Noname manuscript No.
(will be inserted by the editor)

Cheap Talk, Cooperation, and Trust in Global Software Engineering

An Evolutionary Game Theory Model with Empirical Support

Yi Wang · David Redmiles

Received: date / Accepted: date

Abstract In our studies of global software engineering (GSE) teams, we found that informal, non-work-related conversations are positively associated with trust. Seeking to use novel analytical techniques to more carefully investigate this phenomenon, we described these non-work-related conversations by adapting the economics literature concept of "*cheap talk,*" and studied it using Evolutionary Game Theory (EGT). More specifically, we modified the classic Stag-hunt game and analyzed the dynamics in a fixed population setting (an abstraction of a GSE team). Doing so, we were able to demonstrate how *cheap talk* over the Internet (*e-cheap talk*) was powerful enough to facilitate the emergence of trust and improve the probability of cooperation where the punishment for uncooperative behavior is comparable to the cost of the *cheap talk*. To validate the results of our theoretical approach, we conducted two empirical case studies that analyzed the logged IRC development discussions of Apache LUCENE[1] and CHROMIUM OS[2] using both quantitative and qualitative methods. The results provide general support to the theoretical propositions. We discuss our findings and the theoretical and practical implications to GSE collaborations and research.

Keywords Cheap talk · Trust · Cooperation · Communication · Global Software Engineering (GSE) · Evolutionary Game Theory (EGT) · Apache LUCENE · CHROMIUM OS

Y. Wang
Department of Informatics, University of California, Irvine
Tel.: +1-949-3964587
Fax: +1-949-8244056
E-mail: yiw@ics.uci.edu

D. Redmiles
Department of Informatics, University of California, Irvine

[1] LUCENE: http://lucene.apache.org/
[2] CHROMIUM OS: http://www.chromium.org/chromium-os

If a deer was to be taken, every one saw that, in order to succeed, he must abide faithfully by his post: but if a hare happened to come within the reach of any one of them, it is not to be doubted that he pursued it without scruple, and, having seized his prey, cared very little, if by so doing he caused his companions to miss theirs.
-Jean Jacques Rousseau: *On the Origin of the Inequality of Mankind*, The Second Part, 1754.

1 Introduction

You are chatting with a remote colleague on Facebook Messenger, sharing several photographs of last night's homemade dinner. You are so focused that you fail to notice your manager standing behind you. Your manager becomes angry, yelling: "*Stop your IM! It's wasting work time.*" This is a common reaction for most managers when they see employees using working hours to chat online about something irrelevant to the job. Conventional wisdom claims that such informal, non work-related communications, referred to in some Economics literature as *cheap talk*, is meaningless and a waste of valuable time! Indeed, economists Farrell and Rabin (Farrell and Rabin, 1996) claim that *cheap talk* "*does not generally lead to efficiency.*"

However, our study finds that *cheap talk* in work environments may in fact have surprising benefits. Previously, we conducted intensive field studies of Global Software Engineering (GSE) teams, focusing on their trust of remotely-located collaborators (Al-Ani et al, 2013). The data we collected reveals that those who engage, in their cross-site communication with remote collaborators, in "*cheap talk*," i.e., non-work-related conversational behavior, generally have higher trust (see Section 2). Many interviewees indicated willingness to spend extra effort or sacrifice some work time for cheap talk when interacting with remote collaborators. Trust is important for collaborating with unfamiliar remote GSE collaborators. If an individual decides to cooperate, she actually faces the risk of others' uncooperative behaviors. To overcome the fear of this risk, she needs to be confident in also expecting others to be cooperative. Thus, it is difficult for trust to automatically emerge (Camera et al, 2013; Jarvenpaa et al, 2004). It seems that cheap talk (even with some extra cost) is quite helpful; however, its role in the dynamic trust emergence and development process is unclear.

Proceeding from the result that small cost *cheap talk* over the internet correlates with higher trust, we can hypothesize that it also brings about better cooperation. We seek to investigate this question from a novel cost-benefit perspective (rather than, for instance, a social relationship perspective) in order to precisely and dynamically describe individuals' **strategic** behavior.

First, we use Evolutionary Game Theory (EGT) as the major theoretical tool. Enabling us to analytically study interpersonal cooperation dynamics, it provides insights into, and techniques for, exploring individual strategic behaviors and social group interaction trends (Nowak, 2006b). Leveraging EGT, we can examine how *cheap talk* over the Internet influences interactions between GSE practitioners and potentially gives rise to the emergence of trust, as well as, in the long run, cooperation. Moreover, EGT can describe team level dynamics by predicting the proportions of members using different strategies.

The Stag-hunt game, which was introduced by Rousseau (see the prologue), is a natural description of mutual cooperation (Skyrms, 2008), and can be used as an abstract description of GSE collaborations. We modify its classic form and add a new strategy[3] ("*cheap talk-cooperate*[4]") to associate cheap talk over the internet with concrete cooperation. With the modified game, we describe the dyadic interactions among team members, enabling players to update their strategies through social learning.

Using this approach, we can investigate the following research questions (RQs):

RQ1 : Can *cheap talk-cooperate* self-reinforce if it is secured by situation-intrinsic incentives?

RQ2 : Can trust emerge and eventually become a cooperation-ensuring team convention if defectors' punishment compares to the cost of *cheap talk*?

RQ3 : What are the long-term dynamics (including frequency) of using *cheap talk*, particularly as cooperation and trust are established over time?

Then, to validate the results derived from our theoretical model, we performed two empirical case studies on Apache LUCENE and CHROMIUM-OS by mining their logged IRC discussions. Our analyses provided general support to the theoretical results and predictions. For example, we observed consistency between *cheap talk* and cooperation development, their precedence relationship, and *cheap talk's* disappearance in the long run. From a methodological perspective, this study demonstrates the feasibility of a novel approach, which integrates theoretical game theory analysis with empirical study to develop relevant and generalizable GSE knowledge.

The remainder of this paper is organized as follows. Section 2 briefly introduces the prior empirical study motivating the present work. Section 3 introduces the Stag-hunt game, and related GSE research on *cheap talk* and trust. Incorporating the results presented in (Wang and Redmiles, 2013), Section 4 elaborates on the evolutionary game theory model, and Section 5 presents the theoretical analysis and results, respectively. Two empirical case studies into the development IRC discussions of Apache LUCENE and CHROMIUM-OS are presented in Sections 6 and 7 to validate the model and its analytical results. Finally, Sections 8 and 9 respectively discuss related issues and conclusions.

2 A Motivating Empirical Example

While investigating trust in distributed teams, e.g., (Al-Ani et al, 2013), we found rich evidence of user's favorable attitude toward and frequent use of

[3] In game theory, a strategy refers to a pre-defined action that a player may use in interacting with other players.

[4] We will use *C-C* to denote ***Cheap talk-Cooperate*** in the reminder of this paper.

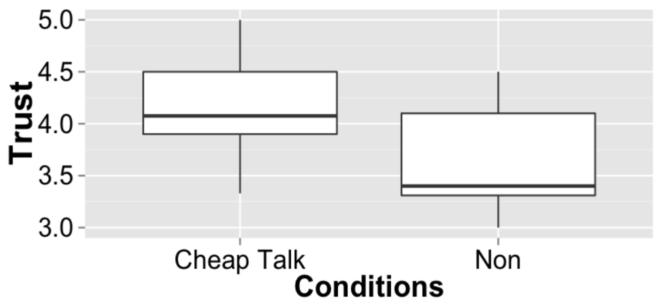


Fig. 1 Collaborators engaging in *cheap talk* express higher trust levels compared to those who do not. The horizontal bold line in each box represents the median value.

cheap talk during interactions with remote colleagues. Moreover, we noticed an association between *cheap talk* and trust. When we interviewed software developers in distributed teams, some of them reported that non-work-related *cheap talk* helped build trust. According to the interviewees' narratives, we found that *cheap talk* had potential, for instance, for building common ground, developing close interpersonal relationships, and dealing with cultural differences. One interviewee emphasized *cheap talk's* trust-building role as follows:

> *Yes, I think it's critical* [*for building trust*]. *We do try to bring people over at key planning junctures or transfers of technology. So some of the Poland engineers were here in early Q1 for cross training. A couple of them are here this week. I took them out for beer* [*and*] *laughs last week. One thing we do is we have a quarterly, we call it off site, where we just go out and do a social activity for team building. So that happens every quarter. When we have visitors we try to do something social in addition to the planning.*

In a GSE setting, as opposed to the face-to-face context, it is difficult to maintain "real world" *cheap talk* at a satisfying level. However, collaboration tools allow for the adaptation of *cheap talk* over the Internet. The interviewees mentioned that they spent extra effort and time on non-work-related activities via various tools, such as casual talk over IM, sharing personal pictures, and discussing hot topics. For example, one interviewee reported a preference for instant messaging, and another described the type of *cheap talk* afforded through remote online communication, and how it can be used in collaboration.

> Example 1. *...but I like instant messaging. Mainly because with instant messaging you can put in little characters like a smiley face or a wink or something like that. I like that.*
>
> Example 2. *Well, first time I am talking with someone, well, "Where are you calling from? Oh." Because maybe we are in different time zones, at home or whatever. "What's the weather like?" I mean, "Oh, is it hard to have to work at home in the evenings?" to maybe build up some*

sympathy. "Is it hard for you because you have to call in, in the evening to talk with us?"

However, some interviewees did not express any interest in *cheap talk* with remote colleagues, and in fact considered *cheap talk* valueless. Based on their opinions on *cheap talk*, the interviewees were coded into two categories (*non-Cheap talk* and *Cheap talk*). We also computed each individual's average trust score towards their collaborators. During the interview, each interviewee was asked to locate her collaborators on a trust spectrum, which was then coded into a 5-level interval scale to produce her aggregated trust score. In total, 43 interviews were collected. However, 2 of these were incomplete and did not include a trust score. Thus, in the end, we coded and analyzed 41 of these, classifying 9 as *non-Cheap talk* and 32 as *Cheap talk*.

To identify whether there was a difference between these two categories, we performed a simple *independent sample t-test*. The results showed significant difference between these two groups' trust: *P-value: 0.013*, *Effect Size: 0.921 (Cohen's d)*, and hence significance at the 0.05 level, while the *effect size* indicates the sample size is sufficient. The interviewees in the *Cheap talk* category exhibited higher trust (mean: 4.152) than those in the *non-Cheap talk* category (mean: 3.607). Fig. 1 illustrates the results.

3 Background

3.1 Prior Empirical Research on Cheap Talk in Software Engineering

Building on the literature in non-work social workplace interactions, we first introduced the formal concept of *cheap talk*[5] in software engineering in a previous paper (Wang and Redmiles, 2013). Social interactions in workplaces are usually conceptualized in terms such as "socialization" (Dittrich and Giuffrida, 2011), "small talk" (Cassell and Bickmore, 2003; Steed et al, 2003), and "water cooler" (Herbsleb et al, 2002), and their importance has been noted in (Ducheneaut, 2005). Dittrich and Giuffrida (2011) identified socialization as one of four usage dimensions of Instant Messaging in their qualitative study of a Danish/Indian global software team. They found that a few of the *Skype* chats were purely social, such as everyday chats around the water cooler in co-located settings, and argued that informal communications provide a channel for building trust and social relationships. However, they neither explained why the social chats helped to build trust, nor did they assess the measurable influence on the cooperation. Similar results were also reported by Cramton and Hinds (2007), who argued that casual interactions smoothed the cooperation among individuals from three different cultures.

Guzzi et al (2013) reported on a qualitative study of the LUCENE mailing list focusing on the communication patterns of open source developers.

[5] Cheap talk in Economics usually assumes a zero cost; however, in our usage , we allow cheap talk to associate with non-zero, yet minimal cost.

Three of their four "social interaction" categories were work-related and only one ("social norm": 3 in 506) had nothing to do with work. The other three ("acknowledgement of effort," "coordination," and "new contributors") were at least partially related to work, as their labels suggest. In general, software engineering researchers are beginning to pay more attention to how personal and affective factors influence collaboration in the development process. Some preliminary work by Calefato et al (2013) demonstrates that informal information from social media can augment social awareness and improve trust.

