Experiences Gamifying Developer Adoption of Practices and Tools

Will Snipes
ABB Corporate Research
Industrial Software Systems
Raleigh, NC USA
will.snipes@us.abb.com

Anil R. Nair
ABB Corporate Research
Industrial Software Systems
Bangalore, India
anil.nair@in.abb.com

Emerson Murphy-Hill
NC State University
Computer Science
Raleigh, N.C. USA
emerson@csc.ncsu.edu

ABSTRACT

As software development practices evolve, toolsmiths face the continuous challenge of getting developers to adopt new practices and tools. We tested an idea with industrial software developers that adding game-like feedback to the development environment would improve adoption of tools and practices for code navigation. We present results from a pre-study survey of 130 developers' opinions on gamification and motivation, usage data from a study with an intact team of six developers of a game on code navigation practices, and feedback collected in post-study interviews. Our pre-study survey showed that most developers were interested in gamification, though some have strong negative opinions. Study results show that two of the six study developers adjusted their practices when presented with competitive game elements.

Categories and Subject Descriptors

D.2.9 [**Software**]: Software EngineeringManagement [Productivity]

General Terms

Management, Economics, Human computer interaction

Keywords

Development, Productivity, Code Navigation

1. INTRODUCTION

Sascha is a developer with years of experience maintaining code, who knows how to navigate through code fairly well. Sascha heard about some other navigation tools, but kept using the built-in features of the Visual Studio Integrated Development Environment (IDE) mainly out of habit. When Sascha's company started a game around code navigation in Visual Studio, he became interested in being a part of the fun. Sascha tried the code navigation tools and practices that were part of the game and soon found himself at the top of the game's leaderboard. After participating in the game, Sascha used the new practices more often and felt recognized for his achievement.

Toolsmiths and researchers are continuously finding ways to improve the work of software developers. However, as new practices and tools become available, developers like Sascha may not be motivated to start using them unless they interact with a peer developer who promotes them [14]. Of course they are intrinsically motivated to improve, but urgent demands interfere with experimenting using new tools and they become less efficient than otherwise possible. Thus, we need to both communicate with and motivate developers like Sascha to start using new tools.

One option for motivating people to adopt new practices is to apply elements of game mechanics or "gamify" the practice. Gamification is the incorporation of game elements into an activity that people do not typically consider a game [1]. We envisioned a tool in our prior work that would motivate developers by providing constant feedback and allowing them to compare their improvements to how their peers are doing [22]. This tool, called Blaze, creates a competitive game with continuous feedback on how well the developer adopts the improved practice. The more they change their practices to adopt new techniques or tools, the higher they score in the game.

Would software developers writing serious software be motivated by this gamification concept? To answer this question, this work contributes an assessment of developers' response to game-like elements tied to their use of new tools and practices. We assess 130 developers' response and relevant concerns about the idea of gamification through a web-based survey. As a research study, we piloted the Blaze tool that introduces game elements focused on improving navigation practices to developers using Visual Studio. Developers' reactions to gamification were mostly positive with a few detractors, and results of the study show that game elements influence some developers, but have no effect on others' use of tools and practices.

The rest of this paper is organized as follows: Section 2 reviews results from the pre-study survey validating the acceptance of gamification with developers, Section 3 contains the design considerations for developing the Blaze tool/game, Section 4 gives the study design for attempting to influence developers' work patterns with Blaze, Section 5 results from the study and post-study survey, Section 6 covers related

ICSE '14, May 31 – June 7, 2014, Hyderabad, India

work, Section 7 discusses threats to validity, and Section 8 has our conclusions.

2. PRE-STUDY SURVEY

2.1 Pre-study Background

We conducted an assessment survey questionnaire to determine whether the developer community in ABB would be receptive to a gamification approach to software development. ABB is a multi-national corporation and established leader in power and automation technologies that enable utility and industry customers to improve performance while lowering environmental impact. The developer survey received over 130 responses from software developers at ABB.

2.2 Pre-study Survey Results

A consulting firm, Saatchi and Saatchi, use a question in a survey that is often cited in gamification initiatives. This study reuses this question [18] tailoring it to software development by asking "How interested they would be in working for a company that incorporated some aspects of games into software engineering tools as a way to increase productivity in the workplace"? Results are segmented by country in Figure 1 and found gamification of software engineering is more interesting to developers in India, Poland, and Finland than in other major countries. The responses indicate 74% of developers in India, Poland, and Finland were at least somewhat interested, while the US, Sweden, and Switzerland had slightly lower interest in applying gamification to their work.

Comments from participants reflected a more interesting controversial reaction than these measurements might otherwise indicate. A developer from India said "Gamification is a juvenile way of getting programmers to do something that you want." A different respondent expressed a general concern that "Gamification of best practices can have nasty side effects." Another participant said "there is already high a level of competition between software development teams." indicating they are under considerable pressured to get the work done. Another developer reacted with a statement that expresses a positive intrinsic motivation "I am using practices and tools because I want to do my job well',' indicating that gamification was unnecessary to motivate people to improve.

On the positive side, a developer from Poland said, "In a previous work place we had a 'game' between developers - scoring broken/unbroken builds and new/passed/failed tests. It was helping to improve quality and it worked well. [It is] recommended especially for junior developers. " Other comments indicated competition or points are acceptable "as long as it doesn't interrupt or slow down my work. "

The Figure 1 compares the overall response from our survey to the Saatchi and Saatchi survey [18] upon which this question was based. Our results land on both sides of the 75% mark reported by the Saatchi and Saatchi study.

We asked developers how likely they would be to try tools and practices recommended to them through an automated usage tracking system. Answers showed 95% of the developers surveyed are likely to try the recommended tools and practices. Thus, a recommendation system would positively impact the deployment of software engineering tools and practices at ABB.

