5-2013

Web 2.0 Technologies in the Software Development Process.

Jocelyn Borgers
East Tennessee State University

College of Business and Technology
Honors Thesis Signature Approval Form

Web 2.0 Technologies in the Software Development Process

April 2, 2013

The members of the Thesis Committee approve the Senior Honors Thesis for *Jocelyn Borgers* _____

Thesis Committee Co-Chair
Bill Pine _____

Thesis Committee Co-Chair
Dr. Martin Barrett _____

CBAT Thesis Committee Member
Dr. Suzanne Smith _____

External Thesis Committee Member
Dr. Robert Price Jr. _____

CBAT Honors Director
Dr. Tom W. Moore _____

EAST TENNESSEE STATE UNIVERSITY COLLEGE OF BUSSINESS AND TECHNOLOGY

Web 2.0 Technologies in the Software Development Process

Jocelyn Borgers

4/2/2013

Contents

Intro 3

Web 2.0 in Software Development 3

Social Media for Software Engineering 5

Business Impact of Web 2.0 Technologies 6

Using Wiki's in Software Development 8

The Impact of Social Media on Software Engineering Practices and Tools 9

Collaboration Tools for Global Software Engineering 10

A Survey of Social Media Use in Software Systems Development 11

Communication Patterns in Geographically Distributed Software Development and Engineers' Contributions to the Development Effort 12

On the Perceived Interdependence and Information Sharing Inhibitions of Enterprise Software Engineers 13

Mining Task-Based Social Networks to Explore Collaboration in Software Teams and Predicting Build Failures Using Social Network Analysis on Developer Communication 15

Summary 18

Web 2.0 Technologies in Software Development Survey 19

Appendix I: Participant Letter 21

Appendix II: The Survey 22

Acknowledgements 24

Works Cited 26

Intro

Web 2.0 is another way of saying social media. The idea is that Web 1.0 is a user interacting with a web page or web site; Web 2.0 is a user interacting with another user, or users, through a web page or web site. Web 2.0 technologies are designed to help people communicate without distance being an obstacle.

The software development process varies from company to company but all of the methodologies used can be broken down into five phases of requirements, design, implementation, quality assurance, and maintenance. Requirements is the phase where the developers talk with the customer to try and determine what product the customer is asking for and get an idea of the features that the customer needs. The Design phase is where the requirements that have been gathered are looked at and molded into a design of a system. Implementation is where the code for the system is written. Quality Assurance is to debug the software created in the Implementation phase. The Maintenance is the day to day upkeep of the software once it has been implemented. Throughout these phases, communication is imperative. The further along in the software development process, the more it costs to fix any kind of bug.

Web 2.0 technologies are designed to help with communication making it quicker and easier. This can be a great benefit to global companies who have software developers scattered around the world all working on the same project. This thesis explores the connection of Web 2.0 technologies with the software development process by surveying local software developers. It attempts to connect developers' usage of social media with specific phases of the development process.

Web 2.0 in Software Development

Various authorities have noted the importance of effective communication for effective software development. This includes DeMarco and Lister (1987), who state that developers spend 70% of their

time working with others, and Hayes (2003) , who notes that even small miscommunications can create major problems for a software project.

Social media have gained recognition as an effective means of communication for software development. Research into the use of social-media based communication for software engineering includes surveys of Web 2.0 use in real-world software development, case studies of communication within geographically distributed projects, and attempts to correlate communications in social networks with build failures (Wolf, Schroter, Damian, & Nguyen, 2009). Types of social media that are being used in software development include technical Q&A sites, Wikis, Blogs, Microblogs, and social networks:

- A technical Q&A site is a place for users to ask technical questions of and get answers from other software engineers or programmers who are users or site employees. An example of this is Stack Overflow (Stack Exchange Inc., 2013).
- A wiki is online form of "collaboration and knowledge sharing" (Louridas, 2006). Ideally wikis should be easy to use and update, even for people who may lack programming experience. An example of a technical wiki is Hughes Systique Corporation's wiki, HSC Technical Wiki (Hughes Systique Corporation, 2013).
- A blog is a series of writings on a website by an author, or a group of authors, upon which readers can then comment. Software engineering related blogs are used to share and get feedback from employees and customers about technical information. One such blog is Adobe's secure software engineering team (ASSET) blog (Adobe Systems Incorporated, 2013).
- A microblog, like Twitter, is a blog that limits users to short messages. Microblogs are useful for sharing links, status updates, and meeting reminders because they are quick and concise. Some companies have their own microblogs for these purposes like Yammer (2013).
- A social network site is a website, like Facebook, that links people into a graph. This graph allows users to traverse its links to learn about others in the network. Microsoft supports an internal social network, Codebook, which provides its employees with social connections along

with access to shared work artifacts like code, documentation, and bug reports (Begel, DeLine, & Zimmermann, 2010).