3.2 Moving Toward A Novel Perspective and New Approach

Prior studies reviewed above emphasize the role of a "*social relationship*" in informal interactions (see also (Mislin et al, 2011)). However, the concept of social relationship cannot fully explain an individual's strategic choice, nor how a communication strategy evolves into a team level norm. Adapted from the Economics literature, *cheap talk* affords us a unique cost-benefit perspective on studying non-work-related interactions among developers, and enables us to use a variety of analytical techniques. Whereas economists develop Evolutionary Game Theory (EGT) models to generatively explain social behaviors, software engineering researchers are accustomed to using empirical field studies to develop practice-oriented knowledge. With the goal of developing rigorous, elegant theoretical explanations and predictions (which EGT does well), while maintaining the validity and relevance of theoretical results (which empirical field studies can accomplish), our study uniquely combines these two approaches.

By leveraging EGT techniques, we can dynamically investigate the strategic interactions among a fixed number of individuals (an abstraction of a "*team*"), as well as derive analytical results and predictions for describing long-term individual strategy changes and team dynamics. Descriptive studies focusing on social aspects are more static and often fail to characterize team level dynamics. Furthermore, with simple assumptions (the Stag-hunt game and strategic behavior change rule), EGT enables us to deductively establish propositions to answer "how" *cheap talk* influences trust and cooperation. We can then use empirical study to validate these generatively-developed propositions, which forms the integrated process of scientific knowledge development that ensures both rigorousness and relevance (Montuschi, 2003).

3.3 Game Theory and Group Dynamics

Game theory provides a framework to analytically model and investigate decision-maker strategic interaction situations, in which each participant's utility for the outcome depends on both individual and overall team decisions and relative positions (Easley and Kleinberg, 2010). Game theory has been successfully applied in many research areas to model people's strategic decision-making and the long term dynamics of their interactions. Game theory has been successfully applied in various research areas (Bowles and Gintis, 2011), including Political Science, Management, Sociology, Computer Science, Network Science, and even Professional Sports management (e.g., Chiappori et al (2002) on penalty kicking in French and Italian professional soccer leagues).

We can apply game-theoretic analysis to settings in which individuals can exhibit different forms of behavior. Doing so enables us to identify which forms of behavior have the ability to become stable norms in the group after long-term interaction; this type of analysis is evolutionary game theory (EGT). The key insight of EGT is that different types of behavior may bring different benefits to the individuals in a group, making some individuals become more successful. Hence, these successful individuals have some evolutionary advantages, i.e., in the long run they consistently receive better results. Other individuals may also learn these successful behaviors. Eventually, some behaviors become a group social norm.

Organizations are coordinated action systems comprising individuals and groups with differing preferences, information, interest, and knowledge (March and Simon, 1993). In essence, an organization is (1) a multi-agent system, with (2) identifiable boundaries and structures, as well as (3) system-level goals, toward which (4) the constituent agents are expected to individually or collectively contribute by interacting with others (Puranam et al, 2014). Game theory social network models reflect these organizational essences, only requiring a few assumptions. Researchers can easily validate its conditions and evaluations in experimental and natural settings (Gintis, 2000). For example, Banerjee et al (2013) modeled the behaviors and relationships among individuals in a microfinance network, and empirically validated it with data from a rural Indian village. The results directly modeled the characteristics of a strong microfinance network, and, moreover, provided insight into how one might design a sustainable one. At the individual level, game theory can help determine optimal interpersonal interaction strategies. Jackson et al (2012) employed similar techniques to investigate the robustness of the informal exchange of favors among a society's members.

3.4 Stag-Hunt Game

The classic Stag-hunt game is a non-zero-sum, 2 by 2 game in which each player has two strategies: *cooperate* or *defect* (see Fig. 2). In ancient times, two men hunt for food. If both use *defect*, they would hunt individually, and each would get a hare. If both cooperate, they could kill a stag, and each would receive one half stag. If one cooperates while the other defects, the "*cooperator*" would receive nothing and the "*defector*" would receive a hare. Formally, the Stag-hunt game's payoff structure can be represented by the first matrix in Fig. 3 if $R > T > P > S$. When a player uses C in interaction, she receives R ($Cell_{11}$ if the other player uses C; and S ($Cell_{12}$) if the other player uses D. When she decides to play D, she receives T ($Cell_{21}$) if the other player uses C; and P ($Cell_{22}$) if the other player uses D.

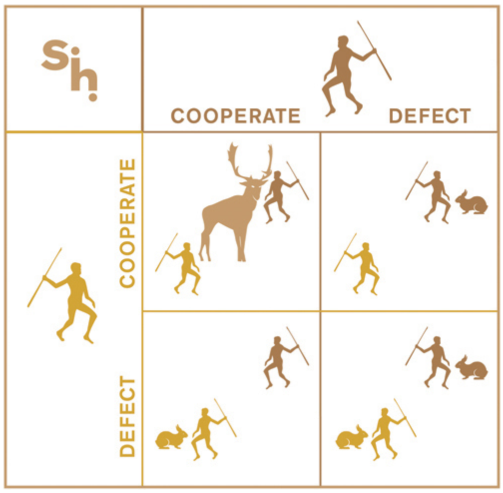


Fig. 2 A visual illustration of the Stag-hunt game, Licence: CC BY-SA3.0, Credits to C.Jensen & G. Riestenberg Jensen and Riestenberg (2012).

Even in the Economics literature, the Stag-hunt game is generally less applied than prisoner's dilemma in investigating the evolution of human cooperation. However, it is a better and more natural representation of real world cooperation (Skyrms, 2001). The most significant advantage of the Stag-hunt game is that it allows "*cooperation*" to be achieved by rational individuals while an individual's only choice in prisoner's dilemma is *defect*. In Stag-hunt, (*cooperate*; *cooperate*) is a Nash equilibrium [6]. Because once both parties agree to *cooperate*, none has the intention to defect. But (*cooperate*; *cooperate*) is not the only equilibrium. When players don't trust each other, neither wants to cooperate and they would keep using *defect*. *cooperate* is not strictly dominated by *defect* in this game. The state of (*cooperate*, *cooperate*) is a payoff-dominated equilibrium (everyone gets the best payoff), while (*defect*, *defect*) is a risk-dominated equilibrium (everyone minimizes potential loss). Which one can be achieved is determined by players' belief in their opponents: namely, their trust (Skyrms, 2001).

3.5 Software Development as Stag-Hunt

The Stag-hunt game is a natural metaphor of dyadic (one-to-one) collaborations in software engineering activities. In this paper's scope, the cooperation is restricted to dyadic interactions to simplify the analysis and discussion. Multi-individual cooperation can also be viewed as a series of dyadic interactions (Skyrms, 2014). In many cases, developers do not necessarily need to cooperate with others to get their jobs done ("receive a *Hare* as payoff"), even

[6] Nash equilibrium describes a state that no one can achieve a better payoff by working independently.

$$\begin{array}{c|cc} & C & D \\ \hline C & R & S \\ D & T & P \end{array}$$

a. Stag hunt game (R>T>P>S)

$$\begin{array}{c|cc} & C & D \\ \hline C & 2 & 0.5 \\ D & 1.5 & 1 \end{array}$$

b. A numerical example

Fig. 3 Payoff structure of a general Stag-hunt game along with a numerical example.

when their work items are highly interdependent. However, less cooperation may influence their work's quality. Cataldo et al (2008) pointed out that the communications among developers can significantly influence the quality of a software system, even though work items can be independently finished. Thus, collaboration can produce higher quality work ("receive a half *Stag* as payoff"). In some cases, a software engineer may believe that her colleague will cooperate, but things do not go as she expects. Thus, she may experience some "unfavorable" results (e.g., fail to deliver a commitment on time) due to the other's defection, while the other is still able to achieve the utility of individual action ("receive a *Hare* as payoff"). Hence, dyadic collaboration in software development can be analogous to Stag-hunt, allowing us to use standard EGT techniques to investigate software development collaborations.

3.6 From Individual to Team

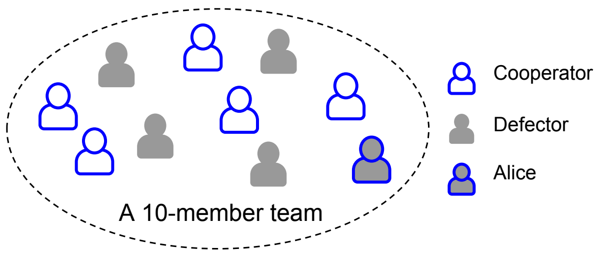


Fig. 4 A 10-member team where 5 members are cooperators, 4 members are defectors, and Alice hasn't decided.

We have shown that the collaboration between two individual software developers can be abstracted to the Stag-hunt game. Now, we generalize it to the team setting. Fig. 4 shows a configuration of a 10-member team in which 5 members are cooperators, 4 members are defectors, and 1 member (Alice) is undecided. Let's assume Alice is going to decide her strategy in this setting.

The probability for her to meet a cooperator is 5 in 9, whereas the probability of meeting a defector is 4 in 9. So, her expected payoffs are as follows:

Being a cooperator: $R \times \frac{5}{9} + S \times \frac{4}{9} = 2 \times \frac{5}{9} + 0.5 \times \frac{4}{9} = 1.33.$
Being a defector: $T \times \frac{5}{9} + P \times \frac{4}{9} = 1.5 \times \frac{5}{9} + 1 \times \frac{4}{9} = 1.27.$

Obviously, being a cooperator is more attractive for Alice, and she might decide to cooperate in the subsequent interactions. The defectors may also learn that being a cooperator may be better for them and, eventually, all become cooperative. This is an ideal case. Sometimes, the majority of team members are defectors, in which case defecting would be more attractive.

Therefore, the proportion of cooperators in a team characterizes the collaboration in a team. The change of the proportion of cooperators can be expressed by EGT. Hence, we can analytically identify whether or not a team has the potential to reach a set of desirable states in which most individual members become cooperators. Core members in a software project team are often stable, or exhibit only small changes (Caglayan et al, 2013). Even when the turnover is high, the team size is not likely to drastically increase or decrease. Therefore, we need to investigate human interaction in software development teams under the assumption of a fixed population[7]. In a fixed population game with two strategies $\{A,\ B\}$, an increase of individuals using strategy A will definitely lead to a decrease of individuals using strategy B. No strategy switch is independent. Therefore, evolutionary dynamics in a team of finite size requires a stochastic theory (Ficici and Pollack, 2000).

3.7 Trust Is Crucial!

To study the long-term interaction, fixed population game, Nowak et al. developed new techniques using stochastic process methods (Fudenberg and Imhof, 2006; Nowak et al, 2004) (see Section 5), yet the results remained similar; both "all-cooperate" and "all-defect" were quite stable. For the numerical example, the time spent on both homogeneous states is almost the same in the long run, while the probability of switches between them are fairly small (Young, 1998). What would make people willing to expose themselves to the risk of others' possible defection? The answer is very straightforward: their **trust** of others. With trust, they can expect their colleagues will also behave cooperatively when they are cooperative (Al-Ani et al, 2013).

4 New Stag-Hunt Game with Cheap talk over the Internet

Given the discussion in the previous section, trust is crucial for the adoption of a *cooperate* strategy. But, the classic Stag-hunt game does not guarantee the

[7] The fixed population assumption does not exclude the change of a team's size and its members. The term "fixed" only means the population has no potential to be infinite. In fact, small changes do not influence the analytical results, see Nowak (2006a).

emergence of trust. The motivating example in Section 2 suggests that cheap talk over the internet may help develop trust, which inspires us to extend the classic Stag-hunt game to analytically explore the role of *cheap talk* in trust and cooperation development. In this section, we develop a new game that allows *cheap talk* to associate with *cooperate* to form a new interpersonal interaction strategy.

