Social Psychology and Software Teams: Establishing Task-Effective Group Norms

Alvin Teh, Elisa Baniassad, Dirk van Rooy, and Clive Boughton,
Australian National University

// Group success relies on group norms. Norm manipulation can help groups perform software engineering tasks better. //

The major problems of our work are not so much technological as sociological in nature.[1] *—Tom DeMarco and Timothy Lister*

IN A RECENT *IEEE Software* article, Sallyann Freudenberg and Helen Sharp unveiled a burning question: what personalities constituted successful or failed agile teams?[2] This stems from the widely held belief that if we can find the right mix of individual personalities, we'll end up with a successful team. Intuitively, this makes sense. If group members are all introverted, they might be too shy to communicate well. If they're all type-A personalities, they might all clamor over one another to lead, and no one would follow any instructions. Ideally, teams should pick and mix personalities to get the right group.

Several approaches to the personality-combining strategy exist.[3,4] Basically, they identify team members' personality archetypes and then determine the most effective combination. But smart combining can be tricky. Although studies have identified traits and characteristics that can aid in building effective teams,[5] these traits often have little predictive value.[6]

Whether individuals "click" enough to effectively carry out a task is more than just a function of personality compatibility[7] or competency.[8,9] Group success relies on group norms, which are derived as much from the group's context as from the people in it.[10]

We performed a small, preliminary study of how norm manipulation affects groups performing requirements elicitation. The results show that the groups performed this task better when norms emphasized creativity rather than agreeability. More generally, our study suggests that norm manipulation might provide a practical way to enhance group performance in software engineering tasks.

Group Norms

Even in the most artificial and minimal conditions, groups will develop personalities or identities that override the individual members' personalities.[11] These group norms are similar to codes of conduct that are accepted by the group members. They regulate the members' behaviors, thoughts, and personality traits, and ultimately determine group communication, creativity, and productivity.

Norms can be imposed (top-down)

0740-7459/12/$31.00 © 2012 IEEE

UNHELPFUL GROUP NORMS AND COUNTERMEASURES

In *production blocking*, group members must take turns expressing their ideas. This leads to them expending energy to assert their turn, forgetting their ideas, or not listening to others while trying to rehearse presentation of their own ideas. One way to counteract this is the nominal-group technique, which separates individuals during the initial ideas phase and then brings them together to build on pooled ideas.

In *social loafing*, members self-regulate contributions because they perceive low group performance. This might be because they don't wish to outperform other members if they won't receive sufficient credit. You can counteract this by increasing member participation's visibility and accountability.

In *evaluation apprehension*, group members are unwilling to state their ideas for fear of negative judgments by the group. You can address this by convincing group members they're more expert than they thought. This belief has been shown to cause individuals to increase their contributions.

In *groupthink*, a group adopts a strong mindset against external influences, rejecting input contrary to their views. In extreme cases, the in-group excludes members who criticize their beliefs. This results in an "information vacuum": any conflicting input is rejected. You can counteract groupthink by instilling a *critical-thinking norm* that encourages group members to be less accepting of one another's views.

on a group as part of a larger organizational culture or through direct management. They can also arise organically (bottom-up) from social interaction between group members. In the latter case, the norms might differ considerably from those of nearby groups, even in the same organization. Usually, a mix of both top-down and bottom-up processes will give rise to an emerging group identity and its associated norms.

When norms enhance group effectiveness, communication flows well and individuals don't interfere with one another's progress. Other times, however, groups don't function properly, perhaps exhibiting any number of harmful behaviors. In these cases, productivity suffers and communication is ineffective.[12]

Conformity versus Individualism

Strong group cohesion is often good for group performance. Communication within a group tends to be much more extensive, and the group will act more as a whole than as separate individuals. Most important, cohesive groups are typically more successful in achieving their goals because members are more motivated to achieve them. There's also a socioemotional aspect: members of a highly cohesive group are more satisfied with that group, more willing to stay, and more likely to recommend the group to others.

Nevertheless, for some tasks, some degree of individualism rather than conformity is preferable. In particular, cohesion can pose limitations for groups seeking creative solutions. For instance, although brainstorming aims to generate a variety of ideas, evaluation apprehension often keeps people from expressing ideas perceived as too strange or unrealistic. (For more information on evaluation apprehension, see the sidebar.)

Dissent (even when wrong) can cause groups to think more divergently and ultimately to solve problems more creatively.[13] The resultant disharmony changes the group's dynamic and encourages the other members into more lateral thinking.

