Brainstorming Under Constraints: Why Software Developers Brainstorm in Groups

Patrick C. Shih
Department of Informatics
University of California, Irvine
Irvine, CA 92697-3440 USA
patshih@ics.uci.edu

Gina Venolia
Microsoft Research
One Microsoft Way
Redmond, WA, 98052 USA
gina.venolia@microsoft.com

Gary M. Olson
Department of Informatics
University of California, Irvine
Irvine, CA 92697-3440 USA
gary.olson@uci.edu

Group brainstorming is widely adopted as a design method in the domain of software development. However, existing brainstorming literature has consistently proven group brainstorming to be ineffective under the controlled laboratory settings. Yet, electronic brainstorming systems informed by the results of these prior laboratory studies have failed to gain adoption in the field because of the lack of support for group well-being and member support. Therefore, there is a need to better understand brainstorming in the field. In this work, we seek to understand why and how brainstorming is actually practiced, rather than how brainstorming practices deviate from formal brainstorming rules, by observing brainstorming meetings at Microsoft. The results of this work show that, contrary to the conventional brainstorming practices, software teams at Microsoft engage heavily in the constraint discovery process in their brainstorming meetings. We identified two types of constraints that occur in brainstorming meetings. Functional constraints are requirements and criteria that define the idea space, whereas practical constraints are limitations that prioritize the proposed solutions.

Brainstorming, Idea Generation, Creativity, Problem-solving, Decision-making

1. INTRODUCTION

Idea Generation has been a topic of creativity research for centuries. Brainstorming, in particular, has grown to become synonymous with idea generation. Brainstorming was developed in the 1930s by Alex Osborn, an advertising executive who co-founded the world's largest advertising agency network. Osborn (1953) first coined this process "brainstorming" and published it in his book, *Applied Imagination*. Contrary to decision-making techniques that aim to systematically eliminate unsuitable ideas to reach a final consensus, the brainstorming process focuses on gathering as many ideas as possible. The four brainstorming rules are:

(i) Focus on quantity
(ii) Withhold criticism
(iii) Welcome unusual ideas
(iv) Combine and improve ideas

In his initial description of brainstorming, Osborn claimed that by reducing the amount of criticism from self and others during this creative process, a group of individuals could produce better results in terms of quantity and quality.

In the past half-century, a plethora of studies focused on enhancing the brainstorming process have been published. Surprisingly, research has shown that group brainstorming is not as effective as brainstorming separately as individuals. Diehl and Stroebe (1987) reviewed brainstorming productivity losses and laid out five of the most common hindrances of the group brainstorming process:

- *Evaluation apprehension*: "working in a group makes one's contributions visible to others, and despite the usual brainstorming instructions not to evaluate others' ideas, the members of a group can still be reticent to contribute their ideas."
- *Free riding*: "individual members of a group might not expend the effort since other members of the group are contributing ideas."
- *Limited air time*: "when only one person can speak at a time, there is limited time for each individual to contribute."
- *Production blocking*: "at each moment only one line of ideas is being generated, since they are reported serially; groups will therefore tend to pursue fewer different kinds of ideas."

- *Cognitive inertia*: "because of limited air time, individuals often have to hold on to their contributions until they get a chance to report them, and as a result they might forget them, or they might decide not to offer them; in either case, the act of holding on to them will prevent them from thinking of other ideas."

Alternative methods have been developed to overcome these hindrances. The nominal group technique (NGT) is one effective method in which each member is given time to brainstorm alone without communicating with other members. The individually generated ideas are later pooled together in a merged list. This process essentially eliminates idea evaluation and criticism during the idea generation process; some researchers choose to call this "deferred judgment" because idea selection is postponed until a later stage of the process (Grossman, 1984). Dennis and Valacich (1993) reported that in more than 50 laboratory studies, groups employing NGT generated a far greater number of creative ideas than groups brainstorming while communicating with each other (also known as interactive groups). Other reviews of brainstorming research also found brainstorming using NGT to consistently produce better ideas in terms of quality and quantity (*e.g.*, Barki & Pinsonneault, 2001; Mullen, Johnson, & Salas, 1991). However, NGT can lack the social and collaborative aspects of brainstorming as a group exercise because it prevents participants from actively engaging and building upon group ideas. Many studies reported that participants have more positive reactions in interactive group sessions than in nominal groups, including overall satisfaction and perceptions of group effectiveness (*e.g.*, Mullen, Johnson, & Salas, 1991; Stroebe, Diehl, & Abakoumkin, 1992). For this reason, despite the obvious performance gains under the nominal group settings, practitioners continue to brainstorm in interactive groups. Some researchers choose to call this common notion, the "illusion of group productivity" (Pinsonneault *et al.*, 1999; Stroebe, Diehl, & Abakoumkin, 1992).