4.1 Talk is Still Cheap, but NOT Free

A typical way to model *cheap talk* is to treat it as a cost free *signal* with no predefined meaning. Imagine that you talk about your dog with your officemates during lunch; you pay nothing for this type of conversation. Furthermore, although irrelevant to your work, it does convey some kind of signal. These signals gain meaning through the interaction progress, and thereby increase the possibility and frequency of collaboration (Santos et al, 2011). However, in GSE, *cheap talk* often occurs over the Internet via various collaboration tools, such as IM. *Cheap talk* requires individuals to expend extra time and effort; for example, one interviewee said he once intentionally stayed late in his office to have some cheap talk with a new remote colleague over IM. Thus, while not free, this type of interaction is *cheap* when compared to the benefits of cooperation. In this sense, *signal* is no longer a good abstraction; *cheap talk* is actually a concrete action rather than a cost-free signal, although it is still cheap. We denote this type of *cheap talk* as "*e-cheap talk.*"

4.2 The Stag-hunt Game with Cheap Talk

To capture the essence of *e-cheap talk*, we make a slight change to the classic Stag-hunt game. The new game has three strategies instead of two. To set up an *e-cheap talk* over the Internet, the proposer must pay a small management cost; e.g., two LUCENE developers spent 20 minutes talking about the political systems and their differences in the U.S. and Germany (see Section 6.3.1). We reasonably assume that the cost is small compared to the benefits from the cooperation ($e << R$). It may not be constant, but it is likely that the fluctuations do not span different orders of magnitude. Furthermore, to simplify the discussion, we suppose that players' preferences remain consistent (i.e., there is no execution noise). If a player decided to start *e-cheap talk*, she would use the *cooperate* strategy in the following interaction. So, we add a new strategy "Cheap Talk-Cooperate" called C-C and make the new game contain three strategies $\{C\text{-}C, C, D\}$.

If two players chose C-C in their interaction, they would share the cost of *e-cheap talk*, and thereby each could receive R-$e/2$ payoff. Note that, even in cases in which only one pays the cost, the **average** payoff of playing (C-C, C-C) is still R-$e/2$ in the long run. If one plays C-C while the other plays C, the first will pay the cost all by herself. Her payoff is (R-e, R). If a C-C player

$$\begin{array}{c|ccc} & C\text{-}C & C & D \\ \hline C\text{-}C & R\text{-}e/2 & R\text{-}e & S\text{-}e\text{+}g \\ C & R & R & S \\ D & T\text{-}g & T & P \end{array}$$

a. New stag hunt game with cheap talk.

$$\begin{array}{c|ccc} & C\text{-}C & C & D \\ \hline C\text{-}C & 1.9 & 1.8 & 1.3 \\ C & 2 & 2 & 0.5 \\ D & 0.5 & 1.5 & 1 \end{array}$$

b. A numerical example, *e=0.2, g=1.*

Fig. 5 The payoff structure of the Stag-hunt game with *e-cheap talk* and a numerical example ($e = 0.2$, and $g = 1$). The differences between new game and classic stag hunt are marked with red.

meets another D player, the second player may be punished for refusing to cooperate after the first player showed some signal of cooperation[8]. We assume that the punishment equals g. Meanwhile, the C-C player (let's call her "*cheap talker*" for convenience) receives a compensation. To simplify the calculation, we assume the compensation is equivalent to the punishment. Thus, the payoff of this interaction is *(S+g-e, T-g)*. Retaining this part of the payoff structure as it is in the classic form,[9] Although assuming the punishment equals the compensation is a common technique (Pereira et al, 2012), there may be cost or even potential extra payoff associated with the punishment and compensation process. We provide further discussions in APPENDIX A. Fig. 5 illustrates a modified Stag-hunt game with *e-cheap talk* and a numerical example. For this numerical example, "all-cooperate" is the only evolutionarily stable state in an infinite population setting. However, how "all-cooperate" is achieved with *e-cheap talk* is still unclear, and the result does not fit the fixed population setting.

5 Theoretical Analysis & Results

To simulate the team setting, the new Stag-hunt game will be played by a fixed number (N) of players. Using the analysis technique to be introduced in Section 5.1, we reveal the long-term dynamics of cooperation with a 100-member team as an example, and conditions to enable long-term cooperation (see Section 5.2). The findings provide answers to the research questions highlighted in the introduction. First of all, let's begin with a real world example.

[8] Punishment may take many forms, e.g., reputation loss.

[9] It is reasonable to assume that, in the classic game, the player who plays *defect* would not be punished, since there is no reason for her to give up a risk-dominated strategy without any hint of her opponent's action.

5.1 A Real World Example

Consider a 10-member team, in which 7 players are defectors, while the rest are cooperators. In classic Stag-hunt game, the team will inevitably reach a state in which all its members defect. However, with the new strategy, a member named Alice may consider switching from *defect* to *cheap talk-cooperate*. In this case, she may not have enough confidence to trust others, but is willing to try. For her, the expected payoffs are:

Being a cooperator: $R \times \frac{3}{9} + S \times \frac{6}{9} = 2 \times \frac{3}{9} + 0.5 \times \frac{6}{9} = 1$.
Being a defector: $T \times \frac{3}{9} + P \times \frac{6}{9} = 1.5 \times \frac{3}{9} + 1 \times \frac{6}{9} = 1.17$.
Being a cheap talker: $(R-e) \times \frac{3}{9} + (S-e+g) \times \frac{6}{9} = 1.9 \times \frac{3}{9} + 1.3 \times \frac{6}{9} = 1.5$.

Obviously, being a cheap talker is quite attractive! Hence, her strategy switches to *C-C*. Her *success* may be imitated by her peers, motivating them to also switch to *C-C*. When cheap talkers become the majority, the advantage of being a cooperator emerges, for cooperators do not have to pay the small cost (e) of cheap talk over the internet. Therefore, it becomes possible for all members to become cooperators.

5.2 Analysis Method

Now, we formalize the process described in the above example. We need a formal analytical method in order to provide general solutions for describing the process and studying its characteristics. Moreover, when a team is large, the state space could be very large; the total state for a N-member team is: $\frac{N+2}{2} * (N+1)$. For instance, there are 5151 total possible states for a 100-member team. Therefore, we may not want to examine them one by one. Moreover, to reflect a player's *bounded rationality* (Simon, 1991), it also allows players to make a mistake with a small probability.

First, the above example can be described as a Markov process. At period *t*, Alice has a chance to review her strategy. Her decision is based on the strategy profile (the number of people using a specific strategy) of the current period. In above example, the strategy profile is: [*C: 3, D: 7*]. After Alice changes her strategy, the strategy profile becomes: [*C-C: 1 (Alice), C: 3, D: 6*]. And the process arrives at period *t+1*. Using the strategy profile as the state of period *t*, we can find the state of period *t+1* is determined by the state of period *t*. In this example, the change of strategy profile can be viewed as a death of a "defector" and a birth of a "cheap talker." Therefore, the Markov process is a simple birth and death chain.

Since we now have a way to describe the whole process and its states, the next problem is to specify how individuals make decisions about switching strategies. In the above example, Alice considers which strategy brings the best expected payoff, then simply picks that strategy. In EGT, the individuals' payoff represents their *social or economic success* (Nowak et al, 2010).

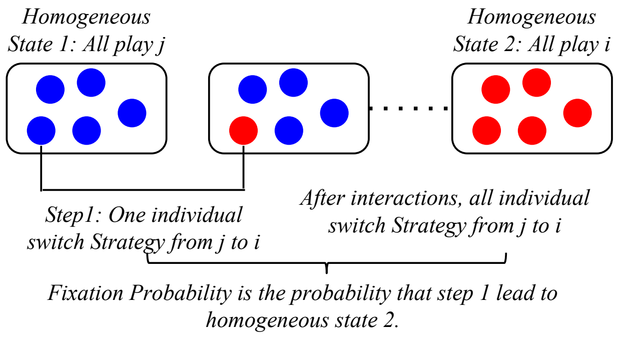

Homogeneous State 1: All play j
Homogeneous State 2: All play i
Step1: One individual switch Strategy from j to i
After interactions, all individual switch Strategy from j to i
Fixation Probability is the probability that step 1 lead to homogeneous state 2.

Fig. 6 Strategy switch process and fixation probability.

In more general cases, the strategy change is followed by some learning dynamics, for example, reinforcement learning (Erev and Roth, 1998), probe and error (Skyrms, 2010), etc. In our discussion, we assume social learning is deterministic. Once a selected player finds a strategy brings a better payoff, she immediately switches to the new one; as Alice does in the above example.

Now we can study the long-term dynamics of the process without considering each period's state. The analysis method is based on EGT techniques for finite populations, e.g., (Nowak, 2006a; Nowak et al, 2004) and stochastic process analysis techniques, e.g., (Fudenberg and Imhof, 2006; Traulsen et al, 2006; Fudenberg and Imhof, 2008). We follow the method in Nowak (2006a) to assume that in any period there are at most two coexisting strategies (for convenience: i and j) for calculating which one is more advantageous in the long run.[10] Suppose one player decides to try i in a state in which the whole population takes strategy j, this would eventually lead to two situations: first, this single i-player may bring the whole population to play i; second, the i-player may return to play j. The probability of the first situation is called a fixation probability: ρ_{ji}, while the probability of the second situation is 1-ρ_{ji} (see Fig. 6 for a more intuitive description).

If $\rho_{ji} = 1/N$, i has no evolutionary advantage (i.e., better expected payoff in the long run) over j, for $1/N$ represents pure neutral selection (Fudenberg and Imhof, 2006). According to our prior discussion, the switches from all-j and all-i result in a birth and death Markov chain. With these assumptions, we can write the fixation probability ρ_{ji} in the following form if $j \neq i$:

$$\rho_{ji} = \frac{1}{1 + \sum_{k=1}^{N-1} \prod_{l=1}^{k} \frac{g(l)}{f(l)}} \tag{1}$$

This equation is directly adapted from (Fudenberg and Imhof, 2006). Here, $g(l)$ and $f(l)$ refer to the expected payoff of playing j or i when there are l

[10] Assuming there are at most two co-existing strategies simplifies the calculation. With this technique, we only need to perform a small number of pairwise comparisons between any two strategies. Then, we can build a whole picture of transitions among states.

individuals playing strategy i. Using π_{ij} to denote payoff in an i-j interaction, they are given by (2) and (3). In fact, we already used it in the example in Section 5.1 when calculating Alice's expected payoff.

$$g(l) = \frac{\pi_{ji}l + \pi_{jj}(N-l-1)}{N-1} \tag{2}$$

$$f(l) = \frac{\pi_{ii}(l-1) + \pi_{ij}(N-l)}{N-1} \tag{3}$$

After calculating all ρ_{ij}, and assuming that the mutation limit μ is positive, we can form an irreducible Markov process to describe the transition between strategies. We can calculate the long-term stationary distribution[11] for all three homogeneous states ("all-cooperate," "all-defect," and "all-C-C")[12] through 6 rounds of pairwise comparison.

5.3 Analysis Results

5.3.1 Dynamic Long-Term Analysis

In this section, we present the analytical results of the specific numerical example in Fig. 5 by using the method introduced in Section 5.1. We assume the team size is 100, since most software development teams have less than 100 members according to the ISBSG (release 7) industrial dataset (Pendharkar and Rodger, 2009). Using equation 1, we can calculate the fixation probability ρ_{ji} for any two strategies. According to Nowak et al (2004), we set the mutation rate $\mu \to 1$ to ensure the mutation does not interfere the deliberated strategic change. Fig. 7 describes the stationary distribution and fixation probabilities (those are stronger than neutral, i.e., $\rho_{ji} > 1/N$) of the three homogeneous states. In figure 7, each vertex represents a homogeneous state, and the percentage shows that the probability of the whole process will end in this state in the long run. The arrows between the three homogeneous states indicates the strength of pairwise transitions (calculated using equation 1, 2, and 3) stronger than neutral (1/100). In this example, the result is very sharp: "all-defect" and "all-C-C" would disappear in the long run, which indicates that almost all individuals eventually learn cooperation and build trust. In this example, the punishment (g in Fig. 5) is large enough so that the "*defect*" strategy is strictly dominated by "*C-C*," which leads to the disappearance of "all-defect." Besides, *cooperate* also strictly dominates *C-C*. However, if the punishment (incentive) was small enough, e.g., $S - e + g < P$, we could expect the "all-defect" might survive in the long run.

The strengths of the transitions differ, which helps us to identify precisely how a transition occurs. As figure 7 indicates, the transition from "all-defect"

[11] For an irreducible Markov process, the stationary distribution is the unique eigenvectors with eigenvalue 1 (Karlin, 2014).

[12] They are homogeneous states because all players take the same strategy in each of these three state.