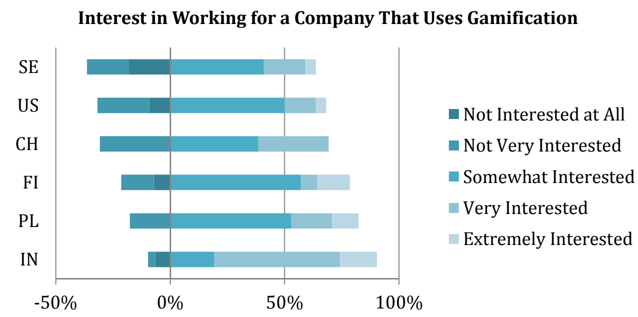

Figure 1: Acceptance of gamification

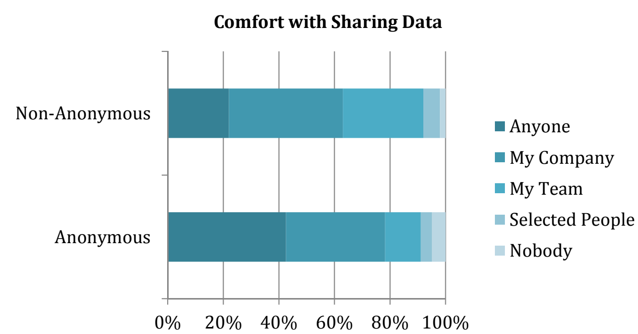

Figure 2: Comfort with sharing usage information

Another question was whether sharing detailed usage information with colleagues and the company as a whole would be a concern for developers. The question provided a graduated scale for the scope of who data was shared with and divided the questions between whether sharing was anonymous or not. Figure 2 shows, as the community with which the data is shared gets smaller, more developer are comfortable with sharing the data. If we take scope as the main concept, and limit the sharing to "My Team", we can say that 90% of respondents are comfortable with sharing either anonymous or non-anonymous data if the scope of data were limited to "My Team". Developers were less comfortable sharing with anyone (the category for people outside ABB) particularly if the data was not kept anonymous. The first conclusion from these questions on sharing data is that people are very willing to share information that could help their team. Second, sharing within the company is acceptable with anonymized data.

We asked developers what would motivate them to use tools and practices and gave them four choices to rank. In Figure 3, we see that 75% of respondents ranked coworker recommendation in 1st or 2nd place. Of several comments, a developer in Sweden captured the following scene, "[The] most motivating [thing] is when a new practice / tool is discussed during a continuous improvement meeting or even coffee break, where one co-developer tells about the benefits and there is a joint discussion on where it (tool or practice) could be best applied."

The second ranked motivator for using tools and practices, "Team Goal", shows the survey response of "It elevated my team's achievement level in a game among software engi-

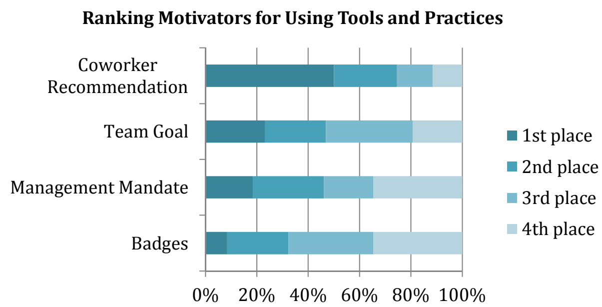

Figure 3: Motivation for using tools and practices

neering teams at ABB." This response indicated that 47% of developers would be motivated to use new tools and practices if that helped their team in a competition between development teams.

Just behind team competition is management mandate where developers would be asked to follow specific practices by the organization. The lowest was having badges posted on their social network profile where 32% ranked this in 1st or 2nd place. Badges and management mandate received the most negative written comments in the survey. A developer from the United States captured the lower opinion of badges that several participants expressed "I am motivated because ABB allows me to choose the tools that best work for me. A badge will not really do anything for me."

The respones on this question, supported by findings in [4, 13, 14], influenced the development of the studied game to include a competitive leaderboard and a common recommendation contest to influence ABB developers to change their practices.

In another question, developers ranked several actions that they would like to receive awards for. Overall 68% of developers ranked receiving an award for consistently performing quality practices as either 1st or 2nd place. Thus, providing awards to developers for using good practices would motivate developers to do them consistently.

To summarize, the high-level conclusions from each key question are:

- 95% of developers responding would try tools and practices suggested by an automated recommendation system
- Developers are motivated by collaboration and team goals more than mandates and individual awards
- Awards for good quality development practices ranked highest
- Over 90% of respondents approved of sharing data with their team members

Thus, this pre-study survey provided a positive indicator for going forward with a software engineering tool focused on sharing developer data and rewarding developers for using good practices and tools.

3. GAME DESIGN

Learning from the pre-study findings, we implemented a game-like tool for developers to use in Visual Studio. The tool focused on motivating the adoption of good practices by including elements of games in the development environment. This section describes the selection of a practice suitable for gamification, considerations for designing the game, and competitive elements in the game.

3.1 Select a Practice

We defined the first criteria for selecting a practice based on prior work applying gamification to other domains outside software development. In their paper on game design patterns, Hamari et al. stated that assigning achievements to required tasks reduces intrinsic motivation because player autonomy is reduced by the achievements [6]. Thus we avoided selecting practices like bug fixing or task completion because they are simply requirements of the job.

Another criteria imposed by the instrument limits us to practices performed entirely in the IDE so they can be monitored using Blaze. Potential practices for gamification include test-driven development, refactoring tool use, debugging practices, navigation practices, eliminating static analysis bugs, and frequent configuration management submissions.

Frequent configuration management submission and static analysis bug elimination are highlighted in this paper's related work section with prior gamification studies in classroom environments. Johnson and Kou achieved automated monitoring of test-driven development practices with Zorro [8] thus may provide a candidate for future gamification application. Refactoring tool use also has studies that make it a good candidate for gamification [11, 15].