Subsequently, researchers have considered how collaborative technologies such as electronic meeting systems and group decision support systems might influence the brainstorming process. These electronic brainstorming systems (EBSs) often incorporate benefits of NGT by supporting anonymous and parallel input (*e.g.*, Connolly, Jessup, & Valacich, 1990), which eliminate both evaluation apprehension and production blocking (Diehl & Stroebe, 1987). Usage of an EBS typically involves multiple users entering ideas anonymously into a central repository in parallel. The ideas stored in the central repository are selected at random and presented to the participants at specific intervals or upon request in order to trigger new ideas. A review of prior studies by Barki and Pinsonneault (2001) reported that EBS groups produced more, as well as better ideas, than conventional interactive brainstorming groups. However, existing empirical evidence that compares performance of EBS and nominal brainstorming groups remains inconclusive. Despite the increased idea performance experienced in laboratory settings, most of these solutions have not been widely adopted because they have failed at providing group well-being, such as reinforcement of group culture and socializing activities, and member support, which involves establishing personal status by demonstrating expertise and building knowledge networks (Dennis & Reinicke, 2004). For this reason, brainstorming in interactive groups without the help of EBSs continues to dominate brainstorming practices in both corporate and academic environments (Dennis & Reinicke, 2004).

In software engineering, brainstorming is often taught and used in the context of a design methodology (Robinson, 2004). Studies have shown that brainstorming is often used in the informal and early-stage idea generation phase of software design (e.g., Wu, Graham, & Smith, 2003) and has been incorporated into requirement engineering and agile software development methodology as a standard idea generation technique (Nuseibeh & Easterbrook, 2000; Paetsch, Eberlein, & Maurer, 2003). In fact, some researchers claimed that brainstorming and its variations have become the "definitive basic method for finding ideas" in software design (Koberg & Bagnall, 1974). Given the prevalence of brainstorming practices in software design methodology, the proposed work aims to characterize brainstorming practices in the domain of software development. The resulting findings can then be used to provide design recommendations that are both feasible and adoptable for brainstorming groups in real world settings. First, we discuss the motivation and research questions for this study. Then, we present work that is relevant to brainstorming in context. We then describe the research site, Microsoft Corporation, and the employees that use brainstorming to complete their tasks. After, we discuss the data collection and analysis methods. We then discuss preliminary findings from the data analysis. We highlight the key contributions of this research. Finally, we discuss future work that would further enhance the research.

2. MOTIVATION AND RESEARCH QUESTIONS

Most of the past attempts on improving brainstorming productivity by inducing process and

technology modifications have been shown to be successful in laboratory studies but have failed to gain acceptance in practice. To date, interactive group brainstorming in face-to-face settings continues to be the preferred method. Researchers attributed the failures to the lack of support for group well-being and member support (Dennis & Reinicke, 2004). The lack of support for social needs resulted in lowered overall satisfaction and perceptions of group effectiveness when using the EBS systems (Mullen, Johnson, & Salas, 1991; Stroebe, Diehl, & Abakoumkin, 1992). However, besides the social benefits, little is known about how brainstorming is appropriated and adapted to fit a variety of different contexts. There have not been many studies that focus on detailing why and how brainstorming is *actually* practiced, rather than how brainstorming practices deviate from the prescribed rules. Furthermore, research has shown that people who produced the fewest ideas produced about the same number of outstanding and high quality ideas as the people who produced the most ideas (Briggs, Reinig, & Shepherd, 1997). Therefore, there appears to be more than just the sheer performance measurements such as quantity and quality at stake when practitioners view brainstorming efficiency. Assessing the underlying intention and understanding brainstorming practices in real settings is essential for developing brainstorming productivity metrics that matter to practitioners.

Since brainstorming is a technique that is designed to support the inherent needs of generating better ideas, we want to understand idea generation from the perspective of its practitioners. This work focuses on a naturalistic field study of brainstorming practice in the domain of software engineering. The observational study attempts to address the following research questions:

- Why do software teams continue to brainstorm in groups? Is it because they strive for idea quantity, quality, or a combination of the two? Are there other external factors that are not typically measured by the traditional brainstorming productivity metrics? If so, what contextual requirements do software teams fulfill in the group brainstorming sessions?
- How do software teams brainstorm? Do software teams follow the brainstorming rules in their brainstorming sessions, or do they practice brainstorming differently?
- Given that the software industry is one that focuses on technological innovation and that the employees tend to be early adopters of technology, why is there not a collaborative brainstorming tool that has been successfully developed and adopted? Are there technology or process changes to the brainstorming process that would be helpful for this population?