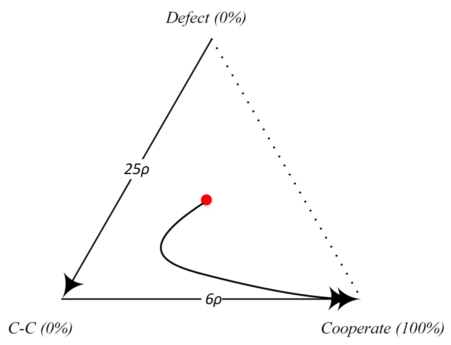


Fig. 7 The simplex describes the fixation probability between three homogeneous states and their stationary distribution in the long run ($N = 100, \rho = 1/N,\ \mu \to 1$). Only transitions stronger than ρ are shown. A possible route for a non-homogeneous state (represented by the red point in the middle) to reach the stable state is depicted.

to "all-C-C" is rather strong, while the transition from "all-C-C" to "all-cooperate" is not so strong. The other transitions are relatively weak, for example, the transition probability from "all-cooperate" to "all-C-C" is only $2.62{\times}10^{-5}$. This indicates the important *catalytic effect* of *C-C* in moving individuals from *defect* to *cooperate*, although it would disappear in the long run. For a team, members may first become cheap talkers, and then convert to cooperators. It is almost impossible for the majority of its members to directly switch from *defect* to *cooperate*.

In the real world, it is almost impossible that all members are defectors when a project starts. The team members may be well-mixed. For hybrid states consisting of populations with different strategies, there is a similar way to reach a final "all-cooperate" state. At the individual level, a defector will first become a cheap talker, and then become a cooperator. A hybrid state (indicated by the red point in the central of the simplex) may follow the red curve in Fig. 7. Some defectors become cheap talkers, while in parallel some cheap talkers change to cooperators. However, in the long run, all hybrid states will disappear.

In our analysis, *C-C* eventually disappears, but it plays a central role in bringing a team from the unfavorable state "all-defect," to the more satisfying state "all-cooperate." Without it, a team is likely to be trapped in the "all-defect" state, because the direct transition from "all-defect" to "all-cooperate" is impossible (no direct line in fig. 7). A team could achieve "all-cooperate" fairly fast if learning is quick. In our case, we assume learning is deterministic and sensitive to any positive payoff improvement. Under these assumptions, the total number of periods should be in $\mathcal{O}(N)$ level for an N-member team, depending on the initial states. Given that most software development teams have less than 100 members (Pendharkar and Rodger, 2009), it is reasonable to assume the homogeneous "all-cooperate" state can be reached in a relatively short period of time.

In some cases, if people are less sensitive to payoff change, sticking to *C-C* is also an acceptable and possible alternative. When compared to the cost of setting up or managing the *e-cheap talk*, cheap talkers can avoid being exploited by the defectors, while maintaining approximately equally good payoff against the cooperators for $\epsilon << R$.

5.3.2 Conditions for Long-term Cooperation

Independent of the above numerical example, in general, what conditions ensure the transition from *defect* to *C-C*, and from *C-C* to *cooperate*? If these conditions can be identified, they could be applied to high-level organization or computer-supported collaborative systems design to promote cooperation in a straightforward way. To answer this question, we directly use an important analytical measurement to identify pairwise evolutionary advantages in the long run. The measurement was developed in Nowak (2006a) by guaranteeing *equation* $1 > 1/N$ (one strategy has more evolutionary advantage than the other). For the two strategies i and j, if the following condition holds, the transition from j to i is more probable:

$$(N-2)\pi_{ii} + (2N-1)\pi_{ij} > (N+1)\pi_{ji} + (2N-4)\pi_{jj} \tag{4}$$

If N is large enough ($N \pm 2 \approx N$), the above condition can be reduced to:

$$\pi_{ii} + 2\pi_{ij} > \pi_{ji} + 2\pi_{jj} \tag{5}$$

The transition from *C-C* to *cooperate* is guaranteed to be more probable if:

$$R + 2R > R - e + 2(R - \frac{e}{2}) \ \Rightarrow -2e < 0 \tag{6}$$

which automatically holds when $e > 0$. Obviously, this transition is guaranteed by the payoff structure of the model. The transition from *defect* to *C-C* is guaranteed to be more probable if:

$$R - \frac{e}{2} + 2(S + g - e) > T - g + 2P \ \Rightarrow \tag{7}$$

$$g > \frac{T + 2P - R - 2S}{3} + \frac{5e}{6} \tag{8}$$

So far, we have presented numerical and analytical results that indicate that *e-cheap talk* sufficiently improves the probability of cooperation, so long as the punishment to *D-player* is comparable to her opponent's cost of using *C-C*, and the transition from *C-C* to *cooperate* is almost destined. Moreover, this result is independent of team size, for "N" has been eliminated in the step from equation (4) to (5). Since all team members adopt *cooperate* as their strategy, it is reasonable to assume they build the expectation that other team members would also cooperate. This expectation exactly comprises the trust among team members(Al-Ani et al, 2013). Therefore, the development of the expectation reflects the emergence of trust.

5.4 Analytical Answers to Research Questions

The main analytical findings are summarized in three propositions:

- Proposition 1. *e-cheap talk is sufficient to promote cooperation if punishment to a defector is comparable to the opponent's cost of using C-C.*
- Proposition 2. *The improvement of cooperation is explicit (more players become cooperators). Trust (expecting others to cooperate) is implicitly developed, and further ensures cooperation.*
- Proposition 3. *e-cheap talk decreases, or even disappears, once trust and cooperation are fully developed. Hence, it functions as a catalyst in this process.*

These three propositions provide answers to the research questions in the introduction. Proposition 1 provides confirmation to RQ1. Proposition 2 conceptually answers RQ2 by pointing out trust implicitly develops, while explicit cooperation becomes a team social norm. Proposition 3 answers the long-term dynamics of *e-cheap talk* (RQ3). They can be empirically validated. We can expect the following empirical observations: (1) there is strong correlation between *e-cheap talk* and cooperation. If a pair of individuals use *cheap talk*, they are likely to also cooperate on work-related issues. (2) *e-cheap talk* should occur between individuals prior to work-related talk. (3) *e-cheap talk* will gradually decrease with cooperation's progress.

For a real world GSE team, the theoretical model provides a possible explanation for how trust develops within the team using *e-cheap talk*. At the beginning of GSE cooperation, team members have less confidence about whether or not their remote colleagues will be cooperative. Therefore, they may prefer to use *e-cheap talk* to exhibit their willingness to cooperate. The collaboration then establishes whether remote colleagues respond cooperatively (either using *C-C* or *cooperate*). The individuals who initiate *e-cheap talk* would not experience significant loss, even if the others decide not to cooperate, for the defectors' punishment ensures the cost of *e-cheap talk* will be compensated. However, once cooperation is established, people may feel they can directly devote all their effort toward collaborating with remote colleagues without using *e-cheap talk* as a costly "probe." Although others may still defect, they are willing to take this risk, and they begin to trust that others will not defect. In GSE practices, we can expect that team members will use less *e-cheap talk* as cooperation progresses.

6 Case Study I: Apache Lucene's IRC Discussions (#Lucene-dev)

To validate the results derived from the analytical model, we performed a case study of Apache Lucene's IRC discussion. The case study design follows well-established guidelines in empirical software engineering literature, such as Easterbrook et al (2008); Kitchenham et al (2008); Runeson and Höst (2009), etc. The results of this empirical study provided general support to the theoretical propositions, thus supporting the validity of the analytical model and its results in a real world GSE setting.

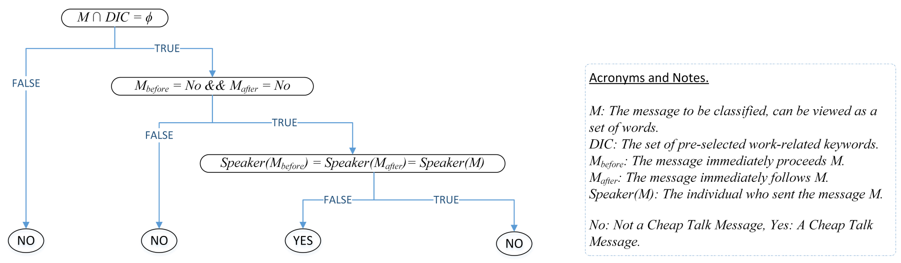
6.1 Case Study Design

LUCENE is an open source information retrieval framework and API. We chose LUCENE and its IRC channel #lucene-dev for the following reasons: (1) LUCENE is a mature project with a stable core development team, which enables us to study long-term interactions; (2) LUCENE has two IRCs: a general IRC channel #lucene and a logged channel #lucene-dev, with #lucene-dev existing solely for the purpose of discussing and recording issues related to development among developers, which allows us to be more focused, as well as reduces "noisy" general user message interference; and, (3) compared with email archives, the interactive, near-synchronous IRC chats are more similar to day-to-day offline conversations. According to Guzzi et al (2013), pure *cheap talk* email threads only account for 0.6% of all emails (see section 3.3). An email may contain both *cheap talk* and work-related talk, making it difficult to classify. Besides, developers participate in less than 75% of all email threads, while development-related threads only account for 35% (Guzzi et al, 2013).

Our study sample comprised all logged messages from 04/15/2010 (the start date of #lucene-dev) to 12/31/2012, which we collected from #lucene-dev's plain text file online archive. We downloaded and extracted messages with three elements: time, sender, and content, excluding the auto-generated login (out) messages. In total, we have 18,216 messages (excluding the simple greeting or confirmation message, e.g., yes/no, "thanks" or "good morning"). Using the three expectations (see Section 5.3) derived from theoretical propositions as working hypotheses, our empirical analysis consisted of the following steps. (1) We classified all messages into two categories: *cheap talk* messages and work-related messages. A lightweight qualitative analysis was performed on all *cheap talk* messages to gain a basic understanding of *cheap talk* in LUCENE. In later analysis, we open-coded and merged them. (2) We identified all work-related dyadic cooperation pairs and *cheap talk* pairs to explore the relationships between *cheap talk* and cooperation (Proposition 1 and 2). (3) We applied statistical methods to study the dynamic pattern of *cheap talk* (Proposition 3). In general, the empirical data provides moderate level support to the theoretical predictions, and no contradiction was found.

6.2 Data Preparation: Message Classification

We developed a simple, rule-based, decision-tree classifier to classify the messages into two categories automatically: *cheap talk* messages and work-related messages. Three heuristics determined classification: (1) *Cheap talk* should


$M \cap DIC = \phi$
TRUE
FALSE
$M_{before} = No \ \&\& \ M_{after} = No$
TRUE
FALSE
$Speaker(M_{before}) = Speaker(M_{after}) = Speaker(M)$
FALSE
TRUE
NO
NO
YES
NO

Acronyms and Notes.

M: The message to be classified, can be viewed as a set of words.
DIC: The set of pre-selected work-related keywords.
M_{before}: The message immediately proceeds M.
M_{after}: The message immediately follows M.
Speaker(M): The individual who sent the message M.

No: Not a Cheap Talk Message, Yes: A Cheap Talk Message.

Fig. 8 The heuristics based decision tree for classifying *cheap talk* messages and corresponding rules.

not contain a set of specific keywords (dictionary[13]) such as: issue, bug, error, commit, Java, etc. (2) If messages immediately before and after an unclassified message are work-related, there is a near impossibility the message is *cheap talk*. (3) *Cheap talk* is usually more interactive than work-related talk, whereas the latter may comprise several messages in clarifying a problem. Therefore, if a message is surrounded by messages from the same sender, it is less likely to be a *cheap talk*. Simple greeting or confirmation messages (e.g., "yes/ok,") were not classified, but they were retained for the rule (3), requiring their presence as context. Fig. 8 describes the decision process and the rule applied to each decision step. The classification yielded two sets of labeled messages. The number of *cheap talk* messages (n=1296) is far less than the number of work-related ones (n=16920).

Since the number of *cheap talk* messages is far less than that of work-related messages, we simply performed a manual post-classification results check. The decision rules ensured that it is almost impossible for a *cheap talk* message to be classified as "work-related." Specifically, the classification method almost only generates *false positive cheap talks*. We randomly sampled 500 messages classified as "work-related" and found only one instance of a *cheap talk* message ($precision = 99.8\%$). After finishing the classification, we read through all of them and manually excluded all false positives ($n = 38$). Compared with more sophisticated techniques (e.g., *D-tree* induction), our simple approach avoided the tricks in constructing proper training set and works well for this specific classification. The dictionary of pre-selected work-related keywords integrates common and domain knowledge into the classifier, and is highly extensible. We compared the results with two classification algorithms (Naive Bayes, Decision Tree) in `Python scikit-learn` library in terms of Precision and Recall. The heuristics-based decision tree classifier outperforms both.

[13] This study's dictionary contains 66 keywords. See Appendix B.1 for the full dictionary.