This use of social media to foster software development has attracted the attention of various researchers, who have studied how social media are being used in software development along with how effective its use has been. Examples include Begel et al.'s description of possible social media applications for each stage of Tuckman and Jensen's five-stage, forming-storming- norming-performing-adjourning model for group dynamics (2010); Andriol's interviews regarding and surveys on social media in software engineering (2010); Louridas's (2006) analysis of Wiki's in software development (2006); Storey et al.'s and Lanubile et al.'s assessment of social media tools in software development (2010); Black et al.'s pilot survey of Twitter followers (2010); Cataldo et al.'s comparison of social-media-based patterns of communication used by teams from two different geographically distributed projects (2008); Grubb and Begal's analysis of how well software developers think they communicate (2012); and Wolf et al.'s research on the use of data mining to create a social network that could detect project failure (2009).

Social Media for Software Engineering

As a starting point for identifying further uses for social media in software development, Begel et al. consider possible applications of social media to Tuckman and Jensen's five-stage model for group dynamics. In the first stage of Tuckman and Jensen's model, forming, workers are organized into teams. Social media like social networks can be used to find and recruit team members during this first phase of team development. During the second, storming phase of team development, a team sets goals for a project. Begel et al. see blogs as a tool for "enabl[ing] coworkers to discuss application ideas, future trends, potential markets, and their own personal requirements for working in a team" (Begel, DeLine, & Zimmermann, 2010). Blogs are also good for getting customer feedback. During the third, norming, phase, social media can help teams to select a software methodology and engineering process. In this stage microblog-based communication can be helpful especially if team members can't meet face to face. According to Begel et al., frequent use of microblogs helps teammates to understand each other and

quickly distribute knowledge. In the fourth, performing, stage, social media can help with software creation. Begel et al. cite Amazon's use of an application, Mechanical Turk (MTurk), which partitions programs into smaller tasks for to others to implement, at a rate of between $.01 and $.10 per task. This type of crowd sourcing has been successful in creating speech recognition software. Finally, social media can support the last, adjourning, stage by facilitating the process of reflecting on a project's processes.

Business Impact of Web 2.0 Technologies

Andriole (2010) describes research into how companies are using Web 2.0 technologies and how well those technologies were being used. Andriole interviewed "[a]pproximately 15 senior managers"[sic] (Andriole, 2010) from five different companies, including a big pharmaceutical company, a global chemicals company, a national real estate and mortgage company, a global IT company, and a large financial services company. On simple questions like "Which Web 2.0 technologies have you piloted?" answers differed little. All companies, for example, had piloted blogs and Wikis. For more complex questions like "How would you quantify the impact in knowledge management, rapid application development, customer relationship management, collaboration, communication, innovation and training?", answers were longer and had more variation. One of the more important points learned through the interviews was "that Web 2.0 technologies, in spite of the hype, are entering the enterprise slowly but deliberately" (Andriole, 2010).

Andriole also surveyed ninety-eight companies from around the world, including the five companies that participated in his interviews. The survey mainly focused on six different performance areas: knowledge management, rapid application development (RAD), customer relationship management, collaboration and communication, innovation, and training. The results, given in Table 1, showed that companies had avoided most external uses of Web 2.0 technologies for fear of information leaks—but that this fear was starting to dissipate. The two most used Web 2.0 technologies were Wikis and blogs. The least used were virtual worlds followed by crowdsourcing and mashups. 22% of the participants did

not use any Web 2.0 technologies at all. Andriole believes these results also show growth of external deployment, confirming what was learned from the interviews.

According to Andriole "[c]ollaboration and communications were slightly exaggerated in the expectations survey data, though collaboration and communications were still highly affected by Web 2.0 technologies." All six areas showed overly enthusiastic expectations. Web 2.0 technologies for knowledge management were found to be more descriptive and operational than prescriptive and strategic. Wikis, blogs, and folksonomies/content management are the top three technologies that improve knowledge management. The leading technology for RAD was Wikis. Mashups would have value in RAD but according to the survey they are not yet used in this way. For improving customer relationship management external customer blogs and external social networks were found to be the most beneficial. Compared to the other five performance areas, Web 2.0 technologies were used very little for customer relationship management. This suggests that not enough research has been done in this field. In the area of communication and collaboration Web 2.0 technologies are so far having a significant effect. Wikis again are most common followed by blogs and internal social networks. The survey data indicates that these technologies are not yet being used to their full potential.

These companies were in their initial deployment of Web 2.0 technologies and still testing their impact. In the area of innovation Wikis and internal employee blogs, according to the survey, had the most impact. External crowd sourcing was surprisingly underused. Training was the last area evaluated. As noted by Andriole "survey respondents have not yet defined how Web 2.0 technologies can contribute to training." Wikis were still at the top of the list with 40.8% of participants saying that they had seen improvement in training with Wikis. 28.4% had not seen any improvement in training with any Web 2.0 technologies.