These questions can lead to understanding the factors that influence brainstorming behaviors and best practices in organizational settings. By taking the different goals, expectations, and norms of users and those in the organizational ecosystem into account, we are hoping to discover the underlying values of organizational brainstorming practices that are essential for making idea generation more effective in different parts of the software development cycle.

3. RELATED WORK

As the "illusion of group productivity" continues to puzzle brainstorming researchers, many have attempted to resolve the mystery in laboratory experiments. Researchers have uncovered important social benefits such as active social interaction, criteria negotiation, and social comparison—the act of being exposed to a high number of ideas and to common ideas, and found that they enhanced the generation of additional ideas (*e.g.*, Dugosh & Paulus, 2005; Shepherd *et al*., 1995). Rietzschel *et al.* (2006) questioned the effectiveness of reducing evaluation apprehension by postponing idea evaluation and compared the differences between quality and quantity of ideas generated in interactive and nominal groups. Their results indicated that without actively engaging in group discussions—though nominal groups are capable of generating higher quantity ideas that are more original—the ideas were generally less feasible than those generated by interactive groups. More importantly, the final decisions made from ideas generated in both nominal and interactive groups were of equal quality. Therefore, higher idea quantity resulting from following strict brainstorming rules is not sufficient to lead to better solutions, and groups may choose to continue brainstorming in interactive groups because the inconvenience brought by the imposed process changes does not outweigh the perceived performance gain by the group. While these prior studies can be used to explain why brainstorming practitioners choose to brainstorm in groups than alone, the performance metrics still focus strictly on idea quantity and quality. Other hidden yet dominant factors that continue to drive groups to brainstorm interactively still remain to be uncovered.

In the organizational context, the word "brainstorming" has been appropriated to fit a variety of different idea generation needs (Isaksen, 1998). Sutton and Hargadon (1996) published an influential study on the brainstorming practices at IDEO, an internationally renowned product design firm. Deviating from the quantity- and quality-centric

views, they exposed the organizational, social, and economical dimensions that were not evident in traditional brainstorming literature. Brainstorming groups at IDEO did not care about the productivity losses. Instead, brainstorming acted as an "idea theater" that satisfied other practical needs such as supporting the organizational memory and impressing clients when it was practiced within IDEO's organizational context.

Olson *et al.* (1992) conducted a series of studies on how software designers carried out their design meetings. They found that teams spent about a third of their time on idea clarification – both orchestrating and sharing expertise among group members. Further research has shown that creative thinking in different design domains is predominantly characterized by their different patterns of criteria constraints and the methods designers employ to manage these constraints (Stacey & Eckert, 2009). In the domain of software development, software designers face a different set of contextual requirements from those encountered at a product design firm, a controlled research laboratory, or an advertising agency. The requirements are different among the various software teams across different parts of the software development process cycles. In order to better support their idea generation process, a thorough understanding of the intricate details of the brainstorming sessions at different software development stages is paramount.

4. FIELD STUDY

4.1 Research Site

Microsoft is one of the world's largest software development organizations. We selected Microsoft's headquarter campus in Redmond, WA as our field study site because that is where the majority of the research and development groups are located. Approximately half of Microsoft's 90,000 employees are in the Puget Sound region. The workforce is spread across over 100 buildings designated by product lines and job functionalities. Since the goal of this work was to provide a contextualized account of activities in brainstorming meetings at different software development phases, Microsoft Corporation was the ideal venue of investigation.

4.1 Software Development Cycle

Software teams at Microsoft typically consist of members with the following job roles: program manager (PM), software development engineer (SDE), and software development engineer in test (SDET). A standard software development cycle involves the following stages:

(i) Planning
   a. PMs gather information about product features and draft specs for them.
   b. PMs, SDEs, and SDETs meet and discuss the features, make changes, and get general agreement on the specs.
   c. SDEs define the architecture plan for implementing the features.
   d. SDETs define the object models to the features and describe them in test specs.
   e. PMs, SDEs, and SDETs meet and discuss the feature specs, architecture plan, and test specs, and agree that the specs are accurate representations of the features.

(ii) Implementation
   a. SDEs begin implementing the features.
   b. SDETs implement the object models, and then start writing test cases against them.
   c. As features become available, SDEs and SDETs run tests together against the features.

(iii) Stabilization
   a. SDEs fix bugs while SDETs continue to analyze test failures.

(iv) Future Planning
   a. PMs, SDEs, and SDETs meet and discuss about customer feedback, requirement changes, feature support changes, and bugs.