6.3 Empirical Results and Findings

6.3.1 e-Cheap Talk in #lucene-dev

e-cheap talk covers various topics, including weather, food, hometown, politics, or even personal life.[14] Compared to the social interactions messages in mailing lists (Guzzi et al, 2013), the topics are much more diverse. Furthermore, *cheap talk* can be intentional or impromptu. Intentional *cheap talk* is usually used to show one's friendliness and minimize social distance (Cassell and Bickmore, 2003); for example, presenting some information that may be unfamiliar and interesting for others (example 1), or introducing one's background (example 2). The latter is more situational, and often triggered by an event. For instance, example 3 was triggered when a Russian developer joined the #lucene-dev. In general, intentional *cheap talks* are more common than impromptu ones. For intentional cheap talk, participants may need to interrupt their on-going work, e.g., "back to coding" or similar phrases are common messages after some cheap talks. These words reflect there is a small cost (e in 5) associated with cheap talk over the internet.

> Example 1: *state vs federal rights is still a big topic in the US. do you know about the health care legislation that recently passed the US congress? (American Politics)*
> Example 2: *hamburg is near but its also a separate region. and hanover but it does not share. hannover is the capitol of lower Saxony, so bremen is an island :-) (Hometown)*
> Example 3: *my girlfriend can speak Russian, she is an Estonian. (Personal Life)*

As our model predicts, ignoring *cheap talk* is not proper and triggers punishment. Punishment for uncooperative responses to *cheap talk* does exist and takes various forms. The most common punishment is rejecting a *cheap talk* defector's future cooperation request. However, there is no pre-assigned, centralized authority to administrate the punishment. Rather, it works by way of an individual's personal choice. In our sample, there are six instances of explicit punishments via direct refusals to defector's cooperation request. In one extreme case, the defector left #lucene-dev since no one was interested in cooperating or talking to him any longer. In this case, the punishment (p in 5) is very large. The punishment was administrated by all community members who witnessed the defector's misbehavior. The compensation is not direct, but we may expect those who use cheap talk with defectors may gain some reputation in the community.

6.3.2 e-Cheap Talk and Cooperation

Our study found strong correlations between *cheap talk* and cooperation. We extracted all dyadic cooperation pairs by identifying "send-reply" patterns

[14] We quote three representative examples (example 1-3), making some formatting changes.

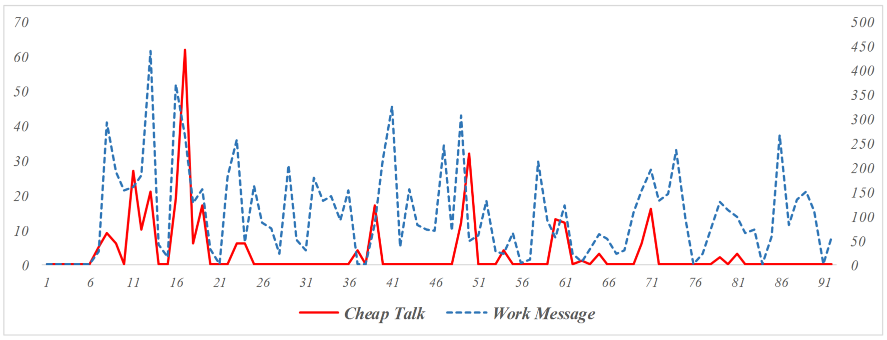


Fig. 9 The dynamics of *cheap talk* and *work-related message* over time (left axis: *cheap talk*, right axis: *work-related message*).

from work-related messages. For *cheap talk* messages, we identified dyadic pairs using the same method. In total, there are 136 pairs of cooperators engaging in work-related discussions, and 101 pairs of *cheap talks*. Almost 70% of work-related pairs (95 in 136) are also in the set of dyadic *cheap talk* pairs. Only six *cheap talk* pairs did not exchange any work-related message. These pairs seem to be those who share the same interests, such as games, but do not have overlap in their work. These talks were often triggered by external events, such as the release of a new Xbox game. We have no conclusive reason for why some work-related pairs did not have any cheap talk. However, 19 out of 35 of these pairs contained at least one highly respected community member. Perhaps people recognize their reputation and automatically trust them. We also noticed a "compensated" individual was involved in three of these pairs. This at least provides some indirect evidence for how compensation works. We further explored the temporal relationship between a *cheap talk* pair and its corresponding work-related pair. 90.5% (86 in 95) *cheap talk* pairs appeared before the corresponding work-related pairs. These results at least partially indicate: (1) *Cheap talk* correlates to cooperation. If two individuals use *cheap talk*, it is probable that they also cooperate. (2) A cooperative relationship may result from *cheap talk*. At least, *cheap talk* is not caused by cooperation, as most cooperation occurs after the appearance of *cheap talk*. These form some support to Propositions 1 & 2. Although we cannot directly measure trust in this study, the literature on virtual collaboration often uses the establishment of cooperation as a good indicator of trust (Bos et al, 2002).

6.3.3 The Dynamics of Cheap Talk over Time

The frequency of *cheap talk* drastically changed since the launch of #Lucene-dev. From 04-15-2010 to 04-30-2011, #lucene-dev was very inactive except for the first two weeks. During that period, there were few examples of *cheap talk*, and the number of work-related messages was also small. Specifically, there were zero chats in 90 consecutive days (12-04-2010 to 03-03-2011). In May

2011, the LUCENE project decided to better utilize the #lecune-dev to ensure the discussions among code contributors would be logged, and discussions did increase in frequency.

However, the dynamic patterns of *cheap talk* and work-related message are quite different. The number of *cheap talks* continued to decrease. *Cheap talk* eventually almost disappeared in about three months after the migration of development activities to #lucene-dev. Fig. 9 visualizes the differing dynamics of both *cheap talk* and work-related message from 05-01-2011 to 07-31-2011. Unlike the decreasing trend of *cheap talk*, there are no significant variations for work-related talk, except for a slight weekly cyclical effect. We performed a simple trending decomposition using the STL method Cleveland et al (1990). The trend for cheap talk message is decreasing while the trend for the work-related message is irregular with slight seasonal effect. More work-related messages appeared during the weekend, perhaps because people usually made open source contributions when they were off from work in the weekend. There were several boosts in *cheap talk*, but those were mainly due to the several core developers joining the new IRC channel later than usual.

By modeling the dynamics of the *cheap talk* and the work-related message as two time series (from 05-01-2011 to 07-31-2011), we statistically tested their cointegration[15] to examine their distinctiveness. We followed *Engle-Granger*'s two-step method: we first constructed the spread through Ordinary Least Square (OLS) regression, and then tested the unit root with the *Augmented Dickey-Fuller* (ADF) test. The test was performed using `R` package `fUnitRoots`. The ADF *p-value* is 0.162, hence these two dynamics are not co-integrated, and do not exhibit similar change patterns.

We also examined the Granger causality (Granger, 1988) between two time series, using the `grangertest` in `R` library `forecast`. We parameterized a series of different lags from 1 to 7, which represents exactly one week in time range. We perform 14 Granger causality tests (cheap talk $\rightarrow$ work-related message: 7, work-related message $\rightarrow$ cheap talk: 7). None of them are significant (minimal $p - value = 0.261$). The results fit our expectation that the decrease in cheap talk message did not result from the change in work-related message (or vice versa), at least not in the sense of Granger causality.

Exactly as the model predicted, the *e-cheap talk* plays an important role in developing cooperation and trust. Once cooperation and trust becomes stable, an individual may not be willing to pay the cost of *e-cheap talk*, directly switching to *cooperate* mode, since she can expect and trust the others will have similar expectation or behavioral choices. This supports the theoretical proposition 3's prediction of the decreasing trend of *e-cheap talk* in the long run.

[15] Two time series are cointegrated if they share a common stochastic drift, hence are mutually predictable, see Pfaff (2008) for more details.

7 Case Study II: Chromium OS' IRC Discussions (#chromium-os)

The analyses towards Lucene's IRC discussions provide general support to the theoretical results. To further validate theoretical results, we performed another case study with the IRC discussions of Chromium OS (#chromium-os). It is the project behind Chrome OS, which is a cloud-based operating system based on the Linux kernel and designed to work with web applications and installed applications.

Although Chromium OS is an open source project, it is different from Lucene due to the heavy involvements of a commercial organization (i.e., Google). Many developers are from the same company, and hence their participation may be quite different from voluntary behaviors in #lucene-dev (Wagstrom, 2009). For example, some developers may know each other fairly well offline, and may be not so "free" to leave the community. This allows us to examine our results' generalizability in a slightly different setting. Google does have an internal IRC channel; however, many Google employees still maintain their presences in #chromium-os[16]. One disadvantage is that #chromium-os does not only contain messages from developers, although it is the designated development IRC channel. Due to the popularity of Chromebook in the general public, many end users who are supposed to use #chromium-os-users and #chromium-os-discuss do post something irrelevant on it. We had to restrict the time frame when constructing the study sample.

7.1 Case Study Design and Data Preparation

The case study design is almost as same as the prior one except for several small differences. First, the study sample consists of the first 18 months (11/19/2009 - 04/30/2011) of IRC messages, to ensure the quality of the data (Prikladnicki et al, 2014). There was a substantial amount of messages from end users since 05/2011, due to the debut of Chromebook to mass consumer electronics market in June/July 2011. Since then, messages from end users account for a nontrivial proportion ($\geq$ 5%) in all messages, while almost all discussions were among developers in the first 18 months. Second, #chromium-os is not an officially logged IRC channel, so we utilized an unofficial log on echelog[17].

The study sample contains 93617 messages. We adapted the "dictionary" used in message classification while keeping the same classification rules. 19 Lucene specific keywords were dropped, and 71 new Chromium OS specific keywords (most of them are the names of tools and Linux package) were added (See APPENDIX B.2). Using the same data processing steps, we found 3280

[16] We retrieved the "tree" (the sum of the various source repositories used to build the project) status history on http://chromiumos-status.appspot.com/. Then, we manually matched the usernames in tree status (associated with email address) with #Chromium-OS users and found the majority of #chromium-os active code contributors have email addresses in the form of "XXX@google.com", which indicates they are Google employees. We also consulted their Linkedin profile to determine their affiliations.

[17] http://echelog.com/logs/browse/chromium-os/

cheap talk messages and 90337 *work-related* messages. We read all *cheap talk* messages, and 3127 messages remained "cheap talk" after excluding 153 false positives. We performed a similar precision test for work-related messages. In 500 randomly sampled messages classified as "work-related," no cheap talk message was identified ($\approx 100\%$ *precision*). The classification results achieve acceptable precision, although we cannot rule out the possibility that some cheap talk messages were mislabeled. The number of mislabeled messages should be at a relatively low level.

7.2 Empirical Results and Findings

7.2.1 e-Cheap Talk in #chromium-os

The fraction of *cheap talk* in #chromium-os is much lower (3.45% vs. 6.96%) than that in #lucene-dev. However, the topic is even more diverse due to the larger number. Besides the topics we mentioned before, we noticed there is a specific set of cheap talk messages poking fun at competitors and their products. These messages usually arouse interesting and funny conversations. For example, a contributor shared a link to Google's stock price, and another one commented: "*that's because shareholders probably expected a Microsoft killer...like journalists*". Since developers may come from the same company (even same department/team), using this strategy may help them quickly regain their offline common ground. Another noticeable fact is that the cheap talks were not in the one-to-one conversation patterns prevalent in #lucene-dev; rather, more than half of the cheap talks (1632 in 3127) took a "meeting" style in #chromium-os. That is, an individual posts a cheap talk message without explicitly specifying who is the receiver in #chromium-OS, then some other people talk about it when they see it. It is more like a group of people coming together and chatting about funny things during their lunch than a one-to-one conversation. In fact, the many-to-many strategy is more cost-effective. In #chromium-os, many contributors may have already known each other quite well in their offline lives and be confident with their peers, so they are confident to take the many-to-many strategy and broadcast their willingness to cooperate. Whereas an individual may have to talk to everyone in the team in order to ensure their willingness to cooperate will be properly delivered to those unfamiliar collaborators #lucene-dev.