Studies of developer effectiveness by Robillard et al. [17] (discussed in Related Work 6) identify structured navigation as a practice more effective developers use when maintaining programs. Structured navigation is the use of navigation commands and tools that follow or represent the structure of the code. Using structured navigation provides a shortcut that confirms the existence of and affirms the mental model the developer has of the code. Robillard et al.'s experiment [17] showed developers who maintain mental models and navigate structurally are more effective than other developers when given the same maintenance task and code artifacts. Structured navigation fits the requirements of a practice contained in the IDE, shows benefits, and is not specifically assigned task for a developer. Thus we selected structured navigation as the practice to demonstrate gamification.

3.2 Design the Game

To setup the game, the Blaze tool provides an XML configuration file where the researcher can configure Blaze to categorize and assign points to the commands that are part of a software engineering practice. Blaze allows the researcher to define multiple command category levels. Thus we can categorize commands as navigation then further classify them into structured and unstructured navigation.

To evaluate whether developers improved their practices for structured navigation we established a metric, Navigation ratio, as the number of structured navigation events in a session or period of time over the number of unstructured navigation events. Our hypothesis is that instant feedback from points combined with game information would result in an increase in the navigation ratio for developers in the study.

Structured navigation events included:

- Navigate To (Ctrl+,) is a fuzzy search interface that lists identifiers matching the selected string
- Go To Definition (F12) brings up the code that defines the selected identifier
- View Call Hierarchy (Ctrl+K Ctrl+T) provides a two way analysis of an identifier's dependencies and uses
- Class View (Ctrl+W, C) provides a browser and search function for classes and class hierarchy
- Find All References provides a list of lines that reference an identifier
- Navigate to Event Handler in the XAML editor shows the event handler for an object
- View Class Diagram generates a class diagram
- View Object Browser is a search tool and browser

Unstructured navigation events included selecting a file in an explorer window or selecting the tab for a file, using arrow and page up/down keys to go up/down through a file, scrolling, clicking on a file element, and using any of the built-in find commands such as "Find in Files" or "Quick Find".

To evaluate developers' use of recommended tools in Visual Studio, we categorized them by their tool name. As part of the study, we wanted developers to learn about and adopt the Sando search tool. Sando is part of the recommended developer tool suite for ABB and supports structured navigation using code search [19].

Gamification practices consider the feedback of achievements and points as critical to a successful outcome. The design of achievement awards should gradually encourage the participant to reach higher and higher levels of success in the activity. Achievements must feel earned to the participant so they recognize the effort to get the award was significant and feel satisfied [6]. Hamari and Eranti describe a general relationship between achievements in games and regular game play. Achievements in games typically create a parallel scoring system to the main game play. They make the game more engaging by providing multiple ways to increase your score and multiple challenges in one interface.

Considering these guidelines, we assigned one point for each use of structured navigation commands while unstructured navigation commands received zero points. We gave additional emphasis to using the Sando search tool by assigning ten points for each use. To create levels following the guidance from Hamari [6], we applied an exponential curve based on points scored. Users initially "level up" after working a day or two; the next level may require a week's work to achieve as the difficulty increases. We designed the points and levels so above average developers can pass all levels during the study period.

By selecting an intact team for the study, we hope to leverage aspects of motivation such as the Leaderboard to spur feelings of competition between teammates. The leaderboard allows developers to see their own score in relation to the top five people on the team. However, the other participants' identifiers are auto-generated as three capital letters assigned based on their position on the leaderboard. Not knowing who of your colleagues was ahead of you could reduce motivation, but this avoids one potential source of conflict in an intact team.

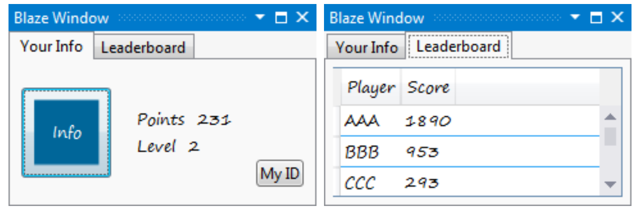

Figure 4: Blaze Tool Window

4. STUDY DESIGN

Utilizing the continuous monitoring capabilities of Blaze, we designed the study to conduct a longitudinal study of the effect of Blaze on developers' navigation practices. The Blaze tool revealed information about navigation in three stages of one week duration each. A post-study survey gathered developers' opinions of Blaze and some confounding factors that could influence results.

4.1 Intervention Staging

The key research objective was to determine whether we can drive adoption of a practice through gamification. As previously discussed, we designed the game in Blaze to influence developers towards using more structured navigation practices. We constructed the interventions during the study to roll out in stages each week they use the tool. This allowed us to establish a baseline and determine the changes as the study continued.

In the initial stage, participants were informed of the purpose of the study, and what to expect from the Blaze tool. The participants installed the tool in their Visual Studio environment. During this stage, Blaze did not present a window for them to see and simply collected data in the background.

After a week's data collection marking the end of the first stage, Blaze started to automatically pop-up a window containing a button for a web page with information. The page included a link to download Sando and information about using Sando in a video demo. The sub-page on navigation contained a page of tutorial on the built-in structured navigation commands in Visual Studio. The "About Blaze" page contained details of the point scoring system and background on Blaze's purpose. The usefulness of the communication site is rated as part of our post-study survey.

In the third stage, developers received instant feedback from Blaze on the points they accumulated during the development session. The tool's appearance at this stage is shown in Figure 4. The "Info" button provides a level indicator via colored contents and activates a chart display window when clicked. Points shows the points accumulated for the developer's history. Level is the level of the game achieved based on an exponentially growing curve. The developer could click on the Leaderboard tab to see how their scores compared with the others in the study. This final configuration remained as the display until the end of the study.

4.2 Study Participants

As mentioned in Figure 1 from our survey, the acceptance of gamification is highest in India. Hence, we conducted this study with an intact team of 6 developers working in an R&D facility in India on a large industrial software system.