4.3 Brainstorming Occurrences

In order to understand the status of brainstorming usage at Microsoft's Redmond campus, we sent out a short survey to 100 randomly selected software developers asking for their brainstorming practices. We received 42 responses, a 42% response rate. The results indicated that a majority of the software teams have participated in brainstorming meetings (95.2%). Our interviews also revealed that most of the software developers have received formal training on the brainstorming process as described by Osborn (1953). The software developers also noted that brainstorming meetings served a very particular purpose and were unique from other early-stage activities such as design and storyboarding meetings.

As Table 1 shows, although brainstorming is considered to be an early-stage activity that typically occurs in the beginning of the software development cycle, the activity actually takes place across all software development stages.

Table 1: *Brainstorming Occurrences across Different Stages of Software Development*

Planning	*89.5%*
Implementation	*42.1%*
Stabilization	*36.8%*
Future Planning	*23.7%*

In terms of frequency, over half of the developers brainstorm on a weekly basis, and over three quarters of the developers participate in brainstorming meetings at least once a month (Table 2).

Table 2: *Frequency of the Brainstorming Meetings*

Daily	*21.1%*
Weekly	*34.2%*
Monthly	*21.1%*
Yearly	*10.5%*
Every Milestone	*23.7%*
As Needed	*13.2%*

5. RESEARCH METHODS

5.1 Data Collection

Initial informal interviews and email exchanges were used to build relationships with the software teams and to understand the basic brainstorming practices at Microsoft. Knowledge learned from the informal discussions was used to identify groups that are representative of the software development process. We conducted an in-depth case study by selecting one team from each of the software development phases and observing each team complete a brainstorming topic in its entirety over time:

- *Planning*. PMs meet daily over a span of a week to brainstorm about feature specs
- *Implementation*. SDEs meet biweekly over a span of 8 weeks to brainstorm about an architecture plan for a security update platform.
- *Stabilization*. SDETs meet once per milestone to brainstorm about possible causes and solutions to software bugs and strategies for supporting products on different operating systems.

Overall, we observed about 20 hours of brainstorming sessions. All meetings were video and audio recorded. Informal follow-up interviews were conducted to clarify intentions, job specifications, and other technical details.

5.2 Data Analysis

Both quantitative and qualitative methods were used in analyzing the collected data. All recorded meetings were transcribed. The transcripts were first coded using the grounded theory approach to see if any general themes emerged from the brainstorming conversations (Strauss & Corbin, 1998). We then compared the emerged categories with existing literature in order to correlate and cross-validate our findings. We found that the Issues-Alternatives-Criteria coding scheme developed by Olson *et al.* (1992) for analyzing coordination in design meetings of software development teams had many overlaps with our meeting coordination categories, and that the creative process stages described by Amabile (1996) best described our task categories. Therefore, we constructed a coding scheme based on our analysis while borrowing the language used in prior literature. The coded results were used to compare brainstorming activities across different software development stages. The coding scheme is described below (Table 3).

Table 3: *Frequency of the Brainstorming Meetings*

Theme	**Coding Category**	**Examples of Activity**
Task-focused	*Identifying Problem*	*Raising an issue, Clarifying goals, Discussing agenda items*
	Gathering Information	*Looking up specs*
	Generating Idea	*Proposing alternatives*
	Evaluation	*Criteria, Constraints, Limitations*
Coordination-focused	*Logistics*	*Project Management, Meeting Management*
	Recap and Scenarios	*Summary, Walkthrough*
	Miscellaneous	*Digression, Other*

The coding scheme is separated into two major categories: activities that are task focused and activities that are geared toward group coordination. Considerable theoretical work (*e.g.*, Amabile, 1996; Stein, 1953) has suggested that the creative process involves several stages, including: (1) Identifying a problem/opportunity, (2) Gathering information or resources, (3) Generating ideas, and (4) Evaluating, modifying, and communicating ideas. Therefore, the task-focused activities mimic those different creative process stages:

- *Identifying a problem/opportunity*. This includes agendas, goals, problems, and action items that are the focus of the brainstorming meeting.
- *Gathering information or resources*. This includes looking for requirements and background information that are essential in understanding and resolving raised issues.
- *Generating ideas*. This includes proposing solutions and alternatives.
- *Evaluating, modifying, and communicating ideas*. This includes evaluating whether the proposed idea is suitable for the issue at hand.

The group coordination focused activities contain the following categories:

- *Logistics*. This includes scheduling, delegating tasks, keeping time, and other activities not directly related to the brainstorming topic.
- *Recap and Scenarios*. This includes restating and summarizing the current ideas, as well as scenario walkthroughs in order to make sure all meeting participants are on the same page
- *Miscellaneous*. This includes digressions, jokes, and other activities that are not relevant to the categories listed above.