7.2.2 e-Cheap Talk and Cooperation

Although the majority of cheap talks took the "meeting" style, there are still some correlations between *cheap talk* and cooperation. Using the same method, we extracted all dyadic pairs of "work-related" cooperation and *cheap talk* relationships. In total, we identified 239 pairs of cooperators who have explicit cooperation (one mentioned the other in at least one message), and 125 pairs of individuals who have explicit cheap talks. 46% of work-related pairs (110 in 239) are also in the set of dyadic *cheap talk* pairs. 15 *cheap talk* pairs did not have any explicit work-related talk. 42.7% *cheap talk* pairs (47 in 110) appeared before the corresponding work-related pairs. Compared with the results from Case Study I, the effects are not so strong, e.g., less than half of the work-related pairs were developed after the occurrence of *cheap talk* pairs. However, most individuals participated in meeting-style many-to-many cheap talks during the project kick-off. Counting these cheap talks, it is still true that the majority of work-related pairs followed *cheap talk*.

7.2.3 Cooperation Reparation with e-Cheap Talk

While individuals simply quit participating if they were explicitly punished by others in #lucene-dev, developers proactively took efforts to repair trust and cooperation even before explicit punishment in #chromium-os. For instance, one developer (Alpha) insisted he had fixed a bug (actually he did not), and behaved in a very uncooperative manner when another contributor (Beta) attempted to use a joke about "bug" to persuade him to review the code together. Beta then stopped the conversation and left. However, once Alpha realized his misbehavior, he made a joke about himself and mentioned Beta as the receiver without an explicit apology. Beta accepted this, and they started to work together to review Alpha's code. In this example, both Alpha and Beta seem to accept *cheap talk* as a proper and sufficient action to repair the cooperation. Many Chromium OS contributors are very probable to working in the same company and the contribution to the project may be part of their regular jobs. They are not free to quit their participation in #chromium-os without any negative influence on their career. Therefore, they tend to switch from *defect* to *C-C* once they realize that *defect* is not proper. In this scenario, we find that formal offline social realities (e.g., co-worker) in an organization may facilitate cooperation development in virtual collaboration.

7.2.4 The Dynamics of Cheap Talk over Time

Fig. 10 plots the differing change patterns of both cheap talk and work-related message from 11-21-2009[18] to 02-28-2010. At the beginning, both the number of *cheap talk* messages and work-related messages (most of them are about setting up the development environment) are high. However, the number of *cheap talk* messages dropped very fast and almost disappeared after two weeks. In total, the cheap talks in the first month accounts for over 1/3 of all cheap talks in the whole sample (18 months). Although there were less work-related messages, which also dropped after reaching its peak on the second day, the number remains at a relatively stable level. We also utilized co-integration analysis to examine their distinctiveness. The ADF *p-value* is 0.379, and hence these two dynamics are not co-integrated and do not exhibit similar change patterns. Trend decomposition analysis also suggests similar results. However,

[18] No message in 11/19 and 11/20.

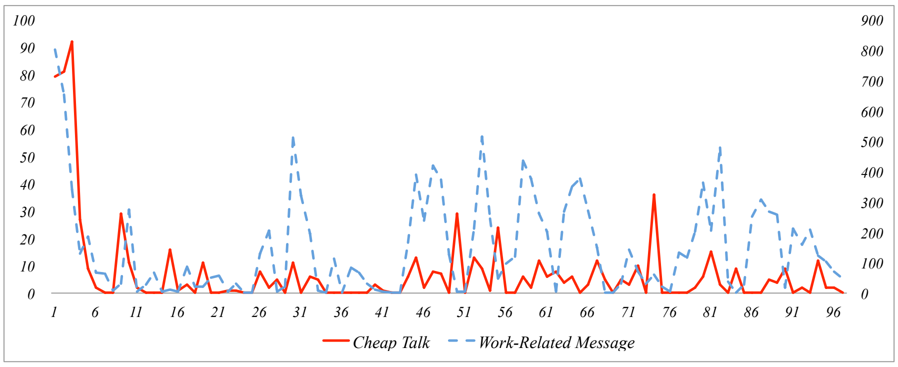


Fig. 10 The dynamics of *cheap talk* and *work-related message* over time, (left axis: *cheap talk*, right axis: *work-related message*).

we did not observe the seasonal effect for work-related messages. As we did before, we performed a series of Granger causality tests. Again, no Granger causality was found (minimal $p-value = 0.302$). The disappearance of cheap talk was not caused by the change in work-related talk.

8 Discussion

Table 1 summarized the research questions, theoretical propositions, and corresponding empirical evidence. Combining game theory deduction and empirical study, the study described in this paper provides an alternative explanation of how, in the presence of punishment, *e-cheap talk* can create trust. In the Stag-hunt game, cooperation requires at least moderate trust. Without trust, individuals may not expect others to choose the cooperate strategy in their interaction, and thereby defect to avoid the potential risk. With *e-cheap talk* acting as a **catalyst**, trust emerges from and ensures cooperation. The *e-cheap talk* guarantees the loss in a failed "cooperation trial" can be offset by punishment/compensation enforcement, leading to increased willingness to cooperate. In this process, the trust toward others, or more precisely, the trust of others' "rationality" eventually develops and "cooperation" becomes the mainstream choice. The improvement of cooperation can be observed, while trust development is implicit (not easily directly observed at the behavioral level). However, they are essentially two faces of the same process. Even in the case that people already knew each other very well (Case Study II), they still utilized cheap talk at the early phase of virtual collaboration. The model in this paper employs cost-benefit perspective to study "social interactions" among software developers in strategic level brings new insights. For example, our model predicts that *e-cheap talk* will gradually disappear with cooperation's progress, which is rarely mentioned in the prior literature. We furthermore demonstrate *e-cheap talk's* positive impact in GSE collaboration.

Table 1 Summary and mapping of research questions, theoretical propositions, and empirical findings

RQs	**Theoretical Propositions**	**Empirical Findings**	
		Study I: #lucene-dev *(Total: 18216 messages)* *(Cheap talk: 1296 messages)*	Study II: #chromium-os *(Total: 93617 messages)* *(Cheap talk: 3280 messages)*
RQ1	Prop. 1	Section 6.3.1, 6.3.2	No direct evidences
RQ2	Prop. 2	Section 6.3.1, 6.3.2	Section 7.2.2, 7.2.3
RQ3	Prop. 3	Section 6.3.3	Section 7.2.4

8.1 e-Cheap Talk Jump-Starts Trust

The results presented in this paper reinforce social chat's role in jump-starting trust and cooperation (Zheng et al, 2002) in the early phase of collaboration, as well as the quick decrease of using this "jump-starter" after the establishment of trust and cooperation. The empirical results show that the jump-starter (cheap talk) is not only used in the early phase of collaboration, but also applied to repair trust (see Section 7.2.3). These findings provide more comprehensive and direct support to the "jump-starter" metaphor in real distributed collaboration settings. Also, the theoretical model provides a plausible explanation for the mechanism underlying the "jump-starter" media effect. When a collaborator see a cheap talk message from another one, he or she will be more confident to assume that the other collaborator will behave cooperatively in following interactions. In case in which trust was broken, cheap talk represents the intentions to repair the relationship even with extra efforts.

8.2 Cooperation & Trust in A Social Network

The model we proposed does not consider a network effect. All players are equally important with respect to their positions in a fully connected network. However, in the real world, team members are often located in their social network with different positions (Gharehyazie et al, 2014). The network structure influences their communication and interaction (Damian et al, 2007), hence affects the evolution of cooperation and trust development. In such cases, the social network plays a key role in enhancing organizational learning with key members being a source of information for software engineering activities (Tamburri et al, 2013). For example, people in different locations often have to communicate through specific individuals as "hubs." If trust were first built among these "hubs," it may take much less time for the whole team to develop trust. Moreover, top-down, dictator-style communication may be disadvantageous to the emergence of cooperation and trust (Burt et al, 2013). Furthermore, the dynamic nature of developer networks (Hong et al, 2011) will also allow us to investigate the cooperation network's evolution over the time. In future research, we plan to extend the model presented in this paper to incorporate social network effects.

8.3 Design Implications

Our theoretical and empirical results lead to some design implications. For example, as equation (8) indicates, ensuring the transition from *defect* to *C-C* requires the punishment/incentive to be comparable to the cost of *e-cheap talk*. Without drastically increasing the punishment, reducing the cost of *e-cheap talk* is a matter-of-course (see the right part of the equation(8)). Innovative tools may help to fulfill this requirement, and organizational networks will potentially improve or hinder cooperation's development. Hence, redesigning organizational structure would help. We will discuss the implications to both collaboration management and tools.

8.3.1 Implication to Collaboration Management

The rapid development of trust and cooperation in globally distributed software teams has been a long lasting problem (Olson and Olson, 2000; Al-Ani and Redmiles, 2009), and cheap talk over the Internet may provide a sound solution. Sharing non-work-related information should be encouraged, especially in the early phase of cooperation and trust development (see our propositions in Section 5.3). Prior studies (Schumann et al, 2012) show that even a personal picture may be very helpful to trust-building. Furthermore, organizational communication structure affects collaboration, work performance, and the quality of software deliverables (Herbsleb and Grinter, 1999; Kwan et al, 2012). Specific communication structures promote or hinder trust and collaboration development among developers (Layman et al, 2006). As we mentioned in Section 8.2, the top-down, dictator-style communication is disadvantageous to the emergence of cooperation and trust. Therefore, cooperation and trust would benefit from increased participation of general developers who are not in central positions, which may eventually speed up global level collaborations (Herbsleb and Mockus, 2003). Furthermore, although some communication network structures are very efficient, they are not stable enough for the imbalance of cost. For example, the individual (often leaders of project) in the center have to pay most of the cost, since all communications rely on her as a "repeater." If the cost is high, the network may break down before the development of cooperation. Hence, reducing the cost is crucial. Open and transparent communication will help to reduce the cost and mitigate the imbalance of cost distribution among members (Dabbish et al, 2012), and thus should be encouraged in management practices (e.g., logging all conversations). The model developed in this paper provides a framework to enable agent-based simulation for examining mechanism design alternatives (Ren and Kraut, 2014).

8.3.2 Implications to Tools Design & Usage

Both software engineering and CSCW communities agree that interaction among software developers should be encouraged (Wagstrom et al, 2010). Many collaboration tools have been designed to encourage this interaction.

However, few intelligent tools can tell software engineers how to interact with their remote collaborators, who may be quite different from them. Using game theory as the analytical engine for such intelligent tools may be a good alternative. By simulating and summarizing interaction scenarios with pre-defined dynamics and individual characteristics, users would be able to identify which strategy brings the best payoff. We are currently working on an intelligent tool that suggests the proper strategies for interacting with an unfamiliar collaborator. The tool extracts information from an individual's past work traces. Using game theory modeling and simulation as the core engine, it can help a developer decide which is a proper strategy (i.e., trust without any reservation or cheap talk first) at collaboration's initial stage.

In current practices, Web 2.0 and social media tools (Storey et al, 2014) are primary *e-cheap talk* channels. However, (Al-Ani et al, 2012) pointed out that their utilization is not encouraged; for instance, managers often believe these tools cause interruptions to normal work, and hence do not want to support their use at the organizational policy level. Our analysis shows that *e-cheap talks* over Web 2.0 tools are worth a try. Managers may need to be more open to the usage of Web 2.0 tools to promote *cheap talk*; a more proactive/aggressive alternative may be to integrate *cheap talk* into normal workflow. A developer does not need to intentionally interrupt work to initiate *cheap talk*; for instance, an automatically-captured photo of the developer shared on a group page offers a form of cheap information. This approach can serve as an alternative to integrating IDE with social media, e.g., internal "Facebook" plug-ins for Eclipse IDE (Calefato et al, 2011). These tools would also help to reduce the cost of *cheap talk*. In some situations, although *cheap talk's* cost is small, its distribution may be very imbalanced, i.e., a few people always pay the cost (usually several core members). A simple tool that balances individual effort may solve this problem. For example, the tool can identify the members less frequently initiating *e-cheap talk* by analyzing the logs, and then suggest they take initiative with *e-cheap talk*.

8.4 Methodological Implications

The human and social aspects of software engineering are increasingly well studied in the software engineering area (Harper et al, 2013). A large body of empirical literature has focused on many aspects of human interactions in software development, including coordination and trust. Empirical studies contribute rich evidence of real-world practice, but are relatively void of explicit theory. While greatly informing research, approaches that can combine empirical studies and predictive theories have obvious advantages. When assumptions are clearly articulated, a theory can provide predictive power and avoid costly "trial and error" decisions.