Together the interviews, survey, and observation show that tend to restrict the use of social media for a number of reasons. The main reason is that social network are not very secure; by having an internal

social network a company can control it easier and make it more secure. Some companies that use social networks are concerned about the amount of time employees spend on them. This is also the reason some companies block social networks. Still the interview and survey data shows that "Wikis, blogs, and social networks, perhaps due to their consumer-to-consumer origins, have been deployed more than the other Web 2.0 technologies" (Andriole, 2010). Simpler technologies like Wikis and blogs have been adopted quicker than complex ones, e.g. mashups and crowdsourcing. Companies also have to consider the possibility of cyberbullying, or people looking to harm them through their Web 2.0 technologies.

Using Wiki's in Software Development

Wikis were created by Ward Cunningham in 1995 as a way of creating "new forms of collaboration and knowledge sharing on the Web and empowering many people to contribute to online content." (Louridas, 2006) The word "wiki" is Hawaiian for easy. Ease of use is the main requirement for a wiki implementation device or a wiki engine. Wikis should be written in natural language with a few extensions for text entities like headers, titles, and links. Wikis should also be easy to update by people who lack programming expertise.

Making Wikis easy to update by any user can invite errant and malicious changes. To address this issue, Wikis should maintain a history file that records who made changes, what they changed, and when they made those changes. A history file can also support collaborative document creation by providing a central location for letting collaborators know who did what when.

Wikis have different uses for software development. According to Louridas "[a]t the simplest level, you can use Wikis as a project documentation repository" (Louridas, 2006). Wikis can also be used to create and maintain project documentation as well as administrative documents like meeting agendas and meeting minutes. With a wiki, meeting agendas can become the meeting minutes. Conducting project-related discussions on Wikis allows users to see the development of ideas, and keep things neat as opposed to email where long conversations with many people can be hard to follow.

The Impact of Social Media on Software Engineering Practices and Tools

Storey et al. concur with Begel et al's observation that research into the use of social media in software engineering has been limited. Storey et al. attribute this to the rate at which new social media technologies are being adopted, which they see as too fast for researchers to keep up with (Storey, Treude, Deursen, & Cheng, 2010). While some empirical studies have examined the adoption of social media in software engineering as a whole, most have focused on social media in individual projects.

Integrated Design Environments (IDEs) were originally designed to help individual developers to write, debug, refactor, and reuse code. Newer IDEs, known as Collaborative Development Environments (CDEs), provide network-based features for team development that include version control, release management tracking, issue tracking, and mailing lists. According to Storey et al., "[t]he primary goal of CDEs is to reduce friction in collaborative processes" (Storey, Treude, Deursen, & Cheng, 2010). Two examples of CDEs are Jazz and the Microsoft Team Foundation Server. Another category of IDEs are Social Development Environments (SDEs), which use web sites to support collaborative software development. Examples of SDEs include Stackoverflow, Github, TopCoder, and Freshmeat.

Storey et al. discuss the uses and potential use of blogs, microblogs, tagging, feeds, social networks, mashups, and crowdsourcing for software development. Blogs are widely used for exchanging technical knowledge, including "how-to" documentation. Microblogs like Twitter are used to exchange quick bits of information. Tagging is used to track issues, work items, and builds. Research has also been done on the use of tagging in IDEs. Feeds provide short real-time update statues to subscribers. Jazz uses a feed to provide information of project occurrences like workspace changes, developer statuses, and process events to project members. Social networks like Facebook can, when used for software engineering, connect work artifacts with people. Mashups combine data or functionality from different sources. Crowdsourcing can provide developers with feedback on their software, including descriptions of bugs and ideas for new features. One way of leveraging crowdsourcing is to present different versions of a website to different users, to see which page was more effective and easier to use.

Managing globally distributed projects is challenging. Social media can become a virtual water cooler for these global projects, to help get communication flowing. Further research is needed to find best practices of social media for software engineering. Storey et al. believe that "there is considerable evidence that these tools are being used in innovative ways in practice, yet there have been few research studies that articulate the benefits and limitations of these tools on software engineering" (Storey, Treude, Deursen, & Cheng, 2010). They suggest open, closed, and crowd-sourced projects; co-located or globally distributed projects; projects across different domains; and projects using agile and traditional development processes as some possible topics for future research.

Collaboration Tools for Global Software Engineering

Lanubile et al. (2010) recommend various tools that are not necessarily associated with Web 2.0 for global software engineering. One, version control systems, help team members manage software artifacts. A second, trackers, a.k.a. tickers, maintain records of current defects, changes, and requests for support. A third, build tools like Maven and CruiseControl, create and schedule workflows and provide remote repositories. Secure remote repositories are a major need for distributed projects and an even greater need for globally distributed projects. A fourth, modelers, help developers create software artifacts; Lanubile et al. differentiate collaborative software engineering from the simple sharing of files based on the former's use of modelers. A fifth, knowledge centers, are websites where team members can share knowledge. Some examples of knowledge centers are the Eclipse help system and KnowledgeTree. A final type of tool, communication tools, is widely used by software engineers, especially those in globally distributed project teams. These can be divided into asynchronous tools like email, mailing lists, newsgroups, Web forums, and blogs; synchronous tools like telephones, conference calls, instant messaging, voice over IP, and video conferencing; and tools like websites with auto-dialer response that combine asynchronous and synchronous communication.