In order to check for coding reliability, a 30-minute session was coded by two researchers. We achieved high degrees of reliability in our coding (Cohen's Kappa = 0.75). The agreed coding scheme was then applied to the rest of the brainstorming sessions. We describe the findings in the sections below.

6. RESULTS AND DISCUSSIONS

6.1 Time Spent on Activities

The coded conversations are broken down into the following categories listed in Table 4. Overall, software teams spent the least amount of time gathering information about the brainstorming topic. This is due to the fact that most members of the team are domain experts and specialists who already have prior knowledge and experience in dealing with the issues under investigation. In their brainstorming meetings, time spent on gathering information typically involved searching for existing emails, agenda list, and feature specs. Those were then projected onto the projector screen in the conference room. However, the projected information was only meant to be used as a reference to guide the discussion. If a piece of information was deemed to be important, it was usually forwarded to other team members following the meeting. Therefore, the conversation in the brainstorming meeting tended to be fluid, and software team members often responded to raised issues without the need to look up additional information.

The category that had the most disparity across the software development phases was problem identification. This was because of the nature of the tasks that PMs, SDEs, and SDETs face at different phases of the development process. In the planning phase, PMs arrived at the meeting knowing the topic of focus, thus they spent more time on building scenarios and use cases when they generated ideas. In the implementation phase, SDEs broke down the features into multiple problem spaces and attempted to tackle each problem one after another by walking through the necessary components in order to make sure that the proposed architecture accommodated all project requirements. In the stabilization phase, SDETs went down a long list of bugs and support issues and discussed potential solutions for them.

Accounting for over a third of the brainstorming meetings, evaluation stood as the dominant category of activities across all three phases of software development. These preliminary findings suggest that on top of the two primary decision-making phases—idea generation and idea selection—the brainstorming participants spent over a third of their time on "constraint discovery". This was in clear violation of the brainstorming rules that demand participants to postpone evaluation until after the idea generation phase. In fact, teams only spent about 12% to 20% of their time generating ideas. Since the participants all received formal training on the brainstorming process, two questions remain to be addressed: (1) what drove them to conduct brainstorming in this fashion, and (2) why do they continue to call the deviated idea generation process "brainstorming"? We will address the first question in the remaining sections of this proposal and delegate the second question to future work.

Table 4: *Time Spent in Meetings*

	Planning	**Implementation**	**Stabilization**
Identifying Problem	*1.8%*	*7.5%*	*14.8%*
Gathering Information	*2.6%*	*4.4%*	*5.5%*
Idea Generation	*14.5%*	*19.4%*	*12.3%*
Evaluation	*39.3%*	*29.7%*	*44.8%*
Logistics	*21.5%*	*11.9%*	*12.2%*
Recap and Scenarios	*17.6%*	*20%*	*2.7%*
Misc.	*15%*	*7.1%*	*7.7%*

6.2 Constraint Discovery

In our observations, software teams spent the majority of their time seeking clarifications that would more clearly define the problem space, identify resource limitations, and discover constraints that had not previously been discussed. These constraints came in a variety of different flavors. In order to understand the activities coded under the evaluation category, we separated the constraints into "functional constraints" and "practical constraints" and discuss them in more detail below.

6.2.1. Functional Constraints
The functional constraints included domain expertise, past experience, and tacit knowledge about the topic. When a question was raised in the brainstorming meeting:

- A domain expert could shed light on the capabilities or limitations of a certain application platform.
- A developer who had prior experience dealing with a specific API could provide insight on the applicability of the API on the current feature set.
- A program manager who had been informed on a requirement modification could update the group on last minute changes.

Functional constraints are the bread-and-butter of a successful brainstorming meeting. Without them, there would be no way to shape the brainstorming discussions.

6.2.2. Practical Constraints
The practical constraints are limiting constraints that are associated with the lack of human resources, funding, time, or stakeholder interests. These factors are typically used to discount a series of potential ideas that may be useful but is not of primary concern to the group due to more pressing issues.

6.3 Examples of Brainstorming Practices

6.3.1. Functional Constraint Example (Implementation)
In this episode, a team of SDEs were trying to decide whether a top-down or bottom-up way of retrieving relevant system data objects was more suitable for the problem. The two software developers discussed about why the existing process of combining both approaches was applicable instead of going with solely a single approach.