Any application of group norms must be preceded by an analysis of the task. Does it require creativity? Does it require a cohesive group working toward a clearly defined goal? Ultimately, the key is to have the right norm for the right task at the right time.

Encouraging Helpful Group Norms

Social psychology suggests a number of ways to encourage groups to develop task-appropriate norms. In an experimental setting, there are many options for coercing people to behave in certain ways. One researcher, in trying to instill the right communication pathways, put participants into cubicles and forced them to communicate through small slots. However, in the field, people don't appreciate the strong-arm approach. Instead, researchers can use less intrusive interventions:

- give simple, noninvasive instructions,
- place someone with a specific agenda into the group (like a mole), or
- ask groups to perform warm-up tasks (*priming tasks*) that promote the desired norms.

Tom Postmes and his colleagues, for instance, used priming to discourage groupthink (see the sidebar for more information on groupthink) by encouraging a *critical norm* (one that promotes critical thinking).[14] To estab-

lish the counteracting norm, Postmes and his colleagues gave special priming tasks to two sets of groups. They told the critical-norm set to judge a policy proposal and reach a common opinion. They told the other set (called the *consensus-norm* set) to create a poster that required input from all members, and the topic was open ("do anything").

The groups then performed their main tasks (which involved choosing a candidate for a university lectureship). During those tasks, the critical-norm set formed groups that exhibited a dynamic involving debate, tolerance of outlying opinions, and an attempt to promote objective understanding. The consensus-norm set, conversely, showed evidence of groupthink.

Norm manipulation not only regulates behavior in new groups but also nudges existing groups into a state in which members will more likely express sets of traits. Modifications to a group's norms resonate throughout the group, encouraging individuals to adjust their own characteristics to accommodate the new codes of behavior. For example, when a subordinate is promoted to a position equal to that of his or her supervisor, the norms of the relationship between individuals change: they must now communicate as peers. The apparent characteristics of the individuals involved could also change slightly, because the subordinate might become more confident in expressing assertiveness in certain situations where he or she used to defer to the supervisor.

Current social psychological research takes an interactionist (rather than reductionist) approach to the study of group processes. Rather than looking solely at how individual traits contribute to a group's functioning, the focus is on how the interaction between different levels of a group (that is, the individual and group levels) gives rise to a particular type of group norm.

The interactionist viewpoint is useful for dealing with norm manipulation. In our previous example, the individual reason for the subordinate's change might be unknown. The subordinate could have become more aggressive or might have been previously suppressing aggression owing to the prevailing social norms. However, the subordinate's starting personality type and underlying attitudes are less important (and less straightforward to analyze) than the relative change in his or her actions caused by the norm alteration.

Eliciting task-appropriate norms can align group members to a common goal, perhaps by overcoming initial individual differences. Essentially, the "right" norms encourage an optimal level of information sharing and, in general, encourage different behavior styles that enable a group to perform efficiently.

Group Norms in a Software Engineering Context

Although social psychology has repeatedly shown that you can nudge group norms to enhance creativity (by increasing individualism), we wanted to explore this nudging in a software engineering context. First, we examined whether we could use priming in a controlled setting (as Postmes did) to enhance effectiveness in a software engineering task requiring creative input.

As a starting point, we chose requirements elicitation (specifically, the facilitated-meetings format described in the *SWEBOK Guide*, chapter 2). From a psychological perspective, requirements elicitation can be seen as an information-sampling task—to develop an appropriate piece of software, stakeholders must collect and share critical information. Using existing social psychological approaches for these studies let us analyze requirements elicitation in terms of underlying, critical social psychological processes related to memory and communication.

To translate requirements elicitation to the laboratory, we created a model of the task. We would ask groups of participants to combine (from memory) prewritten requirements, which we deliberately made unique though overlapping across participants in each group, to synthesize a full set of system requirements. Participants would have to use their memories of what was important from their own perspectives and would have to be comfortable enough to force their requirements onto the group-created artifact. An optimal result would be a specification with many requirements, many of which were originally known only by individual members.

Before asking the groups to start their task, we would give them a short priming task aimed at forming either a critical or consensus norm. In accord with social psychology, we hypothesized that because information gathering and sharing was an important aspect of the task, the process would benefit from the critical norm.

Participants

The 21 participants were software engineers with one to five years of industry experience, enrolled in a master's program. This study ran within their requirements engineering course.

Step 1: Norm Manipulation through Priming Tasks

We randomly assigned the participants to groups of three, or *triads*. We then assigned these triads the critical or consensus norm.