Lanubile et al. view Web 2.0 as a set of technologies that "extends traditional collaborative software by means of direct user contributions, rich interactions, and community building" (Lanubile, Ebert,

Prikladnicki, & Vizcaino, 2010). Web 2.0 technologies have become common in open source and globally distributed projects. CDEs can contain some or all of the previously mentioned collaborative tools. Companies often don't use CDEs since they have a legacy tool or environment they must use.

Lanubile et al. also classify communication tools for globally distributed projects by function. They place tools in one of four different categories: project management, requirements engineering, designing, and testing. Tools with calendars and milestone tracking are useful for project management. Examples of these tools are ActiveCollab and WorldView, both of which have a web-based interface. Tools that use natural language text are useful for requirements engineering. Doors and IRqA are two such tools that have Word-based interface and a web interface. Lanubile et al. suggest Gliffy and Creately as good design and modeling tools. Gliffy and Creately support team communication with features for commenting and creating blogs. TestLink is an example of a popular testing tool. It has a web-based interface for convenient access. Some other testing tools are Selenium and OpenSTA.

Lanubile et al. argue that collaboration tools fail to support the ready integration of data across tools, due to varying implementations of collaborative features. They also fail to ensure efficiency, consistency, and information security for projects with multiple teams in multiple locations. According to Lanubile et al., "[n]o current tool or CDE supports all the activities necessary for global software engineering" (Lanubile, Ebert, Prikladnicki, & Vizcaino, 2010).

A Survey of Social Media Use in Software Systems Development

In (2010) Black et al. present the results of a pilot study on the use of social networking in globally distributed development teams. Participants were recruited with a Twitter message. This message, which was sent to followers of Dr. Black, asked "Do u work in software systems development use social media 2 communicate? If so please take our survey: http://tinyurl.com/yavuwj7" (Black, Harrison, & Baldwin, 2010). Of the thirty-one participants, 20 were male, 9 were female, and 2 didn't specify gender. A higher proportion of men than women had a bachelors or master's degree. The average age of respondents was

41 and the amount of time that they had worked with their current employer varied widely. Participants ranked Twitter highest among social media, followed by instant messaging, LinkedIn, Facebook, Googlewave, and Plaxo. This result should be considered in view of the authors' use of Twitter to recruit responders. In response to a question about what was communicated using social media, over 50% responded yes to "new ideas". "Social arrangements" was the top answer for ways in which social media were used. Another question that asked respondents for the number of hours they used social media each day was disqualified from consideration since it failed to differentiate background execution of social media from active use of a social media application.

The survey's validity is questionable, given the non-random selection procedure, the small number of respondents, and the way the respondents were informed about the survey. The authors tried to enhance the survey's validity by incorporating some unstructured interviews into their data. Despite these problems Black et al. argue that. "[their] results [were] valid and useful within this restricted context" (ibid.).

Communication Patterns in Geographically Distributed Software Development and Engineers' Contributions to the Development Effort

In (2008), Cataldo compared communications dynamics for two engineering projects that were implemented by two different companies. Both projects involved the use of Geographically Distributed Software Development (GDSD). Both projects used modification requests to track individuals' accomplishments. People who completed more modification requests were said to be more productive.

Project A in Cataldo's study produced a large distributed system for the data storage industry. Project A was studied through 4 releases (about 3 years). This project had 114 developers, divided into 8 development teams at 3 locations. The primary means of communication were an online chat system (IRC), a modification request tracking system, email, and video conferencing. No formal roles were

assigned to the developers. In the absence of formal roles, a group of people from all three locations became the center for coordination and information exchange.

Project B produced a large medical system. Project B's company divided work into 8-week iterations; this study followed the 7th iteration. This project had 83 engineers, divided into 10 teams at 4 locations: two in the United States and one each in India and Eastern Europe. Formal roles were defined and assigned to project developers. This included the use of designated coordinators for managing communications. Far fewer people assumed this role than in Project A.

The modification requests resolved in Project A were analyzed according to factors that included domain experience, common usage, and average change size. It was found that "individuals benefit by ample access to information rather than by controlling the flow of information" (Cataldo, 2008). Since not enough data was collected on Project B to analyze these same factors, a subset of those factors was analyzed instead. This included intercept and degree centrality. As in Project A, individuals from Project B that were highly connected completed more modification requests. In both cases more experienced developers completed more than inexperienced ones.

In Project A, the core group of highly productive developers became the contacts between different locations. In Project B, where the role of liaison was formalized, the liaisons contributed less to the development effort than the rest of the developers. Cataldo suggests future research to account for the differences in productivity between the informal and formal liaisons.