> P1: Well, we know that there are two ways of filling data up and usually we start with bottom-up and then go top-down and that's how it works right now. So, we can stick to that. I don't see a problem with that process.
>
> P2: Right. Another thing is that certain things, you don't have any choice. For certain things you need to do bottom-up.
>
> P1: Correct. Exactly. So…
>
> P2: Otherwise you're not going to be able to create – no matter when you do it. So, even if you – let's say you create all this skeleton and then you don't have the information, you need to come back here again, even if you say that I'm doing top-down, you need to come back here, populate this and then populate this here. It's a bottom-up again. So, certain things you don't have any choice.

P1 suggested keeping the same data fill up process because he did not see any need for changes. P2 further justified with a functional constraint by walking through examples in which both methods must be used in order to satisfy all cases of data.

6.3.2. Functional Constraint Example (Stabilization)

Several SDETs were brainstorming about whether they were able to support automated test cases on different browser environments. An SDET, P3, suggested utilizing the adapter infrastructure, but it was deemed unfeasible for the Safari browser because Apple does not provide adequate support documentation for extension development on Safari. Here, the domain experts who had previously encountered similar issues, P4 and P5, contributed by sharing development limitations about the browser. The functional constraints were necessary in keeping the ideas within the possible ways of implementation as restricted by limitations imposed by technology.

> P3: Something we can do is an adapter and develop [product name]. Because I think there's a couple of other [platform name] browsers and we may be able to just get them all with one solution.
>
> P4: But I don't think that we have a way to hook up. We can't automate that browser, so that's a problem. For [competing product name 1] and [competing product name 2], we know how to talk across processes. For this guy we don't, right?
>
> P5: Right. [competing company name] doesn't do anything like that. They're all [technology name] and they don't talk to anything about technology. That's where we're at.

6.3.3. Practical Constraint Example (Planning)
Two PMs were discussing about the fidelity of the prototype. P8 suggested creating a high visual fidelity prototype in order to generate better

feedback, but it was shut down because of the existing time pressure.

> P6: And no visual.
>
> P7: I would not do visuals at this point.
>
> P6: Even though I feel like it's a help {...} Usually I even get more or better feedback on visuals.
>
> P7: I agree. And we have to keep in mind what this is here is to kind of get us a quick feedback of which direction seems to be the most plausible. We're not necessarily trying to get feedback that we're going to directly incorporate into the scenario itself. If we're lucky we can, but we have a tight deadline, and so it's more about validating direction than it is trying to glean new insights.

P7 acknowledged that while a visual prototype was a superior choice in generating more concrete feedback, but an impending deadline, a practical constraint, was keeping them at bay and was forcing them to settle with a quicker and less time consuming approach that would hopefully generate enough feedback for their deliverables.

6.3.4. Practical Constraint Example (Stabilization)
A group of SDETs engaged in a discussion about whether to remove test support to the feature of a product. Accompanying the discussion, P11 mentioned the need to remove even more test support on other features because there were not enough resources to fulfill the existing assignments. The practical constraint of not having enough manpower to support all the proposed features was the driving force for prioritization.

> P8: So, we can remove our custom proxy now?
>
> P9: Um-hum.
>
> P10: It has been removed. With the [product name] conversion. We tossed that out.
>
> P11: So, one thing that – sorry, we're jumping to a different topic, but one thing I feel that we need to think is, if we always do additive – things that add, right? Like we were saying, we need to do this, we need to provide that. We need to add one more component. I think that one thing we need to look into is, how do we trim down what we have and support? Like, what – it's clear that – the resources that we have are bounded.

7. SUMMARY OF EMPIRICAL RESULTS

As illustrated from the field data, brainstorming in the context of software development is about discovering constraints—drawing the proper boundaries to ground the ideas. However, the high dependency on using constraints to shape the idea generation process contradicts with the fundamental notion of brainstorming, which is to reduce evaluation apprehension by postponing idea evaluation until a later stage. The contradiction between the prescribed brainstorming rules and actual deployment in practice is particularly challenging for brainstorming practitioners. One developer commented that:

> The most difficult part most of the time seems to be keeping the discussion within the realm of true brainstorming. As far as I understand, heavy discussions of implementation, ramifications, etc. of an idea fall outside the bounds. On our team, at least, we tend to think of an idea, then discuss its feasibility for a bit, then maybe come up with another idea, then discuss for a bit. It seems difficult to keep the idea generation the focus.