Critical norm. We instructed these triads to debate and reach consensus on the statement, "Requirements specifications should always reflect design constraints." We told them, "To be successful in this task, the information contributed by all three members of the group needs to be evaluated critically."

Consensus norm. We instructed these triads to list what they had learned in their requirements-analysis course. Each

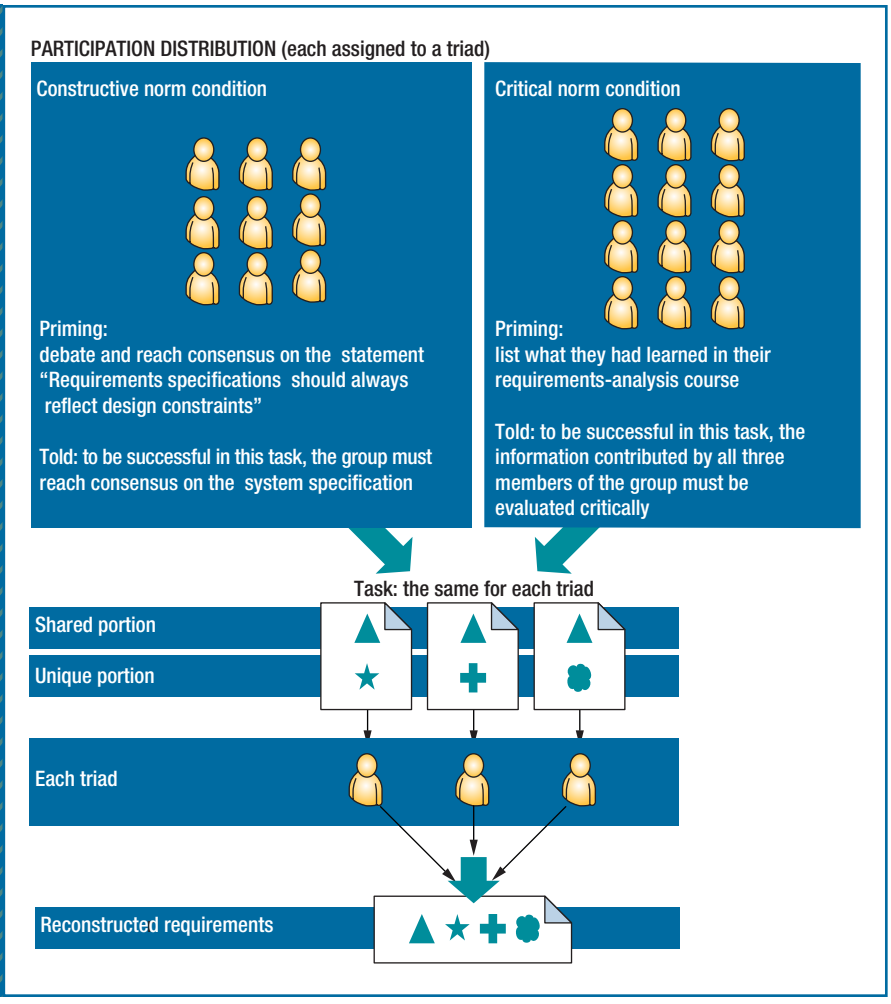

FIGURE 1. The norm manipulation study. We asked triads of engineers to synthesize requirements. Triangles indicate shared requirements; stars, crosses, or clouds indicate unshared requirements.

member was to contribute at least one item to a written list. We told them, "To be successful in this task, the group must reach consensus on the system specification."

Step 2: The Main Task

We assigned the same task to both groups. We presented each member of the triad with a set of specifications for a library management system with different types of users, each afforded different borrowing privileges (see Figure 1). Each participant received a different set of requirements:

- a shared basic block of requirements, consistent in every specification in the triad, and
- an unshared block, which pertained to either fines, destruction of worn books, or borrower treatment.

We arranged the source documents given to participants so that the unshared requirements affected one another and needed to be combined to create a full picture of the system. For instance, one triad member knew that different user classes had different fines but didn't know what the user classes were. Another triad member knew the different classes. Combining these two pieces of information would allow the group to write a more complete requirement—that certain classes of users had certain fines.

We gave the participants 15 minutes to silently study the specifications. Because we wanted to examine how our norm manipulation affects information retrieval and sharing, we didn't let participants take notes. Then, we took away the requirements and asked the triads to form a full set of requirements from memory. They didn't know that each member had distinct (but overlapping) sets of requirements. We also didn't tell them that they needed to use knowledge from one another to better contextualize their own view of the system.