Initially, we met the whole team along with the management staff to explain the research project and the Blaze tool we developed. We explained the aim of the study, what data is collected, and how the data would be processed and managed.

It was important to gain the confidence of the participants that their data were kept confidential so their activities would be as close as possible to real-world. If the participants felt like they are being tracked and the data would identify them to their peers or managers, then they may refuse to participate or deviate from their normal development style.

We assured the team about the confidentiality of the data, and the management reassured them that the data collected by the research group will not be used for any other purpose other than the study. Developers were asked to volunteer for the study by their management, however, managers did not know who participated in the study. Their names and data were kept confidential to ensure that no one apart from the research team knows he/she is participating in the study. To distinguish individuals in the data each user is assigned a generated unique ID. Developers could chose to share the unique ID with us or keep it confidential. All were willing to share their unique ID with us during the follow-up feedback sessions.

Management wished to have no knowledge of who was participating and took great care to remain detached from the study proceedings. Data at this level of detail has not been collected in ABB development organizations. Researchers were careful to create the system where developers controlled whether we could associate their identity with their data, and fortunately management bought into the necessity for participant anonymity.

4.3 Study Survey

The goal for the study follow-up survey was to collect feedback on the tool and the effect it had on the developers' navigation practices and knowledge. The survey questions were segmented into portions that evaluate prior and post knowledge of navigation practices, key steps in the study itself, the influence of game elements on their navigation practices, typical tasks and demographic information.

To find out whether Blaze increased developers' knowledge of structured navigation commands, we asked them to rate their knowledge of structured navigation commands prior to the study on a 1-10 scale. Then we asked them to contrast their knowledge after the study to assess what they learned.

The other key question is did Blaze have a perceived effect on their navigation practices. Here too we asked them to rate how much they used structured navigation prior to the study using a 4 choice rating of "not at all", "some", "many times", and "as much as possible". Then we asked whether they used structured navigation commands "about the same amount", "more", or "a lot more" during the study. These two questions assessed their perceptions of structured navigation command use. We compared the answers with the usage data collected from Blaze.

The final assessment questions tested whether using Blaze influenced their attention to navigation commands by asking how much specific game elements influenced their practices. The responses available were "not at all", "a little", "some", and "a lot".

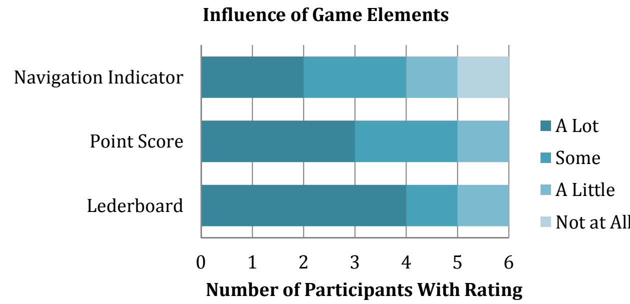

Figure 5: Influence of Feedback Elements

The other questions included whether they reviewed the training materials on structured navigation commands, installed the Sando code search tool, and what types of development tasks they would consider using structured navigation for in general. Another demographic question asked how well they know the code they work in rated from "I work in code that I wrote", "I maintain code that I am familiar with", to "I recently learned the code".

5. RESULTS

5.1 Study Survey Results

We conducted an in-person post study survey in order to cross check the quantitative data from Blaze with developers' perception of their practices and their demographic information.

The study survey queried developers' relevant demographic data to help understand some differences in results. We asked developers to provide their years of experience so we could evaluate any effects experience might have. The developers in the study had between 7 and 14 years experience. One developer who dropped out of the study said their lower level of experience compared to their peers influenced their decision to stop participating.

We asked developers to rate their knowledge of the code they maintain. Four of the six developers reported they maintain code that they wrote, while the two most experienced developers maintain code that they recently learned. One developer reported during our individual feedback sessions that working in code they know well affects their navigation practices leading them to use more unstructured navigation.

On the assessment of prior structured navigation knowledge, half of the participants replied they had very good prior knowledge about structured navigation and they use structured navigation "as much as possible". The other half of the participants rated their prior knowledge of structured navigation as good and reported that they use it "some" of the time. Of the three developers who rated their prior knowledge and use of structured navigation higher, two of them had the highest point scores of the group.

The chart in figure 5 shows how the individual feedback features influenced developers to think about using structured navigation. Four out of six developers said the leaderboard influenced them "a lot", and all rated the leaderboard as having at least "a little" influence. Three out of six said their individual point score influenced them "a lot", and all said the score had at least "a little" influence. The element with mixed ratings was a graphical indicator on whether they were improving over past use of structured navigation commands. The indicator received the influencing "a lot" rating from two participants and a "not at all" influence rating from one participant with the rest having at least "a little" influence. The results indicate the more obvious and more clearly comparative information in the leaderboard was the strongest influence for developers.

The participants requested that Blaze provide more feedback on tips to help them increase their score. They indicated that more obvious feedback, such as hints that pop-up when they launch the tool, would help them more than passively provided information. Their desire for personal metrics and historical views of the data encourages us that this detailed level of measurement and analysis is perceived as helpful. Features to meet these requests are part of future work.

5.2 Study Data Observations

Blaze logs events from actions the developer takes or actions from Visual Studio itself. The event is basically a GUI event managed with Visual Studio by event handlers registered with the application to listen for the event. Blaze becomes a global event handler listening for all events by registering for them in Visual Studio.

Blaze records key attributes along with each event in visual studio. The log captures a time-stamp for each event normalized to GMT. The name of the event provides a low-level classification. The event type reflects an internal classification in Blaze. An optional field for Artifact Reference captures data such as the file-name being edited and the currently selected line number. An anonymized unique identifier for each Blaze user allows us to investigate differences between developers.