On the Perceived Interdependence and Information Sharing Inhibitions of Enterprise Software Engineers

In (2012), Grubb and Begel present the results of a case study on information sharing among 3,000 software engineers at Microsoft in summer 2010. This study involved a survey of people who had worked at Microsoft for at least a year. The survey characterized information sharing among developers in terms of Microsoft's model for software engineering roles, which further categorizes engineers into

developers, project managers (PM), and testers. Information sharing was characterized in terms of two new terms: iDepend, meaning that "the respondent perceives that he depends on work done by others outside his team", and oDepend, meaning that "the respondent perceives that individuals from outside his team depend on him for his work" (Grubb & Begel, 2012).

Grubb and Begel obtained 989 responses to their survey. To determine how respondents related their work to that of other Microsoft employees, participants were asked to affirm or deny two statements:

- "I personally depend on the work of people outside my team."
- "People outside my team depend on the work that I personally do."

Only 67.4% said yes to both statements.

Grubb and Begel identified five reasons for software engineers not wanting to share information. The material requested was sensitive and could not be openly shared; the engineer was uncertain about whether the information should be shared; the information requested was incomplete; the requestor was not trusted by engineer; and the engineer did not believe the requestor would understand the information.

33.2% of participants said that they had been asked for sensitive material that they could not share. 25.1% said that they felt uncomfortable sharing the information requested. Based on job functions it was found that PMs were more comfortable sharing more information than they were allowed to, whereas testers did not feel comfortable with sharing what they were allowed to share. Follow up interviews with testers found this was due to lack of confidence.

Some questions on the survey attempted to determine if some documents were relied on more than participants thought. Most respondents reported that they waited on (iDepend) release schedules and specifications and others were waiting on them (oDepend) for specifications and source code. iDepends were almost always greater than oDepends. For example 53.85% of PMs had an iDepend for others work status, but only 17.41% said they had an oDepend on their work status. Grubb and Begel also found that participants with a higher iDepend were more likely to feel comfortable sharing their work. PMs were

found to be more likely to say that they had oDepend than developers or testers. This was expected because it is the PM's job to coordinate their team with others. Respondents of all three job functions that were more experienced had a higher oDepend.

Grubb and Begel suggest various ideas to improve communications involving work-related dependencies. One is a system that notifies a user of others with a possible interest in a document being developed, then allows that user to choose to whom to send that document. They also suggest that the system allow users to modify their messages for different types of recipients. This, however, might make more work for users than current systems, becoming more of a hindrance than a help.

Mining Task-Based Social Networks to Explore Collaboration in Software Teams and Predicting Build Failures Using Social Network Analysis on Developer Communication

Social networks can be used to study collaboration among members of software development teams. In (2009), Wolf et al. describe "a systematic approach for mining large software repositories to generate social networks that use task-based communication between developers" (Wolf, Schroter, Damian, & Panjer, Mining Task-Based Social Networks to Explore Collaboration in Software Teams, 2009). Once created, these social networks can be used to analyze patterns of communication.

Wolf et al. tested their approach in the context of IBM's Rational Jazz project, an effort to create a development environment that focuses on collaboration support. Because their data was based on a single case study, the authors caution that their results may not apply to other projects. The Jazz project is a collaboration among 16 sites in the United States, Canada, and Europe. Six-week iterations of the Jazz project are divided into planning, development, and stabilization. Work items are traceable across an entire iteration, due to the Jazz project's use of a source code management tool called Stream. Each team has its own Stream that each team member contributes to. When a team is ready to integrate a build, the team publishes all changes from its Team Stream to the Project Integration Stream. These builds produce

an indicator of ERROR, WARNING, or OK. Because an indicator of WARNING was treated the same as an OK, both were regarded as a successful build.

Wolf et al. modeled Jazz-based collaboration using a graph that represents project members and collaborative tasks as nodes and communication paths as weighted lines. Weights indicate amount of communication: for the Jazz project this was the number of comments on a work item. This graph was filtered to create task-based social networks for multiple uses. Wolf et al. described three strategies for filtering this graph, corresponding to different scopes of collaboration: project members, tasks, and communication. These strategies included the use of different filtering orders to obtain different views of communication patterns. One way of ordering the filters produces a social network that identifies candidates for communication brokers. A communication broker is someone who understands what two parties need to communicate and who can easily communicate with both parties. Communication brokers are useful when two people could have trouble communicating, e.g. are in different time zones. To produce this network, all tasks would first be filtered by relevance to the project at hand. Then project members would be filtered based on contributions to those tasks left. After being filtered, those project members are connected to each other based on task communication. This creates a social network that makes it easy to find possible communication brokers for any two project members.

Wolf et al. performed two major studies with the Jazz project. To retrieve data for these studies from the Jazz project's data repository, a plug-in for the projects' client was created. This attempt to monitor communications was complicated by the repository's large size and the need to make data extraction minimally invasive to preserve Jazz's performance. This was accomplished using incremental queries, the results of which were stored into a reporting and analysis database. Information from this database was processed using Java Universal Network/Graph (JUNG).