Measures

We measured group performance in two ways. *Output* involved the number of correct requirements contributed; *content* involved the number of requirements included from the unshared-requirements block. A group producing many shared requirements but few unshared ones has high output but minimal content. Ideally, a group would show both high output and rich content. We considered contributions incorrect if they either weren't requirements or were semantically incorrect (for example, "superuser manages books" instead of "superuser manages user accounts").

We also conducted a short survey to determine what the perceived group cohesiveness was and how interesting participants found the task.[14]

Results

Figure 2 shows the study results. The critical-norm triads had, on average, more unique contributions than the consensus-norm triads. They also had more correct contributions overall, suggesting that this norm manipulation af-

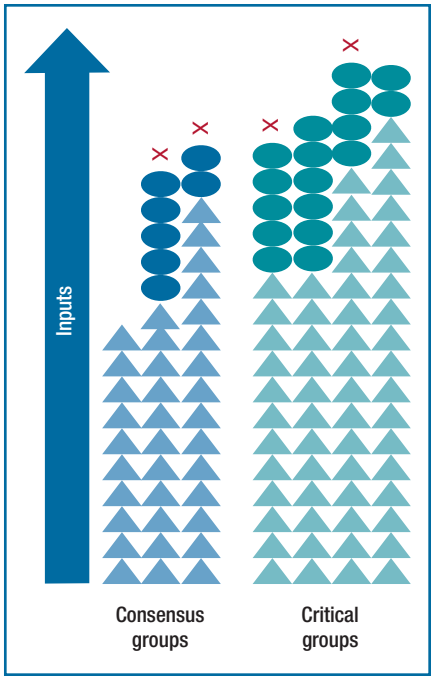

FIGURE 2. The study results. Triangles indicate shared requirements, ovals indicate unshared requirements, and Xs indicate incorrect contributions. The results suggest that manipulation toward a critical norm affected both output and content, positively affecting group performance.

TABLE 1

Bootstrapped point estimates and 90 percent confidence intervals for the median number of requirements in the critical and consensus conditions.

Type of requirements	Condition	Median	Limits of confidence interval (exclusive)	
			Lower	Upper
Unshared	Critical	14	12	16
	Consensus	11	10	11
Shared	Critical	4.5	2	5.5
	Consensus	2	0	2

fected both output and content, positively affecting group performance.

We used resampling to avoid some of the limitations of traditional statistical-significance testing.[15] Using BCa (bias corrected and accelerated) bootstrapping (with 99,999 resamples), we tested the median models for the number of critical and consensus requirements. Table 1 presents the point estimates and 90 percent confidence intervals around each coefficient. It shows minimal overlap in confidence intervals for the number of critical and consensus requirements. The sampling distributions show us that in 90 percent of the cases, the range of median values in the consensus condition would fall well below that of the critical condition.

Groups showed no corresponding change in perceived group norms. Critical and consensus triads felt equally cohesive and equally interested in and focused on the task. And, in both conditions, triads reported that their groups communicated freely. This isn't uncommon. Research shows that groups can become more critical in terms of asking more questions or being more attentive to details without becoming less friendly.[14]

Limitations and Future Directions

Requirements elicitation is critical to software development but is difficult and complex to study. Studies must find a balance between having a design that affords internal validity with sufficient control of critical independent variables and maintaining (as much as possible) realism and generalizability.

We focused heavily on internal validity, but we aimed to maximize generalizability by using the sample of software engineers from a graduate requirements engineering course. This setup gave us access to a small, yet representative sample and task, while letting us manipulate norms in a controlled way. We addressed our study's lack of power owing to the small sample by using resampling. We plan to confirm our results using larger samples.

As with many studies that emphasized how norm manipulation affects groups' ability to retrieve and share information, our study lacked a nominal control condition. This means that we didn't compare our groups' performances against summed individual performances. We plan to perform such a comparison.

Instead of asking what the personalities in successful teams are, social psychology suggests asking a slightly different question: How can we nudge groups into a productive state by cultivating appropriate group norms? This distinction is important. In practice, identifying the "right" personalities is difficult; even if we could, there's no guarantee that this would result in successful groups. Moreover, we rarely have the luxury of creating groups from scratch. Although our study was small and the results should be considered only indicative, our findings do align with evidence from the body of work in social psychology. Priming let us improve group creativity in-place, by nudging groups toward individualistic behavior.