To assist in log analysis, Blaze classifies the events into categories for related activities such as navigating, editing, building, debugging, testing, and using a known tool. Each event name maps to a category from the list and sub-division of the navigation category separates structured navigation from unstructured navigation.

In another processing step, Blaze aggregates events along the time-line into sessions. A session, similar to episodes defined by [9], is an abstract concept grouping multiple events into a sequence with a beginning and an end. Of the many possible ways to define sessions, we chose to define edit sessions to contain all the events that transpire between edits of different files. When the developer edits a file different from the file last edited, all the events after the last edit to the previous file are included in the new session for the new file. For example, the developer edits file A, then navigates through files B and C finally landing at line 100 of file D where they make another edit. All the events between the edit of file A and the edit of file D are considered part of the edit session ending with the edit of file D. When this edit is followed by multiple edits to file D, the additional edits are included in the same session ending with the edits to file D. Events transpiring between edits may involve activities not considered as edit related such as building and testing. This session definition allows us to quantify how much time in each category of events elapses before the developer edits another area of code. Although not at the method level, this way of generating sessions is similar to how Robillard et al. generated sessions based on located methods for their study [17].

5.2.1 *Observations About Drivers of Edit Session Duration*

We selected navigation as the subject of the Blaze study because it was identified by Robillard et al. [17] as correlated with developers who effectively completed a maintenance task. In this analysis, we ask whether we are focusing on the right area of developers' activity in Visual Studio.

In order to test whether navigation is an important area to address, we construct a hypothesis that using unstructured navigation is an important factor in developer productivity. Edit session duration, we assert by opinion, is a factor related to developer productivity though there are many other factors we could consider. With the data available from Blaze, we can explore the categories of events correlated with the duration of edit sessions over the study period.

Using Weka [5], we performed an attribute selection process to identify the most significant attributes related to edit session duration. The attribute search method we chose was Greedy Step-wise search, which selects attributes based on their correlation with the target, and stops adding attributes when the next attribute decreases the correlation. Defining the attributes as the sum of developer time spent by category in each edit session, the top 5 out of the 15 possible categories ranked by this process are as follows:

1. Unstructured navigation
2. Debug - events from running the debugger
3. Edit - events that modify the code
4. Other Actions - other events in visual studio that we do not sub-classify
5. Build - events from running a build

To determine the correlation with edit duration, we used R [16] to construct a linear regression model for unstructured navigation. The linear model found a positive correlation with unstructured navigation with a p-value $<.001$ on 1694 degrees of freedom. The model had an Adjusted R-squared of 0.64 showing a good portion of the variation in edit duration is explained by the unstructured navigation category.

This analysis shows we cannot disprove that unstructured navigation is correlated with edit duration, thus we conclude it is an important factor in developer productivity.

5.2.2 *Observations on Navigation Ratio*

During the study, we staged the deployment of information feedback to developers so they would first have access to learn about structured navigation and then get instant feedback in the form of points for using structured navigation commands.

To evaluate whether developers use more structured navigation when they receive instant feedback, we monitor the change in navigation ratio over the study period. We expected developers learning about navigation commands and tools that count towards the score would begin using the practices more during the second week. In the third week, we turn on points feedback and the leaderboard expecting this will encourage developers to use more structured navigation.

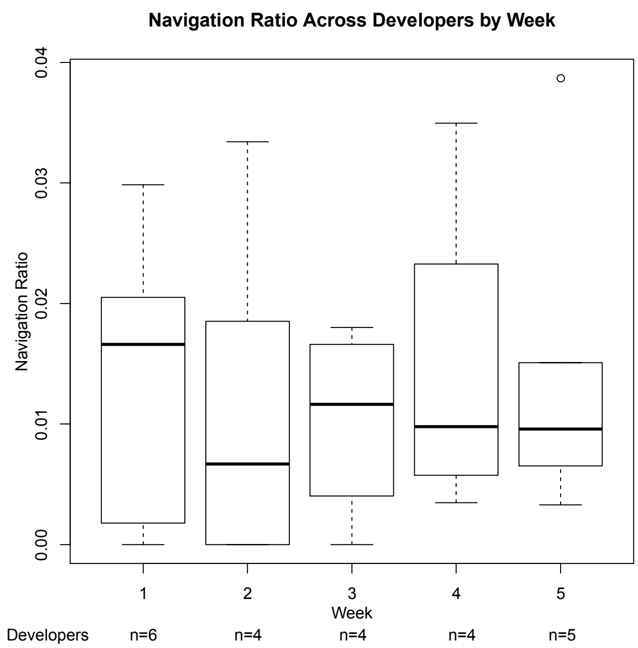

Figure 6: Navigation Ratio by Week

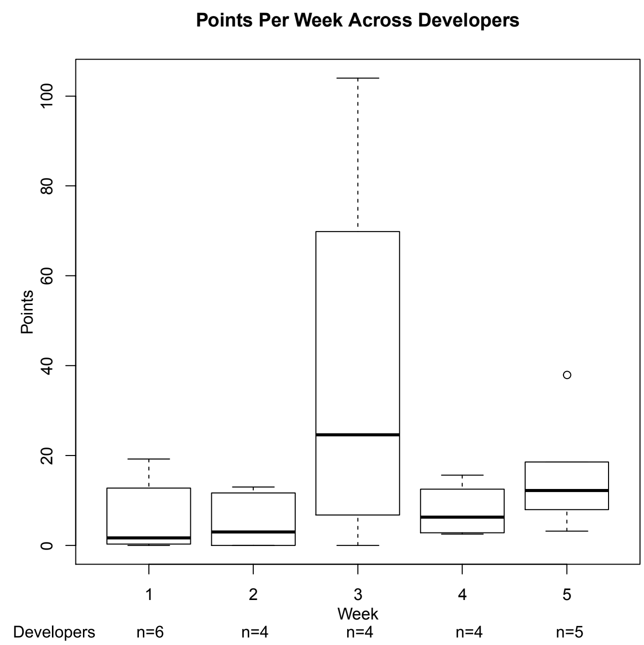

Figure 7: Average Points per Day by Week

We observe visually in Figure 6, that the navigation ratio did not increase significantly from the beginning to the end of the study. We evaluated the week over week differences using a Wilcoxon Rank sum test [7] comparing week one to week two and week one to week three. In week two, when the developers were able to learn about structured navigation commands, there was no significant change. Also in week two, two developers were not providing data. One was working in another environment and returned to the study, while the other dropped for the rest of the study. In week three there was a slight increase in the mean, however the increase is not significant. Weeks four and five follow the formal period of study show the navigation ratio continues in a similar range. Overall, we did not see a significant effect towards increasing navigation ratio for the interventions Blaze provided to the environment.