Particularly in studying human and social aspects of software engineering, we argue that game theory provides an ideal analytical framework for developing theoretical knowledge. Software engineering processes consist of human decision-making activities. Developers have to decide proper strategies for team member cooperation. Game theory has demonstrated its capability in studying human decision-making processes and the long-term attributes of social systems since the 1950s, and continues to introduce new perspectives to cooperation research (Nowak, 2013). Via game theory model reasoning, we can deductively develop propositions characterizing real-world phenomena, which provides generative causality and explanation (Cederman, 2005).

The generative approach of game theory models comports well with current empirical practices in SE research. All theoretical models and reasoning, no matter how sophisticated, are imperfect real world abstractions. Their external validity must be validated through comparison to empirical/experimental results. This process forms a feedback loop between empirical and theoretical work. SE research may benefit from the combination of theoretical and empirical study, which helps to build generalizable and rigorous knowledge, while maintaining strong connections to reality. Leveraging EGT, we made some preliminary attempts in this direction. The study was motivated by real world phenomenon (Section 2) and presented theoretical explanations (Section 3, 4, & 5). We validated the theoretical propositions through two empirical case studies (Section 6, & 7). Our study demonstrates the feasibility of this approach and shows how to use it to develop sense-making theory and generalizable empirical knowledge. However, we admit that a limitation of this methodology is that the theoretical model and its propositions are somewhat hard to validate in some situations using direct empirical evidence. Researchers may need to pay extra attention to empirical study design, or employ several different empirical methods.

8.5 Threats to Validity

The research approach helps us to achieve high internal and external validity. Game theory modeling requires only minimal assumptions that do not involve too much subjective bias. The empirical studies of LUCENE and CHROMIUM OS rely on open human communication records, and hence suffers less from the validity issues raised in data collection practices. The message classification was automatically executed with minimal subjective interventions, further preventing significant internal validity threats. The theoretical analyses also help to solve the generalizability problem of empirical study. However, a potential threat to validity is that the team is fixed in its life cycle, whereas OSS teams are usually very flexible. In its life cycle, newcomers may join, while some familiar faces may leave. However, this does not mean the model's results are essentially wrong. First of all, the "fixed" population assumption only requires the population has no potential to be infinite. Moreover, LUCENE's core team is very stable, and the timeline for achieving universal cooperation is relatively short (at most several months). CHROMIUM OS' project team is even more stable, for many contributors work for Google. Nevertheless, this issue needs to be further investigated in more stable software development teams. It is also possible that the results may be generalizable to other types of virtual collaborations (e.g., collaborative editing or knowledge sharing) if these collaborations could be abstracted into the form of the extended Stag-hunt game. However, this generalizability also needs empirical validation in the specific settings to fulfill the requirements of this paper's approach.

Another limitation is that the model parameter values are assigned rather than surveyed. Although we are confident in their relative quantitative relationships, e.g., the cost of cheap talk (e) $<<$ the benefit of cooperation (R), their absolute value is not precisely estimated. To solve this problem, researchers, including ourselves, may perform field studies to survey software developers and identify their perceptions of these parameters. For example, we may ask them to report the proportion of their perceived cost of cheap talk to the benefit of successful cooperation. We may also collect their perception of punishment/compensation in the team. It also inspires another potential methodological integration of empirical and analytical approaches, i.e., deriving the abstract model and its parameters from empirical observation. Hence, the analytical model will be more realistic and relevant to real world practices.

9 Conclusions and Future Work

In this paper, we investigate how informal non-work related conversations over the Internet (*e-cheap talk*) help to promote cooperation and trust in software development teams. We employ a unique cost-benefit perspective using a novel EGT model inspired by a previous field study. We performed a subsequent assessment by data mining the development IRC channels of LUCENE and CHROMIUM OS. Together, our model and empirical studies lead us to conclude that *e-cheap talk* can help distributed teams build cooperation and trust. However, *e-cheap talk* works as a catalyst in trust and cooperation development, and tends to gradually disappear in the long run.

Studies of trust and other human factors in software development will benefit from rigorous models that enable analytical study and computer simulation and, in particular, employ EGT to better examine the emergence of complex social behavior. Doing so will be informative to organizational concerns as well as software tool development. Furthermore, this study at least partially demonstrates the feasibility of combining theoretical modeling and empirical study towards developing rigorous, generalizable knowledge that can reflect and guide real world practices. More empirical work is needed to show how the model and the real-world are connected. Until then, it is still an open question as to how much EGT can tell us about the dynamics of interactions among distributed collaborators in software engineering activities. We have not yet provided a definitive answer to this question, but the preliminary results reported in this paper are encouraging, at least for trust research in GSE. We will further collect empirical data to validate our model and its analytical results. For instance, we plan to collect longitudinal trust data to directly assess the dynamics of trust (Proposition 2), which this study has not done. We will also develop novel tools for supporting software engineers' decision-making in interactions with unfamiliar collaborators.

The model can be extended to incorporate other factors. For instance, the current model only considers the "punishment/incentives" mechanism, but, in the real world, other factors (e.g., commitment, emotion, culture) also regulate the trust and cooperation process. For instance, by imposing a "commitment" mechanism, it is possible to investigate the trust damage and repair process (Al-Ani et al, 2013; Schniter et al, 2013) and how to motivate individuals to deliver their commitment (Winikoff, 2007). People form subgroups to exercise different strategies to confirm their subgroup identities. It is possible to integrate the EGT model and Social Network Analysis to explore complex real-world interpersonal relationships (Jackson, 2010). The discussions in Section 8.2 also demonstrate the promise of this combination. We will continue our research in this direction.

Acknowledgements We would like to thank Prof. Brian Skyrms for helping us refine the model and the analyses. We appreciate Dr. Ban Al-Ani for collaborating on the empirical data in Section 2. We would like to thank the anonymous reviewers for their insightful comments and suggestions.

References

Al-Ani B, Redmiles D (2009) In strangers we trust? findings of an empirical study of distributed teams. In: Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on, IEEE, pp 121–130

Al-Ani B, Wang Y, Marczak S, Trainer E, Redmiles D (2012) Distributed developers and the non-use of web 2.0 technologies: A proclivity model. In: Proc. ICGSE, pp 104–113

Al-Ani B, Bietz MJ, Wang Y, Trainer E, Koehne B, Marczak S, Redmiles DF, Prikladnicki R (2013) Globally distributed system developers: their trust expectations and processes. In: Proc. CSCW, pp 563–574

Banerjee A, Chandrasekhar AG, Duflo E, Jackson MO (2013) The diffusion of microfinance. Science 341(6144):1236,498

Bos N, Olson J, Gergle D, Olson G, Wright Z (2002) Effects of four computer-mediated communications channels on trust development. In: Proc. CHI, pp 135–140

Bowles S, Gintis H (2011) A cooperative species: Human reciprocity and its evolution. Princeton University Press

Burt RS, Kilduff M, Tasselli S (2013) Social network analysis: Foundations and frontiers on advantage. Annual Review of Psychology 64(1):527–547, DOI 10.1146/annurev-psych-113011-143828

Caglayan B, Bener A, Miranskyy A (2013) Emergence of developer teams in the collaboration network. In: Proc. CHASE

Calefato F, Lanubile F, Sanitate N, Santoro G (2011) Augmenting social awareness in a collaborative development environment. In: Proceedings of the 4th international workshop on Social software engineering, ACM, pp 39–42

Calefato F, Lanubile F, Sportelli F (2013) Can social awareness foster trust building in global software teams? In: Proc. SSE, pp 13–16

Camera G, Casari M, Bigoni M (2013) Money and trust among strangers. Proceedings of the National Academy of Sciences 110(37):14,889–14,893

Cassell J, Bickmore T (2003) Negotiated collusion: Modeling social language and its relationship effects in intelligent agents. User Modeling and User-Adapted Interaction 13(1-2):89–132

Cataldo M, Herbsleb JD, Carley KM (2008) Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In: Proc. ESEM, pp 2–11

Cederman L (2005) Computational models of social forms: Advancing generative process theory. American Journal of Sociology 110(4):pp. 864–893

Chiappori PA, Levitt S, Groseclose T (2002) Testing mixed-strategy equilibria when players are heterogeneous: The case of penalty kicks in soccer. American Economic Review pp 1138–1151

Cleveland RB, Cleveland WS, McRae JE, Terpenning I (1990) Stl: A seasonal-trend decomposition procedure based on loess. Journal of Official Statistics 6(1):3–73

Cramton CD, Hinds PJ (2007) Intercultural interaction in distributed teams: Salience of and adaptations to cultural differences. In: Proc. AOM Annual Meeting, pp 1 – 6

Dabbish L, Stuart C, Tsay J, Herbsleb J (2012) Social coding in github: transparency and collaboration in an open software repository. In: Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, ACM, pp 1277–1286

Damian D, Izquierdo L, Singer J, Kwan I (2007) Awareness in the wild: Why communication breakdowns occur. In: Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on, IEEE, pp 81–90

Dittrich Y, Giuffrida R (2011) Exploring the role of instant messaging in a global software development project. In: Proc. ICGSE, pp 103–112

Ducheneaut N (2005) Socialization in an open source software community: A socio-technical analysis. Computer Supported Cooperative Work (CSCW) 14(4):323–368

Easley D, Kleinberg J (2010) Networks, crowds, and markets: Reasoning about a highly connected world. Cambridge University Press

Easterbrook S, Singer J, Storey MA, Damian D (2008) Selecting empirical methods for software engineering research. In: Guide to advanced empirical software engineering, Springer, pp 285–311

Erev I, Roth AE (1998) Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. American economic review pp 848–881

Farrell J, Rabin M (1996) Cheap talk. The Journal of Economic Perspectives 10(3):103–118

Ficici S, Pollack J (2000) Effects of finite populations on evolutionary stable strategies. In: GECCO, pp 927–934

Fudenberg D, Imhof LA (2006) Imitation processes with small mutations. Journal of Economic Theory 131(1):251 – 262, DOI 10.1016/j.jet.2005.04.006

Fudenberg D, Imhof LA (2008) Monotone imitation dynamics in large populations. Journal of Economic Theory 140(1):229 – 245, DOI 10.1016/j.jet.2007.08.002

Gharehyazie M, Posnett D, Vasilescu B, Filkov V (2014) Developer initiation and social interactions in oss: A case study of the apache software foundation. Empirical Software Engineering pp 1–36, DOI 10.1007/s10664-014-9332-x

Gintis H (2000) Game theory evolving: A problem-centered introduction to modeling strategic behavior. Princeton University Press

Granger CW (1988) Some recent development in a concept of causality. Journal of econometrics 39(1):199–211

Guzzi A, Bacchelli A, Lanza M, Pinzger M, van Deursen A (2013) Communication in open source software development mailing lists. In: Proc. MSR, pp 277–286

Harper R, Bird C, Zimmermann T, Murphy B (2013) Dwelling in software: Aspects of the felt-life of engineers in large software projects. In: ECSCW 2013: Proceedings of the 13th European Conference on Computer Supported Cooperative Work, 21-25 September 2013, Paphos, Cyprus, Springer, pp 163–180

Herbsleb JD, Grinter RE (1999) Splitting the organization and integrating the code: Conway's law revisited. In: Proc. ICSE, pp 85–95

Herbsleb JD, Mockus A (2003) An empirical study of speed and communication in globally distributed software development. Software Engineering, IEEE Transactions on 29(6):481–494

Herbsleb JD, Atkins DL, Boyer DG, Handel M, Finholt TA (2002) Introducing instant messaging and chat in the workplace. In: Proc. CHI, pp 171–178

Hong Q, Kim S, Cheung SC, Bird C (2011) Understanding a developer social network and its evolution. In: Proceedings of the 2011 27th IEEE International Conference on Software Maintenance, IEEE Computer Society, Washington, DC, USA, ICSM '11, pp 323–332, DOI 10.1109/ICSM.2011.6080799

Jackson M (2010) Social and Economic Network. Princeton University Press

Jackson M, Rodriguez-Barraquer T, Tan X (2012) Social capital and social quilts: Network patterns of favor exchange. The American Economic Review 102(5):1857–1897

Jarvenpaa SL, Shaw TR, Staples DS (2004) Toward contextualized theories of trust: The role of trust in global virtual teams. Information systems research 15(3):250–267

Jensen C, Riestenberg G (2012) Stag-hunt Game. http://goo.gl/ieiLmX, [Online; accessed 12-12-2014]

