Keeping Up With Your Friends: Function Foo, Library Bar.DLL, and Work Item 24

Andrew Begel
Microsoft Research
Redmond, WA, USA
andrew.begel@microsoft.com

Thomas Zimmermann
Microsoft Research
Redmond, WA, USA
tzimmer@microsoft.com

ABSTRACT

Development teams who work with others need to be aware of what everyone is doing in order to manage the risk of taking on dependencies. Using newsfeeds of software development activities mined from software repositories, teams can find relevant information to help them make well-informed decisions that affect the success of their endeavors. In this paper, we describe the architecture of a newsfeed system that we are currently building on top of the Codebook software repository mining platform. We discuss the design, construction and aggregation of newsfeeds, and include other important aspects such as summarization, filtering, context, and privacy.

Categories and Subject Descriptors:
D.2.9 [*Software Engineering*]: Management—*productivity* H.5.2 [*Information Systems*]: User Interfaces—*User-centered design*

General Terms: Management, Human Factors

Keywords: Knowledge management, Social networking, Mining software repositories, Inter-team coordination, Regular expression, Regular language reachability

1. INTRODUCTION

Large-scale software development requires many teams to work together in order to successfully build a product. Dependencies between the software components that make up the product link the teams together and form the foundation of their communication network. Ideally, each team should regularly keep one another informed of their status, and communicate about changes to specifications, designs, APIs, component ownership, and schedules. However, many studies have shown that this does not happen [1, 4, 5, 6, 7, 8, 10, 11, 14, 15]. Miscommunication, mistrust, unmet expectations, and dysfunctional relationships between teams [1] can result in increased team anxiety and hinder good decision-making by product managers who lack complete and correct information to manage the risk of depending on others.

If each software team could have transparent access to the information held by the others, much of this anxiety and risk could be ameliorated. The most sought-after information [4] relates directly to the work actions of the teams' software engineers, such as their code checkins, bug report updates, test executions, status reports and explanatory documents. Fortunately, records of these activities are already captured in software development-related repositories, such as version control systems, bug databases, test harnesses, and Sharepoint document repositories. Using our Codebook software repository mining platform [3, 2], we can reveal these actions as events reported in a newsfeed.

Consider this scenario. A program manager, Klaus, is trying to determine if his team will meet their ship date. From the status reports emailed directly to him by every developer and tester on his own team, he determines that *his* team's code will ship on time, but they have an external dependency on the team that is working on Sort.DLL. Klaus missed the last Sort.DLL team status meeting, and has no idea if they are running behind or will ship on time.

Klaus pulls up the Codebook home page for Sort.DLL, shown in a mockup in Figure 1. On the left side of the page, he reads through the newsfeed of recent activity on the Sort.DLL library. He scans through the latest builds and sees that they are successfully passing all of their tests; for a few days the Quicksort had performance issues, but they have been resolved in the most recent build. While that is a good sign, he wants to reassure himself that the code churn is slowing down, which since code churn inversely correlates with code quality, would indicate that the team is getting closer to shipping their code as high-priority bugs are the final blocks to go. [12, 13]. He scans through the newsfeed for all the checkins and notices that Chuck, a developer on the team, is still checking in new features at the end of 2009. Darn. The product is probably going to be late. Klaus resolves to contact the team's program manager (listed in the Team section on the right side of the page) to negotiate a new expected ship date.

The newsfeed Klaus read was mined automatically from the software development repositories of the Sort.DLL team, repositories that are publicly accessible within the company. No private emails are crawled, nor are private shares where the team stores sensitive information, such as its financial reports. In addition, one team member, Johanna, is not found in the Codebook page for her team because she has not opted in to allow her data to be collected for this purpose.

Codebook newsfeeds can be created for any type of artifact in the Codebook database. There are activity newsfeeds for artifacts such as bug reports, functions in the code, people, builds, and specification documents. *News rolls up hierarchically*, so any news that is generated by a member of a team is rolled up together to form the news for the team. The news for a class is the sum of the news for the class itself and the news of all of its lexically enclosed members. *News can also be generated through transitive relationships*.

