for effective rule

configurations and how these might change as a result of differences in the contexts in which teamwork is undertaken. Of particular interest might be ways in which this can contribute to understandings and design of teamwork in settings where team members are geographically distributed, do not share a common national culture, and can never (or rarely) meet face to face. Comparisons can also be made with collective action in non-software domains to determine if there are behavioral characteristics specific to software development that do not arise elsewhere.

There are implications of this approach for practitioners in the design of the rules under which software developers work. These concern applying the theoretical principles associated with eliciting cooperation to the practical work of software development, whether in academic, commercial, or open source settings. The stress on face-to-face communication here echoes (and validates) one of the "Principles behind the Agile Manifesto" (Agile Alliance, 2001), which states "[t]he most efficient and effective method of conveying information to and within a development team is face-to-face conversation." The importance of repeatability suggests that teams that rotate members too quickly might lose the foundation by which reciprocity can develop. The shadow that the future casts can provide developers with increased incentives to cooperate in the present. The same principle suggests breaking teams where recrimination and non-cooperation are also reciprocated, providing these individuals with the opportunity to start anew. The principles of mutual, public, and multi-level monitoring, as well as graduated and multi-level sanctions, can also be applied within software development teams. Although not in contradiction to management theories involving clan control, the research presented here suggests that such controls need not be informal nor based on shared norms or values. Instead, developers can develop their own explicit rules (facilitated, perhaps, by management) that include mechanisms for rule enforcement by developers amongst themselves. It is, in fact, such monitoring and sanctioning that distinguishes rules—and the predictability that they provide through their explicitness—from norms. This can thus relieve management of high transaction costs associated with developer surveillance while preserving developer autonomy and motivation.

Acknowledgments

Thanks to Elinor Ostrom, Vincent Ostrom, Charlotte Hess, and the participants of the Institutional Analysis and Development seminar during Autumn, 2006 at the Workshop in Political Theory and Policy Analysis at Indiana University for their insights about the nature of social dilemmas and the ingenuity that people have used in a variety of settings in crafting institutions to resolve these dilemmas. Thanks to Gwen Arnold, Joohyung Kim, Anna Zachrison, Verlon Stone, and James Nachbaur for discussions on the relationship of social dilemmas to software teams. Particular thanks to James Walker, Sally Fincher, and anonymous reviewers for their extensive comments on earlier drafts of this paper. I am grateful to the University of Washington for providing a sabbatical release during which this research was undertaken, and to the School of Informatics at IU for hosting me during this time. And finally, I am grateful to the students of the software engineering courses I have had the privilege to teach who demonstrate time and again how much can be achieved when people cooperate.

References

Agile Alliance, 2001. Principles behind the Agile Manifesto. http://agilemanifesto.org/principles.html, accessed April 10, 2007.

Axelrod, R., 1984. The Evolution of Cooperation. Basic Books, New York.

Blomquist, W., 1994. Changing rules, changing games: evidence from ground-water systems in Southern California. In: Ostrom, E., Gardner, R., Walker, J. (Eds.), Rules, Games, and Common-Pool Resources. The University of Michigan Press, pp. 283–300.

Boehm, B., 1997. A spiral model of software development. In: Dorfman, M., Boehm, B. (Eds.), Software Engineering. The Institute of Electrical and Electronics Engineers, pp. 415–426.

Cain, B., Coplien, J., Harrison, N., 1996. Social patterns in productive software development organizations. Annals of Software Engineering 2, 259–286.

Campbell, D.T., Stanley, J.C., 1963. Experimental and Quasi-Experimental Designs for Research. Houghton Mifflin, Boston, MA.

Cardenas, J.-C., Stranlund, J., Willis, C., 2000. Local environmental control and institutional crowding-out. World Development 28 (10), 1719–1733.

Casar, M., Plott, C., 2003. Decentralized management of common property resources: experiments with a centuries-old institution. Journal of Economic Behavior and Organization 51, 217–247.

Fincher, S., Petre, M., Clark, M. (Eds.), 2001. Computer Science Project Work: Principles and Pragmatics. Springer, New York.

Hardin, G., 1968. The tragedy of the commons. Science 162, 1243–1248.

Helmke, G., Levitsky, S. (Eds.), 2006. Informal Institutions and Democracy: Lessons from Latin America. The Johns Hopkins University Press, Baltimore, MD.

Henderson, J., Lee, S., 1992. Managing I/S design teams: a control theories perspective. Management Science 38 (6), 757–777.

Kirsch, L., 1997. Portfolios of control modes and IS project management. Information Systems Research 8 (3), 215–239.

Kirsch, L., Sambamurthy, V., Ko, D.-G., Purvis, R., 2002. Controlling information systems development projects: the view from the client. Management Science 48 (4), 484–498.

Levi, M., 1988. Of Rule and Revenue. University of California Press, Berkeley, CA.

Maass, A., Anderson, R., 1986. And the Desert Shall Rejoice: Conflict, Growth, and Justice in Arid Environments. R.E. Krieger, Malabar, FL.

Marwell, G., Ames, R., 1979. Economists free ride, does anyone else: experiments in the provision of public goods IV. Journal of Public Economics 15, 295–310.

Miller, G., 1990. Managerial dilemmas: political leadership in hierarchies. In: Cook, A., Levi, M. (Eds.), The Limits of Rationality. University of Chicago Press, Chicago, IL, pp. 325–357.

North, D., 1990. Institutions, Institutional Change, and Economic Performance. Cambridge University Press, Cambridge.

Ostrom, E., 1990. Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge University Press, Cambridge.

Ostrom, E., 2005. Understanding Institutional Diversity. Princeton University Press, Princeton.

Ostrom, E., Gardner, R., Walker, J., 1994. Rules, Games, and Common-Pool Resources. University of Michigan Press, Ann Arbor, MI.

Ouchi, W., 1979. A conceptual framework for the design of organizational control mechanisms. Management Science 25 (9), 833–848.

Rapoport, A., Chammah, A., 1965. Prisoner's Dilemma: A Study in Conflict and Cooperation. University of Michigan Press, Ann Arbor, MI.

Sefton, M., Shupp, R., Walker, J., in press. The effect of rewards and sanctions in provision of public goods. Economic Inquiry, doi:10.1111/j.1465-7295.2007.00051.x.

Thaler, R., 1991. The Winner's Curse: Paradoxes and Anomalies of Economic Life. Free Press, New York.

Wade, R., 1988. Village Republics: Economic Conditions for Collective Action in South India. ICS Press, San Francisco, CA.

Yin, R., 2003. Case Study Research: Design and Methods, third ed. Sage Publications.