This divide can be explained by more succinctly examining the role of constraints in brainstorming meetings. Due to the complex nature of software development, most large-scale software projects are faced with overly constrained problems. When members of a software team meet in a brainstorming meeting, each member brings a view of non-overlapping constraints that the project must satisfy. The primary purpose of the brainstorming meeting in the context of software development at Microsoft is essentially to identify all possible constraints and to find the right balance to satisfy the most important ones. When asked about his brainstorming experience, one program manager made the following remark:

> Coming up with ideas is not hard, but being able to discover the proper constraints that help shape those ideas and implementing the most beneficial ideas is the difficult part.

Although both are important from the business operation perspective, functional and practical constraints play different roles in shaping ideas. Functional constraints are value-neutral constraints that are necessary for bounding the idea within the desirable context. On the other hand, practical constraints such as time and resources that must be met in order to ensure the survival of a business hinder the creativity of the problem space and room for trial and exploration. Therefore, the brainstorming process could use a helpful amendment that would require brainstorming groups to state their functional constraints upfront without having to worry about the practical constraints. Furthermore, providing a communication channel to negotiate criteria and incentives in evaluating idea quality as a group in the idea generation process could benefit the group decision-making process as a whole.

Researchers on other creativity processes have also started to analyze the role of constraints in early-stage design meetings (Onarheim &

Wiltschnig, 2010; Stacey & Eckert, 2009), and our findings contribute to the existing literature. Overall, our findings provide the following contributions to a relatively scarce body of brainstorming research in the field:

- From the theory perspective, the study generates empirical results that could enhance the current idea generation model by accounting for constraint discovery, which is technically forbidden by formal brainstorming rules.
- From the system perspective, the design recommendations generated from the findings could result in innovative collaborative brainstorming systems. Rather than simply allowing the participants to input their ideas, systems should provide a platform to scaffold the constraint discovery process.
- From the experimental design perspective, the findings could generate more naturalistic tasks that incorporate constraint components to better emulate real world scenarios. The constructed tasks would allow experimental research in the laboratory setting to more accurately measure and predict brainstorming's impact.

8. LIMITATIONS AND FUTURE WORKS

The results of this work identified two types of constraints that occur in brainstorming meetings. Functional constraints are requirements and criteria that define the idea space, whereas practical constraints are limitations that prioritize the proposed solutions. However, there still remains a significant amount of work to be done before these preliminary findings can fully represent the general brainstorming practices. The field study was conducted at Microsoft, which is known to be peculiar in many of its software practices (Cusumano & Selby, 1998), and may not share the same design methodologies, development practices, and culture with other large software companies, smaller start-ups, or open-source communities.

Two eminent questions immediately follow the results of our study. First, why do software teams continue to call these meetings "brainstorming" even if they choose not to follow the conventional brainstorming rules? Is it simply a matter of convenience? This could potentially be a deeper issue, especially because the software teams that we have observed were all familiar with the conventional brainstorming rules. It is important to unpack the meaning so we can more effectively design for the purpose of brainstorming meetings and better incorporate the findings into future brainstorming models and systems. We plan to develop a survey that could be used to validate the empirical findings based on observations of the software teams at Microsoft. Second, can the brainstorming model generated based on the software development practices at Microsoft be generalized to other software communities? We will survey other software organizations in order to verify these findings on a more general level.

9. ACKNOWLEDGEMENTS

This research was supported by the Microsoft Corporation and the Donald Bren Foundation. The authors are very grateful to the software teams that participated in this study, especially given the sensitive nature of early-stage ideation meetings.

10. REFERENCES

Amabile, T.M. (1996) Creativity In Context: Update To The Social Psychology Of Creativity. Westview Press.

Barki, H. and Pinsonneault, A. (2001) Small Group Brainstorming and Idea Quality: Is Electronic Brainstorming the Most Effective Approach? Small Group Research 32, 2, 158-205.

Briggs, R.O., Reinig, B.A., and Shepherd, M.M. (1997) Quality as a Function of Quantity in Electronic Brainstorming. IEEE Comp. Society, 94.

Connolly, T., Jessup, L.M., and Valacich, J.S. (1990) Effects of anonymity and evaluative tone on idea generation in computer-mediated groups. Manage. Sci. 36, 6, 689-703.

Cusumano, M.A. and Selby, R.W. (1998) Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People. Free Press.

Delbecq, A.L., Ven, A.H.V.D., and Gustafson, D.H. (1975) Group Techniques for Program Planning: A Guide to Nominal Group and Delphi Processes. Scott, Foresman, and Company.

Dennis, A.R. and Reinicke, B. (2004) Beta vs. VHS and the Acceptance of Electronic Brainstorming Technology. MIS Quarterly 28(1), 1-20.

Dennis, A., Valacich, J., and Nunamaker, J. (1990) An experimental investigation of the effects of group size in an electronic meeting environment. Systems, Man and Cybernetics, IEEE Transactions on 20, 5, 1049-1057.