Web2SE '10, May 2-8 2010, Cape Town, South Africa

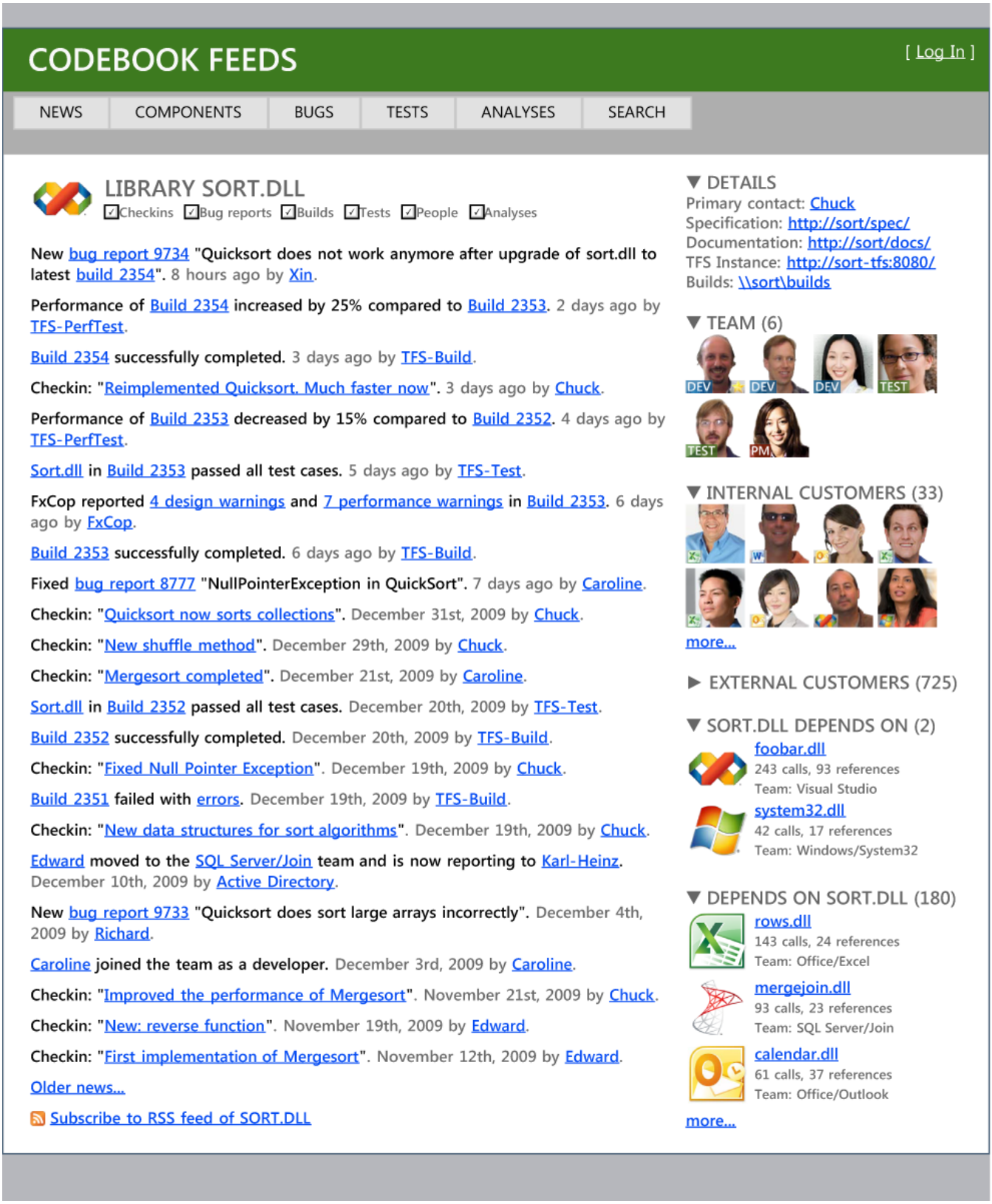

Figure 1: A mockup of a Codebook home page for Sort.dll. On the left is a newsfeed of activity affecting Sort.dll. On the right is information about where Sort.dll's development takes place, the library's contributors and users (inside and outside the company), and a list of dependencies.

For example, Caroline, a developer who recently joined the team, can see a new event on her own newsfeed when Xin files a bug against her code, even though this team's bug database does not have a place for indicating source code relationships.

Newsfeeds can help make product teams' work practices more transparent to one another. With this information available, Klaus can avoid having to pester Sort.dll's project manager for constant status and change notifications. If Klaus needs to speak to some developer about a questionable piece of code, he can quickly find out who to speak to by going to the Codebook newsfeed page for the code and looking at its recent modification activity.

In the rest of this paper, we quickly review the Codebook infrastructure and describe the architecture of the newsfeed system we are currently building on top of it. We discuss various design choices we have made along the way to ensure that newsfeeds are usable and useful by the lay software development teams here at Microsoft. We close with some thoughts on future work.