Karlin S (2014) A first course in stochastic processes. Academic press

Kitchenham B, Al-Khilidar H, Babar MA, Berry M, Cox K, Keung J, Kurniawati F, Staples M, Zhang H, Zhu L (2008) Evaluating guidelines for reporting empirical software engineering studies. Empirical Software Engineering 13(1):97–121

Kwan I, Cataldo M, Damian D (2012) Conway's law revisited: The evidence for a task-based perspective. Software, IEEE 29(1):90–93

Layman L, Williams L, Damian D, Bures H (2006) Essential communication practices for extreme programming in a global software development team. Information and software technology 48(9):781–794

March JG, Simon HA (1993) Organizations. Second Edition. Wiley

Mislin AA, Campagna RL, Bottom WP (2011) After the deal: Talk, trust building and the implementation of negotiated agreements. Organizational Behavior and Human Decision Processes 115(1):55 – 68

Montuschi E (2003) The Objects of Social Science. Continuum International Publishing Group

Nowak M (2013) Evolution, Games, and God. Harvard University Press

Nowak MA (2006a) Evolutionary Dynamic: Exploring the Equations of Life. The Belknap Press of Harvard University

Nowak MA (2006b) Five rules for the evolution of cooperation 314(5805):1560–1563

Nowak MA, Sasaki A, Taylor C, Fudenberg D (2004) Emergence of cooperation and evolutionary stability in finite populations. Nature 428(6983):646–650, DOI 10.1038/nature02414

Nowak MA, Tarnita CE, Wilson EO (2010) The evolution of eusociality. Nature 466:1057–1062

Olson GM, Olson JS (2000) Distance matters. Human-computer interaction 15(2):139–178

Pendharkar PC, Rodger JA (2009) The relationship between software development team size and software development cost. Commun ACM 52(1):141–144, DOI 10.1145/1435417.1435449, URL http://doi.acm.org/10.1145/1435417.1435449

Pereira LM, Santos FC, et al (2012) The emergence of commitments and cooperation. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, pp 559–566

Pfaff B (2008) Analysis of Integrated and Cointegrated Time Series with R. Springer

Prikladnicki R, Boden A, Avram G, de Souza CR, Wulf V (2014) Data collection in global software engineering research: learning from past experience. Empirical Software Engineering 19(4):822–856

Puranam P, Alexy O, Reitzig M (2014) What's new about new forms of organizing? Academy of Management Review 39(2):162–180

Rabin M (1993) Incorporating fairness into game theory and economics. The American economic review pp 1281–1302

Ren Y, Kraut RE (2014) Agent-based modeling to inform online community design: Impact of topical breadth, message volume, and discussion moderation on member commitment and contribution. Human-Computer Interaction 29(4):351–389, DOI 10.1080/07370024.2013.828565

Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering 14(2):131–164

Santos FC, Pacheco JM, Skyrms B (2011) Co-evolution of pre-play signaling and cooperation. Journal of Theoretical Biology 274(1):30 – 35, DOI 10.1016/j.jtbi.2011.01.004

Schniter E, Sheremeta RM, Sznycer D (2013) Building and rebuilding trust with promises and apologies. Journal of Economic Behavior & Organization 94:242–256, DOI 10.1016/j.jebo.2012.09.011

Schumann J, Shih PC, Redmiles DF, Horton G (2012) Supporting initial trust in distributed idea generation and idea evaluation. In: Proc. GROUP, pp 199–208

Simon HA (1991) Bounded rationality and organizational learning. Organization science 2(1):125–134

Skyrms B (2001) The stag hunt. In: Presendential Address of the Pacific Division of the APA, Proceedings and Addresses of the APA 75, pp 31–41

Skyrms B (2008) Trust, risk, and the social contract. Synthese 160(1):21–25

Skyrms B (2010) Signals: Evolution, learning, and information. Oxford University Press

Skyrms B (2014) Social dynamics. Oxford University Press

Steed A, Spante M, Heldal I, Axelsson AS, Schroeder R (2003) Strangers and friends in caves: an exploratory study of collaboration in networked ipt systems for extended periods of time. In: Proc. I3D, pp 51–54

Storey MA, Singer L, Cleary B, Figueira Filho F, Zagalsky A (2014) The (r) evolution of social media in software engineering. In: Proceedings of the on Future of Software Engineering, ACM, pp 100–116

Tamburri DA, Lago P, Vliet Hv (2013) Organizational social structures for software engineering. ACM Comput Surv 46(1):3:1–35

Traulsen A, Nowak MA, Pacheco JM (2006) Stochastic dynamics of invasion and fixation. Phys Rev E 74:011,909, DOI 10.1103/PhysRevE.74.011909, URL http://link.aps.org/doi/10.1103/PhysRevE.74.011909

Wagstrom P (2009) Vertical interaction in open software engineering communities. PhD thesis, Carnegie Mellon University

Wagstrom P, Herbsleb JD, Carley KM (2010) Communication, team performance, and the individual: Bringing technical dependencies. In: Academy of Management Proceedings, Academy of Management, vol 2010, pp 1–7

Wang Y, Redmiles D (2013) Understanding cheap talk and the emergence of trust in global software engineering: An evolutionary game theory perspective. In: Proc. CHASE, pp 149–152

Wang Y, Zhang M (2010) Penalty policies in professional software development practice: a multi-method field study. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2, ACM, pp 39–47

Winikoff M (2007) Implementing commitment-based interactions. In: Proc. AAMAS, pp 128:1–8

Young HP (1998) Individual Strategy and Social Structure: An Evolutionary Theory of Institutions. Princeton University Press

Zheng J, Veinott E, Bos N, Olson JS, Olson GM (2002) Trust without touch: Jumpstarting long-distance trust with initial social activities. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '02, pp 141–146

A Punishment vs. Compensation

In this paper, we simply assume the punishment to the defector equals the compensation to the cheap talker. This assumption has been widely adopted in game theory literature, for it is essentially a redistribution of payoffs between both players (Pereira et al, 2012). The redistribution-based punishment/compensation mechanism is also adopted in professional software engineering practices (Wang and Zhang, 2010). In our case, when a cheap talker meets a defector, the total payoff is $(T - g) + (S + g - e) = T + S - e$. There is no loss during the administration of punishment and compensation. In other words, the punishment/compensation process is free of cost. Given that most punishments are spontaneous and lack any specific administrator, it is a reasonable solution. However, it is not perfect. It is possible that the punishment/compensation process incurs some cost or even creates some extra value in forms of psychological payoff (Rabin, 1993) or advance payment of future benefits.

When punishment does not equal compensation, we can rewrite the game as the following form:

$$
\begin{array}{c|ccc}
 & C\text{-}C & C & \\
\hline
C\text{-}C & R\text{-}e/2 & R\text{-}e & S\text{-}e\text{+}f \\
C & R & R & S \\
D & T\text{-}g & T & P
\end{array}
$$

Fig. 11 Payoff structure when punishment (g) does not equal compensation (f).

We tested whether the main analytical conclusions are still valid when punishment and compensation are not equal. We performed simulations for different combinations of g and f. We assume g and f are in $(0, 1]$, and the interval is 0.1. So g and f can be 0.1, 0.2,... 1. They form 100 combinations. For each combination of g and f, we ran 1000 independent rounds of simulation. The other parameters were from the numerical example in fig. 5, and did not change. The grey-scale in each cell of the following diagram indicates the percentage of simulations that achieve full-cooperation in a specific combination of g and f.

Obviously, cooperation will become prevalent if:

1. Both g and f are comparable to the cost e,
2. Either of g or f is large enough.

This indicates that the validity of the analytical results does not wholly rely on the relative quantitative relationship between punishment and compensation.

With the new game, we need to re-specify the conditions for transitions from C-C to C, and D to C-C. First of all, the condition for transitions from C-C to C remains the same as equation 6, as the payoff structure between C-C and C does not change. For conditions for transitions from D to C-C, we can rewrite equation 7 as:

$$R - \frac{e}{2} + 2(S + f - e) > T - g + 2P \;\Rightarrow \tag{9}$$

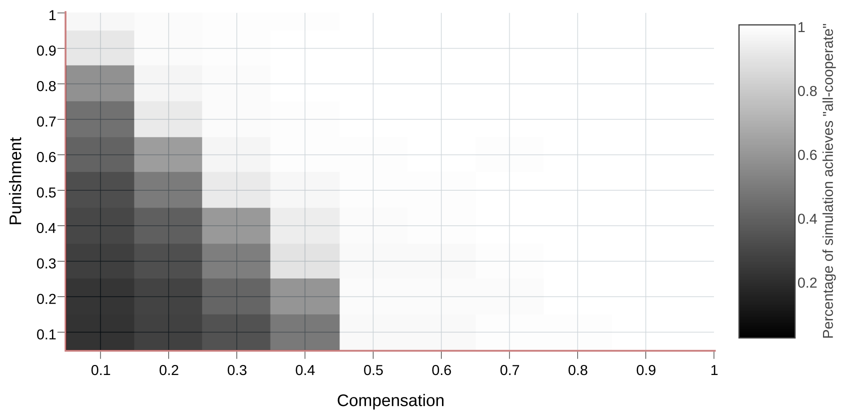


Fig. 12 Probability of reaching "all-cooperate" state under different combinations of punishment and compensation.

$$g + 2f > T + 2P - R - 2S + \frac{5e}{2} \tag{10}$$

Indeed, the original condition in equation 8 is a special case ($g = f$) of equation 10. Equation 10 provides more possibilities for a team to develop trust and cooperation. Equation 8 requires punishment g and is comparable to the cost of cheap talk. However, equation 10 shows that increasing compensation f may achieve the same outcome. These conditions are consistent with the above simulation results.

If a team does not want to implement punishment to *defect*, it may try to increase compensation. However, increasing compensation sometimes requires extra cost from the third party (who use some external resource to fund the extra compensation). Future research may focus on this issue. Modeling external compensation is beyond the scope of this paper.

B The Dictionaries Used in Classification

B.1 Lucene

No. Total Keywords = 66.
information retrieval, ranking, index caching, caches, bounded indices, language detecting(detection), token(ize), word break, Kuromoji, schema, encoding scheme, highlighter, distance measure(s), nightly, bug, issue, compatibility, configuration, JIRA, contention, build #[19], parse(r), test(s), benchmark, trunk, commit, merge policy, error, exception, regenerate, initialized, branch, intermittent, truncated, subprocess, inter-process, manifest file, real(-)time, load-sharing, load-balancing, proxy, main memory, concurrent, enumerate, logging, Java, boolean, comparator, decouple, synchronization, sequentialization, invalid, memory leak, JFlex, FNFE, parameters, character escaping, concatenation, stacked segments, discrepancy(ies), lexicon, unstored, vector, fuzzy query, collection api, wildcard.

B.2 Chromium OS

No. Total Keywords = 118.

[19] The word "build" is not necessary to be a keyword for detecting work-related message, but "build #" always refers a build failure.

B.2.1 Reused Keywords

bug, issue, compatibility, configuration, JIRA, contention, build #, parse(r), test(s), benchmark, trunk, commit, merge policy, error, exception, regenerate, initialized, branch, intermittent, truncated, subprocess, inter-process, manifest file, real(-)time, load-sharing, load-balancing, proxy, main memory, concurrent, enumerate, logging, Java, boolean, comparator, decouple, synchronization, sequentialization, invalid, memory leak, parameters, concatenation, stacked segments, wildcard.

B.2.2 New Keywords

keyboard layout, tree status, gizmo-paladin, boot, firmware, register, kernel, cmdline, native android, GNU/Linux distro, Valgrind test, vanilla linux, autotest, InitSDK, ChromeSDK, parallel, merge conflict(s), chroot, sysroot, syslinux, config param, libdevmapper, librpm, crbug.com, CQ, CL[20], git push, patch(es), deprecated, clang, x86-generic, amd64-generic, module-init, setup_board, resynced, driver, partition, mount, hash, SyncChrome, canary(ies), gpu, symbolizing, prefix, 0x...[21], buffer, userspace, device_tree, KVM, UI image(s), simplechrome, build_image, test image, dependencies, waterfall, code-review, developer(dev) mode, recovery mode, hard-reset, interpreter, use case(s), modeset, dev(-)server, xserver, flash image, ARM, libxml, rootfs, tryserver, Ubuntu, IO[22].

[20] CQ and CL must be capitalized.

[21] This refers any word start with "0x" which indicates a hexadecimal number. Most of them are used to represent a memory address.

[22] IO must be a separated word when considering as a keyword.