Dennis, A.R. and Valacich, J.S. (1993) Computer brainstorms: More heads are better than one. Journal of Applied Psychology. Vol. 78(4) 78, 4, 531-537.

Diehl, M. and Stroebe, W. (1987) Productivity loss in brainstorming groups: Toward the solution of a riddle. Journal of Personality and Social Psychology. Vol. 53(3) 53, 3, 497-509.

Dugosh, K.L. and Paulus, P.B. (2005) Cognitive and social comparison processes in brainstorming. Journal of Experimental Social Psychology 41, 3, 313-320.

Gallupe, R.B., Dennis, A.R., Cooper, W.H., Valacich, J.S., Bastianutti, L.M., and Nunamaker, J.F. (1992) Electronic Brainstorming and Group Size. The Academy of Management Journal 35, 2, 350-369.

Grossman, S.R. (1984) Brainstorming Updated. Training & Development Journal 38, 2, 84.

Isaksen, S.G. (1998) A review of brainstorming research: Six critical issues for inquiry. Creativity Research Unit-Monograph 302.

Koberg, D. and Bagnall, J. (1974) The Universal Traveler: a Soft-systems Guide: To Creativity, Problem-solving, and the Process of Design. W. Kaufmann.

Mullen, B., Johnson, C., and Salas, E. (1991) Productivity Loss in Brainstorming Groups: A Meta-Analytic Integration. Basic and Applied Social Psychology 12, 1, 3.

Nuseibeh, B. and Easterbrook, S. (2000) Requirements engineering: a roadmap. Proceedings of the Conference on The Future of Software Engineering, ACM, 35–46.

Olson, G.M., Olson, J.S., Carter, M.R., and Storrøsten, M. (1992) Small group design meetings: an analysis of collaboration. Human-Computer Interaction 7, 4, 347-374.

Onarheim, B. and Wiltschnig, S. Opening and constraining: constraints and their role in creative processes. Proceedings of the 1st DESIRE Network Conference on Creativity and Innovation in Design, Desire Network (2010), 83–89.

Osborn, A.F. (1953) Applied Imagination. Charles Scribner's Sons.

Paetsch, F., Eberlein, A., and Maurer, F. (2003) Requirements engineering and agile software development. Proceedings of Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 308-313.

Pinsonneault, A., Barki, H., Gallupe, R.B., and Hoppen, N. (1999) Electronic Brainstorming: The Illusion of Productivity. Information Systems Research 10, 2, 110-133.

Rietzschel, E.F., Nijstad, B.A., and Stroebe, W. (2006) Productivity is not enough: A comparison of interactive and nominal brainstorming groups on idea generation and selection. Journal of Experimental Social Psychology 42, 2, 244-251.

Robinson, J.A. (2004) Software design for engineers and scientists. Elsevier.

Shepherd, M.M., Briggs, R.O., Reinig, B.A., Yen, J., and Nunamaker, J. (1995) Invoking Social Comparison to Improve Electronic Brainstorming: Beyond Anonymity. Journal of Management Information Systems 12, 3, 155-170.

Sosik, J.J. (1997) Effects of Transformational Leadership and Anonymity on Idea Generation in Computer-Mediated Groups. Group Organization Management 22, 4, 460-487.

Stacey, M.K. and Eckert, C.M. (2009) Reshaping the Box: Creative designing as constraint management. International Journal of Product Development.

Stein, M.I. Creativity and Culture. (1953) The Journal of Psychology: Interdisciplinary and Applied 36, 2, 311.

Strauss, A.C. and Corbin, J. (1998) Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications, Inc, Thousand Oaks, CA, USA.

Stroebe, W., Diehl, M., and Abakoumkin, G. (1992) The Illusion of Group Effectivity. Pers Soc Psychol Bull 18, 5, 643-650.

Sutton, R.I. and Hargadon, A. (1996) Brainstorming Groups in Context: Effectiveness in a Product Design Firm. Administrative Science Quarterly 41, 4, 685-718.

Valacich, J.S., Mennecke, B., Wachter, R., and Wheeler, B. (1993) Computer-mediated idea generation: the effects of group size and group heterogeneity. Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences, 152-160 vol.4.

Valacich, J.S., George, J.F., Nunamaker, J.F., and Vogel, D.R. (1994) Physical Proximity Effects on Computer-Mediated Group Idea Generation. Small Group Research 25, 1, 83-104.

Wu, J., Graham, T., and Smith, P. W. (2003) A Study of Collaboration in Software Design. In Proceedings of the 2003 International Symposium on Empirical Software Engineering, 304-313.