5.2.3 Observations on Points

The developers using Blaze received feedback in the form of points given for using structured navigation commands and using the Sando search tool [19]. Per the study design, the points display was disabled until the third week following installation to establish the effect of instant feedback apart from other information. The points display and the navigation arrow were enabled in the third week to test whether the instant feedback was successful in driving increased use of the targeted features. During the first two weeks shown in Figure 7, developers were fairly consistent in their use patterns. With the introduction of the points feedback, two developers' points accumulation spiked in week three due mostly to increased use of Sando. We informed developers at the beginning of the study that Sando usage would receive ten points and using built-in structured navigation commands would receive one point.

In these results, we see an effect from the points display in Blaze in week three when the developers could see their points feedback for the first time. We conducted a Wilcoxon Rank sum test on points similar to the way we did for navigation ratio to determine significance. Across developers, the difference is not significant at the .05 level. The p-value for week two against week three is .19, and for week one against week three it is .31. We do know from feedback discussions and usage data that two developers took notice of the points and used Sando more in the third week. Their use of Sando continued at a lower level in weeks four and five. The effect of points feedback is greater than navigation ratio during week three because of the additional emphasis of the ten point bonus given for Sando usage. Results from our post-study survey where at least half the developers said the points and leaderboard influenced them "a lot" also supports our conclusion that two developers were influenced by the game elements in Blaze.

5.2.4 Post-Study Feedback Discussions

Following our data analysis, we conducted one-on-one feedback sessions with the participants. Before initiating the discussion, we assured the individual developer that the data shared with them is confidential and we will not share the same with management. The participants received the feedback in the form of one-on-one presentations of quantitative charts and written analysis of the information. The key quantitative chart shown in Figure 8 compared their distribution of time in each category to the median for the group. The categories were identified based on what would be most relevant to the developer where some categories match findings in the Weka analysis in section 5.2.1. The "Actions" category identified by Weka is omitted in the chart because it is a collection of events not categorized at a level useful for developers. The category for structured navigation shows the practice we are trying to encourage as the opposite of the more common unstructured navigation practice.

We discovered during these feedback sessions that devel-

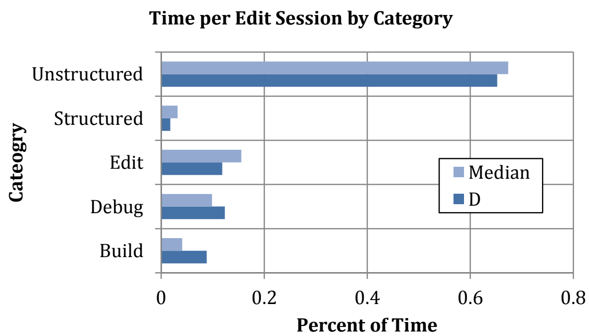

Figure 8: Developer Analysis Against Median

opers did not spend the entire time in their desktop environment because they had a virtual machine environment available for testing. Thus categories like testing and debugging were affected by whether they used the virtual environment or their desktop for these activities. Because those categories are not part of navigation, they are not discussed here. Due to this, there are some instances where developers have no activity recorded on some days.

One developer's build effort and the underlying build frequency was much higher than the group median, see Figure 8. We asked the developer to explain how building more frequently helps them. By doing more frequent build, (s)he is trying to ensure the correctness of the work completed so far. The developer feels like (s)he resolves the bugs faster with frequent builds because the code and logic is fresh in her memory. If they build less frequently, even though it reduces the build effort, there are more bugs each time and the developer often spends more time in identifying the problem area than fixing it. Hence the developer feels more frequent builds improves her productivity.

In another case, we observed a higher level of unstructured navigation than median and asked the developer to explain. Even though they agree structured navigation is better than unstructured navigation, (s)he prefers unstructured navigation in some specific instances. The developer is the author of the code and (s)he knows the code completely. "I know that the line I need to fix is in XYZ method and it is four blocks down from where I am currently. It takes much less time for me to reach the fix location using unstructured navigation. If the code is not written by me, I will prefer structured navigation." This instance cautions us to gather information on aspects such as code familiarity and developer experience with the product when assessing the results of a navigation study.

Developers also reiterated how they expected a different experience from Blaze such as immediate prompts to suggest navigation commands within the context of what they are working on. This contradicts some developers' comments from the pre-study survey that indicated interruptions from a game would be distracting and unwelcome.

In the feedback, we shared a set of achievement badges tied to specific navigation commands with developers. Developers said badges would both inform and motivate them to use some specific tools or commands. This contradicts a result from the pre-study survey where respondents rated badges last in their influence motivating them to improve. The difference in this opinion may result from the context where developers, when presented with badges, see how badges serve as both a feedback mechanism and guidance on the commands for improving their score and practices.

6. RELATED WORK

The two areas of related work we discuss are research studies using other monitoring methods and studies applying gamification to other areas.