2. CODEBOOK AND ITS NEWSFEEDS

Codebook is a repository data mining and analysis platform [3] inspired by the field of social networking. In popular social networking applications like Facebook and MySpace, individuals are connected to one another in an undirected network graph. News generated by anyone in the graph is propagated along "friend" links to all individuals one hop away, and appears in their newsfeeds (e.g., "Cheryl is relieved that her son didn't get H1N1 this winter").

Codebook is also built on a graph of relationships, but the nodes are generalized to be not just people, but also bugs, code, tests, builds, specifications, and other work artifacts related to the software development process. The edges between Codebook nodes describe relationships and activities that have occurred. For example, "Todd committed checkin 34," "Mary closed bug 2333," or "Unit test CanConnectToDatabase() has passed." Codebook's graph is extensible with new repository crawler plugins. Currently there are 11 node types and 18 edge types, for a total of 28 unique node-edge-node triple types (e.g., *WorkItem IsLinkedTo WorkItem*, *Person IsManagerOf Person*, and *Checkin Contains RevisedFile*).

Transitively connected pathways in the Codebook graph reveal distantly connected, yet related, nodes. For example, one graph might indicate that a tester named Mary closed bug 2333, which included a stack trace, that names a function Foo, which was modified in checkin 34, which was committed by a developer named Todd. Therefore, Mary's action to close the bug is connected to Todd's checkin.

Our Codebook platform consists of several repository crawlers which create the graph, a set of analyses to discover interesting relationships between nodes in the graph, and an API for applications to access the discovered relationships. In the remainder of this section, we describe how we are building a newsfeed infrastructure on top of this infrastructure.

2.1 Newsfeed construction

Similar to social networking applications, any work artifact (or person) node in the Codebook graph can generate its own kinds of news. For example, when Sarah checks in a change to a method Mergesort(), a news event is triggered:

> Checkin: "Improved the performance of Mergesort."
> November 21, 2009 by Sarah.

All of the "friends" of Sarah and the Mergesort() function will receive this news and see it on their newsfeeds. In addition, friends of the class SortingAlgorithms, Mergesort()'s parent class, also receive the news, since a change to a method is also a change to the class that contains it. We call this concept *"news rollup."* For items that lexically or logically contain others (e.g. nested scopes, folders and files, DLLs and code, public mailing lists and messages, teams and their members, etc), we define their news to be their own news plus the sum of the news of their parts.

News can also *propagate along "interesting" paths* (i.e. transitively connected pathways) of nodes and edges in the graph. For example, Chuck, the developer who wrote the Quicksort() function, receives news items about bugs that are filed against Quicksort, as indicated by stack traces mentioning the Quicksort function. In Codebook, these paths are expressed as regular expressions of node and edge types. A possible regular expression for this example is

> *Person Committed Checkin Modified File Modified SourceCode MentionedBy StackTrace IncludedIn BugReport (DuplicateOf BugReport)**

Note the repeated DuplicateOf BugReport at the end of the regular expression propagates the pathway to all duplicates of the bug which contained the stack trace.

News events in Codebook can occur anytime a node or edge is created or modified. When a new path satisfies a regular expression, news is generated from the node or edge change that caused the regular expression to be satisfied, propagated to all of the nodes in the path, and then further reported to those nodes' friends' newsfeeds.

2.2 Newsfeed aggregation

Depending on the number of artifacts to which each person subscribes, the amount to read might become overwhelming. For example, if Bill Gates wanted to keep tabs on everything that happens at Microsoft Research, he probably does not wish to see every checkin that I or my co-author make to our Codebook source code repository. Mr. Gates might instead enjoy a summarization of our work that tells him that we are making progress on our implementation and submitted a paper about it to the Web2SE workshop at ICSE.

A news summarization, filtering, and aggregation service is available to help Mr. Gates. This service can customize newsfeeds using a function whose inputs include the news you are getting, what artifact (or person) generated it, what topic the news is about, and who you are in relation to where the news comes from. The actual function used need not take all of these factors into account, but could. We are currently exploring how to further customize the function for each individual, based on their role on a project, or their social relationship to the news source news (as mined from social networking applications), such as Bill Gates' relationship with the authors in the example above.

We have placed a simple news filtering service into the mockup in Figure 1. At the top of the newsfeed are checkboxes corresponding to each news type. Unchecking a box will hide news of that type from the feed, making it easier to concentrate on what you might be looking for.