The first study focused on using communication structures to predict build failures, which can occur nightly for team builds and at the end of each iteration for project builds. First, to find "whether any

individual measure of communication structure can predict integration failure or success" (Wolf, Schroter, Damian, & Nguyen, Predicting Build Failures Using Social Network Analysis on Developer Communication, 2009), data was collected and analyzed from each team and project build. Teams were categorized into groups based upon successful and failed builds. Wolf et al. compared the networks' density, centrality, structural holes, and number of direct connections. Density measures the density of the connections and is calculated as a percentage. Centrality measures the importance of a node in a social network. Structural holes represent a lack of communication between nodes. No correlation however, was found between any of these measures and successful and failed builds. The authors then correlated these measures using a Bayesian classifier and the *leave one out cross validation* method. The results of this correlation were categorized using recall and precision. Recall is the percentage of successful or failed build predictions out of the total number of actual successful or failed builds. Precision is the percentage of actual successful or failed builds out of the total number of successful or failed build prediction. The recall values of failed team builds were between 55% and 75% and the recall values of failed project builds were at least 89%. The precision values were also high.

The second study involved the visualization of communication in the Jazz project. Contrary to expectations, communication response times were largely independent of distance. Wolf et al. attribute this unexpected result to the Jazz team's use of best practices like prioritizing offsite requests and tools for integrated development.

The authors concluded with various suggestions for deploying task-based social networks. If, like the Jazz project, a project's data repository is large and in current use; incrementally mining the data reduces load and performance issues. Storing mined data will help keep a data repository from getting bogged down with re-mining. Using a visualization of a project's social network can help bring new developers up to speed. Wolf et al. advocate further research about what type of information is valuable to developers and the best means for conveying that information. A particularly important finding was that successful integration of a build depends on good communication between developers and team members.

Failures could be prevented by monitoring communication and affecting it, when necessary. There are multiple possibilities for this type of system. A warning system could be created to alert someone when communications are becoming weak so that project workers would know they needed to communicate more. Management could use social network analysis to effectively manage projects. Social networks created as early as in the first quarter of a project proved to be helpful for predictions. A team's communication structure can be easily examined by management with such a social network to identify problems like missing communication links. Once a missing link is discovered a project manager can assign communication brokers to fill it. Possible communication brokers are easy for project managers to find with a task-based social network. Wolf et al. claim that task-based social networks can be adjusted by project managers to meet their specific project needs.

Summary

Communication is an important part of developing a successful software system. Some Web 2.0 technologies are in widespread use as communication tools for software projects. Wikis and Blogs are the most successfully used so far, probably due to their ease of use. IDEs are also using Web 2.0 technologies to help with communication and collaboration between project members. SDEs, social IDEs, are being used but not as extensively as Wikis and blogs.

Global distribution of work has become commonplace for software development. Large companies have offices and workers all over the world. In Cataldo's study, communication dynamics of geographically distributed projects were found to be better when no formal communication roles were assigned (Cataldo, 2008). More research is needed to understand this relationship.

From interviews and surveys conducted by Andriole it was found that Wikis and internal employee blogs were deployed the most. External technologies were used less because of security risks. Use of social networks was less than expected because of employers' concerns about their effect on employee

productivity. Black et al. and Begel et al. agree that Web 2.0 technologies have become popular for software development because they are already in high use; people are comfortable using them.

To understand information sharing Grubb and Begel surveyed software engineers at Microsoft. They categorized survey participants based on the types of project managers, developers, and testers. Their results indicated that people did not realize others depended on their work as much as they depended on someone else's and project managers were more likely than the other two categories to say someone depended on their work.

Social networks can be used to map communication paths between project members and project documents. Wolf et al. showed how this could be used to analyze communication paths and find potential build failures. This kind of social network can also be used to get newly employed developers up to speed. Wolf et al. suggested that other uses could be found with more research.

Web 2.0 Technologies in Software Development Survey

I created a survey to determine the use of Web 2.0 technologies in the five phases of the software development process. The target respondents for this survey are software engineers currently working on a software development project. This is to keep the findings as current as possible with what is used now. The target area for the survey is the Tri-Cities and Knoxville area simply because that is the area I know. If more respondents are needed then the area can be expanded. The survey will be administered using the survey site surveygizmo.com (Widgix, LLC dba SurveyGizmo). SurveyGizmo has a setting to allow survey respondents to remain anonymous. Potential respondents will receive a link via email or social media asking them to take the survey. If they choose to do so, clicking on the link will take them to the first page of the survey. The survey consists of five pages, which can be viewed in Appendix I.