6.1 Monitoring Practice Studies

Robillard, Coelho and Murphy explore hypotheses around how developers can be more effective at performing a maintenance task [17]. Robillard et al. say developers are more successful finding and fixing bugs when they create a detailed plan for implementing a change, use structured navigation with keyword search or cross-reference search, and only review methods once during their search. We build on their work by testing methods for increasing the use of structured navigation in developers' everyday practice.

Johnson and Kou defined Zorro [8], a system for detecting whether developers use Test Driven Development techniques based on data from Hackystat, a monitoring framework developed by Johnson. Zorro divides development activities into episodes delimited by events such as configuration management code check-in, start of a unit test run, or start of a build. Using the distinct events developers follow within these episodes, Zorro determines whether the episode followed Test-Driven Development practices in their prescribed order or a different order. In two student-based studies comparing Zorro classifications with a simultaneous observational screen video, Zorro achieved between 70% [9] and 89% [8] accuracy when classifying episodes into their proper TDD scenarios. Data from our study also classifies a series of events into sessions (a similar concept to episodes) enabling us to determine the most significant factor for edit session duration.

Murphy-Hill, Parnin, and Black [15] use the Mylyn Monitor to explore whether or not developers use the automated refactoring tools present in Eclipse. They look for specific refactoring commands in Eclipse and determine the amount of time developers use tools versus hand refactoring the code. For this study, we focus on code navigation as a high-value area for improving developer efficiency.

The Pro Metrics (PROM) tool provides a framework for collecting data for further analysis from tools used by developers. [3] It provides a flexible data model and a plug-in architecture to facilitate collection from different data sources. Studies conducted using PROM include studying benefits of refactoring on productivity [11] and impact of refactoring on re-usability [12]. Studies using the PROM tool correlate the time spent editing with code metrics obtained through source code analysis. Taking a different focus on techniques, this study uses more detailed events captured from the IDE to detect commands used to navigating through the code in addition to knowing duration of edit sessions for code modules.

Murphy-Hill et al. study a large usage history data set and apply several different algorithms to accurately suggest IDE commands to novice users [13]. The "Most Widely Used" algorithm recommends commands based on the collective usage profile of the team and performed nearly as well as the more sophisticated algorithms. This work takes the idea of making recommendations into a game context to influence how recommendations in a game influence developers to use recommended commands.

6.2 Gamification

When discussing gamification, concerns are raised about replacing an activity like software development, which is intrinsically motivated, with extrinsic motivation provided by points and achievements.

Beecham and colleagues provide a thorough analysis of existing literature on studies of motivation in Software Engineering [2]. They claim that the most common motivator found in cited references is "the work". The list of demotivators is also useful with common job satisfaction items like stress and inequity in recognition, plus poor quality software (low accomplishment) and lack of influence in decision making. The paper also lists characteristics of people in the professions including need for independence (autonomy), desire to learn new skills and tackle new challenges.

Maehr proposes an affirmative theory whereby individuals achieve as a member of a social group, choosing the behavior that meets the expectations and values of the group that are significant to them [10]. A key factor in raising achievement motivation is establishing a social group where the person is motivated to belong and excel.

Hamari et al. analyze achievement patterns in game systems and provide guidance on defining successful achievement systems [6]. They recommend achievement awards be setup as a secondary form of scoring so players have multiple objectives available for a particular game. Players should feel like they earned achievements in the game instead given trivial tasks.

Singer and Schneider demonstrate the use of a message board and points for encouraging students to increase their frequency of commits to the source code repository [21]. The communication mechanism enabled students to see each other's progress and resulted in more frequent commits than baseline. Participants valued the communication and collaboration aspect and some valued the competition enough to change their opinion on optimum commit frequency. The subsequent thesis by Singer [20] describes results of an experiment conducted across two iterations of a class where active feedback on commits was deployed to one course and another course served as the control group where commit frequency was simply monitored. Results show an increase in the frequency of commits at a statistically significant level. This study uses fine grained instrumentation to identify practices used between commits such as structured navigation.

Dubois and Tamburrelli discuss theoretical concepts behind applying gamification to software development [4]. Their approach is to consider how gamification could be applied throughout the software development life-cycle to motivate many practices in different phases. They provide guidance on assessing gamification in the software development domain including analyzing the actors and game mechanics, defining integration with existing tools and processes, and designing the evaluation of results. Dubois and Tamburrelli demonstrate this guidance using a study of students working on a class project. The study results indicate that gamified environments for software engineering practices may improve results by incorporating competitive elements like leaderboards.

7. THREATS TO VALIDITY

The threats to construct validity occur when measurements may not reflect the intended operational definition. The key measures of navigation ratio and points in our study could have results differing from their definition.

Measuring navigation ratio involved categorizing events into structured and unstructured navigation. The categorization of events is specific to this study and documented herein, however, it may not be accurate to the intent of the operational definition of structured navigation depending on how the developer uses some commands like Find. Another factor affecting this measurement may occur when developers repeat a command such as hitting the down arrow key many times, which may be something other than navigation (though it is classified as such for this study). In feedback, we learned developers were reviewing and reading code which may account for these repetitive actions.

The points measurement is more susceptible to influences of simple developer activity compared to navigation ratio. Thus, a very busy developer could accumulate more points using a lower navigation ratio than a developer who has less activity performed more efficiently.

Threats to internal validity are those conditions that create alternative reasons for the conclusions, or confound the conclusion. The Hawthorne effect, where improvement occurs because we are seeking a change in a particular practice, could impact our results because participants received a presentation on the study objectives prior to the start of the study. By making the purpose of our measurement and contest known, we may have triggered the participants to think about navigation commands they use before establishing a baseline. This threat is mitigated by the data showing four participants did not improve over the course of the study while two demonstrated some improvement.

Participants in the study were not randomly selected, they are all volunteers from one intact team chosen due to their advanced practices and understanding of metrics. Therefore the participants may be predisposed to have an interest in the kind of tool and methodology we are studying and have bias towards a favorable opinion.