Social psychology provides three suggestions to practitioners. First, work with whom you've got. Look more at norms than at individual team members' personality types or intelligence.

ABOUT THE AUTHORS

ALVIN TEH is a PhD student in software engineering at the Australian National University. His research interests include group dynamics in the context of software engineering groups and how to harness them for productivity gains. Teh has a BEng in software engineering from Curtin University of Technology. Contact him at alvin.teh@anu.edu.au.

ELISA BANIASSAD is a senior lecturer at the Australian National University's School of Computer Science. Her research interests include empirical studies of programmers at all stages of the software life cycle, and exploration of experimental programming models based on user-driven abstractions. Baniassad has a PhD in computer science from the University of British Columbia. Contact her at elisa.baniassad@anu.edu.au.

DIRK VAN ROOY is a lecturer at the Australian National University's School of Psychology. His research interests include computational models of social behavior, including connectionist and multiagent models. Van Rooy has a PhD in social and experimental psychology from the Free University of Brussels. Contact him at vanrooy@anu.edu.au.

CLIVE BOUGHTON is a senior lecturer at the Australian National University's School of Computer Science and the director of Software Improvements, a consulting company that produces software for large-scale organizations. Boughton has a PhD in physics from the Australian National University. Contact him at clive.boughton@anu.edu.au.

Second, nudge the teams. Use primes that are lightweight, unobtrusive, and transparently relevant to the main task. A priming activity can last as few as 15 minutes (10 minutes from Postmes's method and five more for instruction) for the prime to take hold and encourage the desired norm.

Finally, choose norms to suit the task. Consider whether your task will require diversity of thought or would benefit from agreement. When gathering requirements from a heterogeneous group of stakeholders, a priming task that maintains tolerance of outlying opinions and keeps people from converging too quickly can bring more elusive or uniquely held requirements into the open.

We're further investigating the effectiveness of nudging group dynamics in an industrial setting. By using this interactionist approach, we can focus on the relevant variables such as the nature of a task, the broader context in which the group operates, and how these variables interact to produce emergent group dynamics.

References

1. T. DeMarco and T. Lister, *Peopleware: Productive Projects and Teams*, Dorset House, 1999.
2. S. Freudenberg and H. Sharp, "The Top 10 Burning Research Questions from Practitioners," *IEEE Software*, vol. 27, no. 5, 2010, pp. 8–9.
3. M. Belbin, *Team Roles at Work*, Butterworth-Heinemann, 1993.
4. S. Nash, *Turning Team Performance Inside Out*, Davis-Black, 1999.
5. L.F. Capretz, "Personality Types in Software Engineering," *Int'l J. Human-Computer Studies*, vol. 58, no. 2, 2003, pp. 207–214.
6. I. Ajzen, "The Theory of Planned Behavior," *Organizational Behavior and Human Decision Processes*, vol. 50, no. 2, 1991, pp. 179–211.
7. N. Gorla and Y.W. Lam, "Who Should Work with Whom? Building Effective Software Project Teams," *Comm. ACM*, vol. 47, no. 6, 2004, pp. 79–82.
8. B.W. Boehm, *Software Engineering Economics*, Prentice Hall, 1981.
9. W. Scacchi, "Managing Software Engineering Projects: A Social Analysis," *IEEE Trans. Software Eng.*, vol. SE-10, no. 1, 1984, pp. 49–59.
10. S. McDonald and H.M. Edwards, "Who Should Test Whom?," *Comm. ACM*, vol. 50, no. 1, 2007, pp. 66–71.
11. H. Tajfel et al., "Social Categorization and Intergroup Behaviour," *European J. Social Psychology*, vol. 1, no. 2, 1977, pp. 149–177.
12. P.B. Paulus et al., "Social and Cognitive Influences in Group Brainstorming: Predicting Production Gains and Losses," *European Rev. Social Psychology*, vol. 12, no. 1, 2002, pp. 299–325.
13. C.J. Nemeth and J. Wachtler, "Creative Problem Solving as a Result of Majority vs. Minority Influence," *European J. Social Psychology*, vol. 13, no. 1, 1983, pp. 45–55.
14. T. Postmes, R. Spears, and S. Cihangir, "Quality of Decision Making and Group Norms," *J. Personality and Social Psychology*, vol. 80, no. 6, 2001, pp. 918–930.
15. B. Efron, "Bootstrap Confidence Intervals for a Class of Parametric Problems," *Biometrika*, vol. 72, no. 1, 1985, pp. 45–58.