There are three different hypothetical results for this survey. The first is that software engineers are Facebook friends with their co-workers and as a result talk about work related items over Facebook. This

hypothesis comes from the popularity of Facebook and software engineers desire to keep up with the latest technologies. The second hypothesis is that Blogs, Wikis, and internal social networks are the most commonly used Web 2.0 technologies in the software development process. This hypothesis is based on the fact that previous surveys have found these three to be most commonly used because they are among the easiest to implement and use. The third and final hypothesis is that the most communication and the most Web 2.0 technologies are used in the requirements and design phase. This hypothesis is based on the fact that the most time and money can be saved by having better communication in these first two stages.

Appendix I: Participant Letter

Dear Participant:

My name is Jocelyn Borgers and I am an undergraduate student at East Tennessee State University in the Computer and Information Sciences department. For my undergraduate honors thesis, I am examining Web 2.0 (or social media) technology's uses in software development. Because you are a software developer, I am inviting you to participate in this research study by completing an online survey at the following link *<link will be here>*.

The survey will require approximately 10 to 15 minutes to complete. There is no compensation for responding nor is there any known risk. IP addresses and other identifiable information will not be taken. Copies of the project will be provided to my East Tennessee State University instructors; my thesis advisors Dr. Marty Barrett, and Bill Pine; my thesis board; Dr. Suzanne Smith, and Dr. Robert Price Jr.; and Honors Coordinators Dr. Christopher Wallace, and Dr. Thomas Moore. If you choose to participate in this project, please answer all questions as honestly as possible. Participation is strictly voluntary and you may refuse to participate at any time.

Thank you for taking the time to assist me in my educational endeavors. The data collected will provide useful information regarding how Web 2.0 technologies are being used in software development. If you would like a summary copy of my thesis please feel free to email me at borgers@goldmail.etsu.edu. Completion of the survey will indicate your willingness to participate in this study. If you require additional information or have questions, please contact me or my thesis advisors at the emails listed below.

Sincerely,

Jocelyn Borgers
East Tennessee State University
Computer and Information Sciences
borgers@goldmail.etsu.edu

Faculty Thesis Advisors:

Dr. Marty Barrett
barrettm@etsu.edu
Bill Pine
pine@etsu.edu

Appendix II: The Survey

Web 2.0 Technologies in Software Development

Consent

1. You are requested to participate in undergraduate research that will be supervised by Principal Investig technologies in software development. This survey should take about 10 to 15 minutes to complete. Partic be kept anonymous. However, whenever one works with email/the internet there is always the risk of com anonymity. Despite this possibility, the risks to your physical, emotional, social, professional, or financial we There are no direct benefits associated with your participation. You have the option to not respond to any required. Checking of the check box below and submission of the completed survey will be interpreted as and that you affirm that you are at least 18 years of age. If you have any questions about the research, ple at borgers@goldmail.etsu.edu. *

- I consent to participate and I am at least 18 years of age
- I do not wish to participate

Personal Demographics

2. What is your age? *

3. What is your highest level of education? *

- 12th grade or less
- Graduated high school or equivalent

APPROVED
By the ETSU IRB
APR 11 2013
By aa
Chair/IRB Coordinator

- Some college, no degree
- Associate degree
- Bachelor's degree
- Post-graduate degree

4. How many years have you been in the software engineering field? *

5. Are you currently working on a software development project? *

- Yes
- No

Project Demographics

6. Is the project globally distributed? *

- Yes
- No

7. How long have you been working on this project?(estimate) *

Days

Months

Years

8. What kind of software development methodology are you using? *

- Waterfall
- Agile
- Incremental
- Rapid Application
- Other (please specify)

9. Do you have a team member assigned to communicate with other teams? *

- Yes
- No

10. How many people are on your software development team? *

11. What is your role in this project? *

- Manager

- Designer
- Developer
- Tester
- Other (please specify) [ ]

12. On a scale of 1-10 how well do you communicate with your co-workers (1 being not at all, 10 being gr

-- Please Select --
1
2
3
4
5
6
7
8
9

13. On a scale of 1-10 how well do your co-workers communicate with you (1 being not at all, 10 being gr

-- Please Select --
1
2
3
4
5
6
7
8
9

14. Do you personally use Facebook? *

- Yes
- No

Facebook Questions

15. How much are you on Facebook?

- Several times a day
- Once a day
- Several times a week
- Once a week
- Several times a month
- Once a month
- Less

16. Are you "friends" with any of your coworkers on Facebook?

- Yes
- No

17. Do you discuss work related items while on Facebook?

- Yes
- No

Social Media

18. Please rank the following phases from most to least communication required.

Drag items from the left-hand list into the right-hand list to order them.

Requirements
Design
Implementation
Quality Assurance
Maintenance

A few terms: A technical Q&A site or SDE (Social Development Environment) is a place for users to ask qu users or site employees about technical questions like the site StackOverflow. A microblog is a blog that lir Twitter. Internal Social Networks are social networks that work like Facebook or LinkedIn except they are c company and only contain data and information that is company specific.