The intact team may be more familiar with the code they maintain thus be less likely to change their navigation practices. The post-study survey responses confirmed developers are less inclined to use structured navigation when they work in code they are very familiar with.

Threats to external validity include the fact that the study was conducted in one development location of one company. Other companies and locations may have different cultures incompatible with this type of monitoring activity.

Threats to generalizable results exist because we conducted the study with six developers thus the population size is too small to draw statistically significant inferences from.

The study was conducted in India, the country with the most positive response in our pre-study survey, perhaps where this activity was most acceptable. Thus, other countries who rated their view of gamification lower may be less receptive.

Participants in the study were professional software engineers with at least seven years of experience. In other settings where developers have different levels of experience, the results may differ. In the pre-study survey, experience was not part of the question set, thus it could have results skewed to a particular age group.

8. CONCLUSIONS

We set out to demonstrate whether gamification of software development practices could succeed in the industrial environment at ABB. Through a series of three steps we conducted a pre-study survey with 130 developers, a study of a gamified navigation practices with six developers in an intact team using the Blaze tool, and post-study survey with the same six developers. A common thread of findings through all three steps is developers are interested in the idea of using games to help them learn and improve their practices. On average, 74% of developers were interested in gamified workspace in our pre-study survey, and five out of six developers indicated elements like points and leaderboards influenced them in our post-study survey. The recorded usage data shows two out of six developers in our study responded positively when they started receiving points feedback. We learned developers need more detailed and immediate feedback from Blaze on ways to improve their practices. Some developers also appreciated charts providing a historical view of how they spend their time by category compared with the median for the team. This compliments their feedback on the game showing developers want more information on how to become more effective at their job. Future work includes providing a mechanism that shares tool recommendations between developers, and allowing developers to share their data with a mentor for feedback.

9. ACKNOWLEDGMENTS

The authors wish to thank ABB and the study participants in India for their support of this research. Thanks specifically to Vinay Augustine, David Shepherd, Suzanne Snipes, and Balaji W for your support and contributions to Blaze.

10. REFERENCES

[1] *Oxford Dictionaries Online - American English (US).* Oxford University Press, 2013.

[2] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp. Motivation in Software Engineering: A Systematic Literature Review. *Inf. Softw. Technol.*, 50(9-10):860–878, Aug. 2008.

[3] I. D. Coman, A. Sillitti, and G. Succi. A Case-study on Using an Automated In-process Software Engineering Measurement and Analysis System in an Industrial Environment. In *International Conference on Software Engineering, ICSE*, pages 89–99. IEEE, May 2009.

[4] D. J. Dubois and G. Tamburrelli. Understanding Gamification Mechanisms for Software Development. In *Foundations of Software Engineering.* ACM, 2013.

[5] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1), 2009.

[6] J. Hamari and V. Eranti. Framework for Designing and Evaluating Game Achievements. In *Digra Conference*, Sept. 2011.

[7] T. Hill and P. Lewicki. *Statistics: Methods and Applications.* StatSoft, Inc, 2007.

[8] P. M. Johnson and H. Kou. Automated Recognition of Test-Driven Development with Zorro. In *Agile Conference (AGILE)*, pages 15–25. IEEE, Aug. 2007.

[9] H. Kou, P. Johnson, and H. Erdogmus. Operational Definition and Automated Inference of Test-driven Development with Zorro. *Automated Software Engineering*, 17(1):57–85, 2010.

[10] M. Maehr. Culture and Achievement Motivation. *American Psychologist*, 1974(29):887–896, 1974.

[11] R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti, and G. Succi. A Case Study on the Impact of Refactoring on Quality and Productivity in an Agile Team. In B. Meyer and et al., editors, *Balancing Agility and Formalism in Software Engineering*, volume 5082 of *Lecture Notes in Comp. Sci.*, pages 252–266. Springer], 2008.

[12] R. Moser, A. Sillitti, P. Abrahamsson, and G. Succi. Does Refactoring Improve Reusability? In M. Morisio, editor, *Reuse of Off-the-Shelf Components*, volume 4039 of *Lecture Notes in Comp. Sci.*, pages 287–297. Springer Berlin Heidelberg, 2006.

[13] E. Murphy-Hill, R. Jiresal, and G. C. Murphy. Improving Software Developers' Fluency by Recommending Development Environment Commands. In *Foundations of Software Engineering.* ACM, Nov. 2012.

[14] E. Murphy-Hill and G. C. Murphy. Peer Interaction Effectively, Yet Infrequently, Enables Programmers to Discover New Tools. In *ACM Conference on Computer supported cooperative work*, CSCW. ACM, 2011.

[15] E. Murphy-Hill, C. Parnin, and A. P. Black. How We Refactor, and How We Know It. *IEEE Transactions on Software Engineering*, 38:5–18, Jan. 2012.

[16] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2013.

[17] M. P. Robillard, W. Coelho, and G. C. Murphy. How Effective Developers Investigate Source Code: an Exploratory Study. *IEEE Transactions on Software Engineering*, 30:889–903, Dec. 2004.

[18] Saatchi and Saatchi S. Gameification for business brands and loyalty, June 2011.

[19] D. Shepherd, K. Damevski, B. Ropski, and T. Fritz. Sando: an Extensible Local Code Search Framework. In *Foundations of Software Engineering.* ACM, 2012.

[20] L. Singer. *Improving the Adoption of Software Engineering Practices Through Persuasive Interventions.* PhD thesis, Gottfried Wilhelm Leibniz UniversitÃČÂd't Hannover, 2013.

[21] L. Singer and K. Schneider. It was a Bit of a Race: Gamification of Version Control. In *Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques (GAS).* IEEE, June 2012.

[22] W. Snipes, V. Augustine, A. R. Nair, and E. M. Hill. Towards Recognizing and Rewarding Efficient Developer Work Patterns. In *International Conference on Software Engineering*, ICSE. IEEE, 2013.