News aggregation algorithms can also help with overload. A simple algorithm might roll up individual updates to a bug report into a single news item, "Cheryl closed 3 bugs," and provide a link to see the individual bugs that Cheryl worked on. There is a danger with designing this kind of aggregation, however. What if the algorithm produced comparative results? "Cheryl closed 3 bugs, which was 5 less than she closed last week." Or "Chery closed 3 bugs out of the 105 closed by her team this week." A worry we have with this kind of system is that individuals might try to game the system in order to look good on the newsfeeds. Another worry is that managers may want to use such a system to help them judge their

employees' productivity. We plan to conduct a study on various ways to aggregate news to understand how software engineers and managers perceive themselves and others when reading newsfeeds. This will help us create value-sensitive [9], judgment-neutral summarization algorithms that will avoid unfair characterizations of the people or work artifacts in question.

3. STATUS AND FUTURE WORK

We have built the Codebook backend for mining software repositories, which can build a graph from what was mined, and propagate news along the relationship edges that connect the nodes in the graph. We have a Codebook search portal called Hoozizat which helps end-users to find objects in the system through a keyword-search [3]. We are currently building the Codebook Feed portal seen in Figure 1.

Although the web portal we have described here concentrates heavily on newsfeeds, we plan to include a portal page for each human engineer to keep track of all of their non-human "friends." To help users find out what artifacts they should subscribe to, a program analysis of the code authored by the user could be enlisted to identify important dependencies to be "friended." To avoid listing every single API used, especially common APIs with in base libraries (e.g. printf(), or File.Read()), we would filter the list by a source code analysis that eliminates artifacts whose changes would not have any semantic impact on the user's code.

An enhancement to the news filtering service will allow the user to limit his newsfeed to only events that fit his current focus. For example, he might only want to read news related to his current task, a dependent product, or news generated by a particular colleague or neighoring team. Shrinking the newsfeeds according to these kinds of foci will help the user identify and read relevant information in a timely fashion.

4. CONCLUSION

Software projects can fail when teams communicate inefficiently and suffer dysfunctional relationships with each other. When timely information about other teams' work practices is available to software engineers and managers, they are better able to manage their risk and ensure their product's success. We believe that Codebook's newsfeeds can improve the situation by making each team's inner work processes more transparent, without requiring the teams to do any extra work to record or publish their activities. Newsfeeds inform colleagues of important events, helping everyone on all of the software teams become more efficient, productive, and successful.

5. REFERENCES

[1] A. Begel. Effecting change: Coordination in large-scale software development. In *Proceedings of CHASE*, May 2008.

[2] A. Begel and R. DeLine. Codebook: Social networking over code. In *Proceedings of ICSE, NIER Track*, 2009.

[3] A. Begel, Y. P. Khoo, and T. Zimmermann. Codebook: Discovering and exploiting relationships in software repositories. In *Proceedings of ICSE, Research Track*, 2010.

[4] A. Begel, N. Nagappan, C. Poile, and L. Layman. Coordination in large-scale software teams. In *Proceedings of CHASE*, pages 1–7, 2009.

[5] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In *Proceedings of CSCW*, pages 353–362, 2006.

[6] B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31(11):1268–1287, 1988.

[7] C. R. B. de Souza, D. Redmiles, L.-T. Cheng, D. Millen, and J. Patterson. How a good software practice thwarts collaboration: the multiple roles of apis in software development. In *Proceedings of FSE*, pages 221–230, 2004.

[8] C. R. B. de Souza, D. Redmiles, and P. Dourish. "breaking the code", moving between private and public work in collaborative software development. In *Proceedings of GROUP*, pages 105–114, 2003.

[9] B. Friedman. Value-sensitive design. http://depts.washington.edu/vsdesign/, 2010.

[10] P. Hinds and C. McGrath. Structures that work: social structure, work structure and coordination ease in geographically distributed teams. In *Proceedings of CSCW*, pages 343–352, 2006.

[11] R. E. Kraut and L. A. Streeter. Coordination in software development. *Communications of the ACM*, 38(3):69–81, 1995.

[12] L. Layman, G. Kudrjavets, and N. Nagappan. Iterative identification of fault-prone binaries using in-process metrics. In *ESEM '08: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 206–212, New York, NY, USA, 2008. ACM.

[13] N. Nagappan and T. Ball. Use of relative code churn measures to predict system defect density. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 284–292, New York, NY, USA, 2005. ACM.

[14] C. Poile, A. Begel, N. Nagappan, and L. Layman. Coordination in large-scale software development: Helpful and unhelpful behaviors. In submission.

[15] R. J. Sandusky and L. Gasser. Negotiation and the coordination of information and activity in distributed software problem management. In *Proceedings of GROUP*, pages 187–196, 2005.