19. Do you use any of these social media for anything work related? *

	Yes	No
Facebook *	○	○
Technical Q&A Site/SDE *	○	○

Wiki *	○	○
Blog *	○	○
Microblog *	○	○
Internal Social Network *	○	○

20. During which phase do you most often use these social media?

	Requirements	Design	Implementation	Quality Assurance	Mainter
Facebook *					
Technical Q&A Site/SDE *					
Wiki *					
Blog *					
Microblog *					
Internal Social Network *					

21. How often do you use these social media in the requirements, design, implementation, and quality ass

	Several times a day	Once a day	Several times a week	Once a week	Seve
Facebook	○	○	○	○	
Technical Q&A Site/SDE	○	○	○	○	
Wiki	○	○	○	○	
Blog	○	○	○	○	
Microblog	○	○	○	○	

Internal Social Network	○	○	○	○	

22. How helpful are these social media in the software development process?

	Very Helpful	Helpful	Somewhat Helpful	Neutral	Somewhat Unhelpfu
Facebook	○	○	○	○	○
Technical Q&A Site/SDE	○	○	○	○	○
Wiki	○	○	○	○	○
Blog	○	○	○	○	○
Microblog	○	○	○	○	○
Internal Social Network	○	○	○	○	○

Thank You!

Thank you for taking our survey. Your response is very important to us.

Acknowledgements

This thesis would not have been possible without my thesis advisers Bill Pine and Dr. Marty Barrett. Thank you, Mr. Pine, for teaching my software engineering classes which gave me the knowledge to begin this thesis. Thank you, Dr. Barrett, for giving me the idea for my thesis, agreeing to be my thesis adviser, and getting Mr. Pine involved. Both of you kept me going with words of encouragement and extra ideas and advise. I really expanded my knowledge in this area because of you both. I also want to thank the other members of my advisory board, Dr. Smith and Dr. Price. You both agreed to be readers without hesitation. I also want to thank Dr. Pfeiffer for reviewing my writing. You are the best and fastest grammar reviewer I have ever met; I appreciate your doing this for me. I would also like to thank my computer science friends who tested my survey. Lastly I want to thank all my friends and family who have supported me and offered me words of encouragement throughout this process.

Works Cited

Adobe Systems Incorporated. (2013). *Adobe Secure Software Engineering Team (ASSET) Blog*. Retrieved March 4, 2013, from http://blogs.adobe.com/asset/

Andriole, S. J. (2010, December). Business Impact of Web 2.0 Technologies. *Communications of the ACM*, pp. 67-79.

Begel, A., DeLine, R., & Zimmermann, T. (2010). Social Media for Software Engineering. *FoSER '10 Proceedings of the FSE/SDP workshop on Future of software engineering research* (pp. 33-38). New York: ACM.

Black, S., Harrison, R., & Baldwin, M. (2010). A Survey of Social Media Use in Software Systems Development. *Web2SE'10.* Cape Town, South Africa: ACM.

Cataldo, M. (2008). Communication Patterns in Geographically Distributed Software Development and Engineers' Contributions to the Development Effort. *CHASE '08*, 25-28.

DeMarco, T., & Lister, T. (1987). *Peopleware: Productive Projects and Teams.* New York: Dorset House Publishing Co., Inc.

Grubb, A., & Begel, A. (2012). On the Perceived Interdependence and Information Sharing Inhibitions of Enterprise Software Engineers. *CSCW '12* (pp. 1337-1346). New York: ACM.

Hayes, J. H. (2003). Do You Like Pina Coladas? How Improved Communication Can Improve Software Quality. *IEEE Software*, 90-92.

Hughes Systique Corporation. (2013, January 13). *HSC Technical Wiki | Main / Home Page browse*. Retrieved March 4, 2013, from HSC Technical Wiki: http://wiki.hsc.com/

Lanubile, F., Ebert, C., Prikladnicki, R., & Vizcaino, A. (2010). Collaboration Tools for Global Software Engineering. *IEEE Software*, 52-55.

Louridas, P. (2006). Using Wikis in Software Development. *IEEE Software*, 88-91.

Stack Exchange Inc. (2013). *Stack Overflow*. Retrieved March 4, 2013, from Stack Overflow: http://stackoverflow.com/

Storey, M.-A., Treude, C., Deursen, A., & Cheng, L.-T. (2010). The Impact of Social Media on Software Engineering Practices and Tools. *FoSER '10 Proceedings of the FSE/SDP workshop on Future of software engineering research* (pp. 359-364). New York: ACM.

Widgix, LLC dba SurveyGizmo. (n.d.). Retrieved from SurveyGizmo: http://www.surveygizmo.com/

Wolf, T., Schroter, A., Damian, D., & Nguyen, T. (2009). Predicting Build Failures Using Social Network Analysis on Developer Communication. *ICSE'09*.

Wolf, T., Schroter, A., Damian, D., & Panjer, L. D. (2009). Mining Task-Based Social Networks to Explore Collaboration in Software Teams. *IEEE Software*, 58 - 66.

Yammer. (2013). *Yammer: The Enterprise Social Network*. Retrieved March 4, 2013, from Yammer: https://www.yammer.com/