WhoseIsThat: Finding Software Engineers with Codebook

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

Khoo Yit Phang
University of Maryland
College Park, MD, USA
khooyp@cs.umd.edu

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

ABSTRACT
In this demo, we describe WhoseIsThat, a social search portal which we built using the Codebook framework. We improve the search experience in two ways: first, we search across multiple software repositories at once with a single query; second, we return not just a list of artifacts in the results, but also engineers.

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.

1. INTRODUCTION
Coordination between software teams is a persistent problem in software engineering. Teams are dependent on one another for code, APIs, features, schedules, bugs and documentation, and require frequent and effective communication and cooperation to accomplish their tasks. Since engineers are connected by their shared work, tools that discover connections in their work-related repositories can help to address many of the engineers' coordination needs. In this demo, we discuss the Codebook framework and the WhoseIsThat, which both support coordination needs.

(For a discussion of related work, we refer to our previous publication on the Codebook framework [1].)

2. CODEBOOK
Codebook is a repository mining and analysis platform [1,2,3] inspired by the field of social networking. As in popular social networking applications like Facebook and MySpace, individuals in Codebook are connected to one another in a network graph.

In Codebook's graph of relationships, 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" or "Mary closed bug 2333." Currently there are 11 node types and 18 edge types, for a total of 28 unique node-edge-node triple types (for example, WorkItem IsLinkedTo WorkItem, Person IsManagerOf Person, and Checkin Contains RevisedFile).

FSE-18, November 7–11, 2010, Santa Fe, New Mexico, USA.
ACM 978-1-60558-791-2/10/11

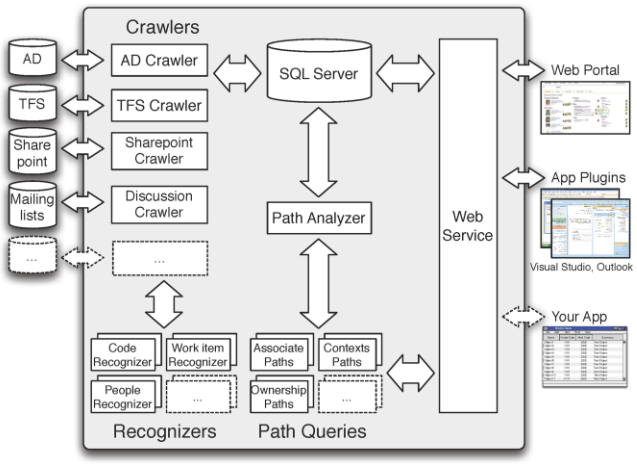

Figure 1. The architecture of the Codebook framework.

Transitively connected pathways in the Codebook graph reveal distantly connected, yet related, nodes. For example, one node 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. In Codebook such relationships are described via graph paths defined by regular expressions.

Figure 1 shows the architecture of the Codebook framework. It 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. More specifically, the Codebook process consists of seven steps:

1. Crawl software repositories and other related repositories such as Active Directory, Sharepoint, and mailing lists.
2. Create a graph of people and artifacts (code, workitems, etc.)
3. Add direct relationships from structure or metadata.
4. Detect additional direct relationships from textual allusions.
5. Add transitive relationships via regular expression paths written by domain experts.
6. Build a search index.
7. Query Codebook data using web services.

New applications can access the Codebook database through web services. In this demo, we show how we used Codebook to build a social search application called WhoseIsThat.

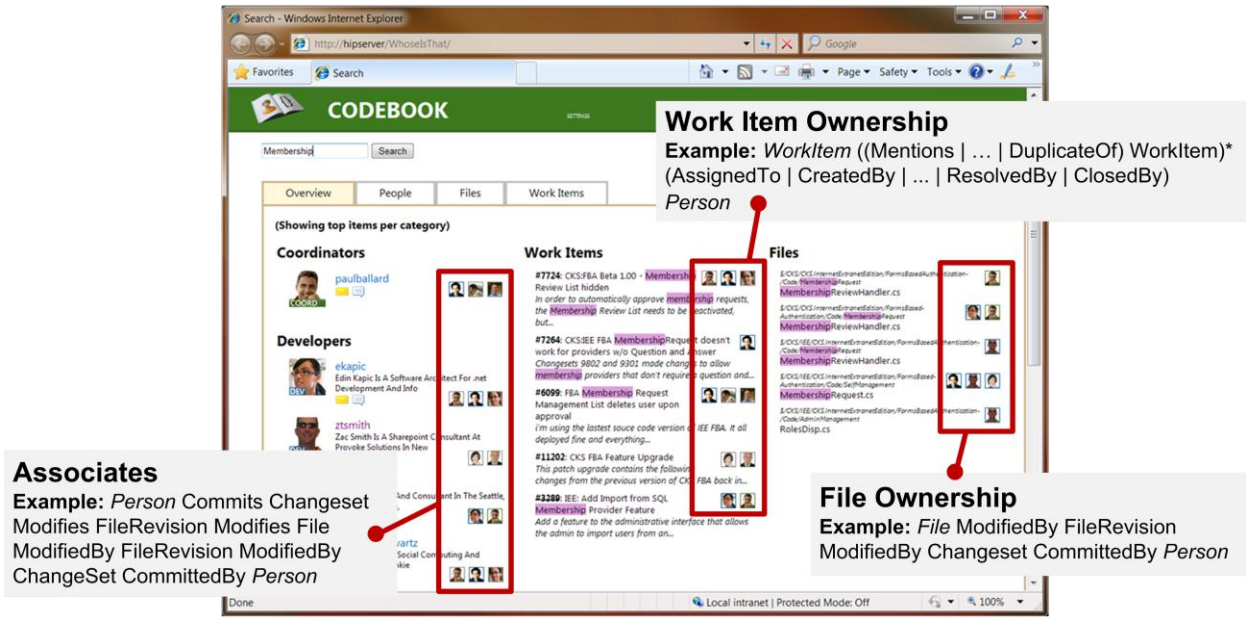

Figure 2. Screenshot of WhoseIsThat search results. Each column shows a different type of result, from left to right: people, work items, and files. Small photos next to each result show the associates for people or owners for artifacts. The screenshot is annotated with the regular expressions used to find associates and artifact owners.

3. WHOSE IS THAT

We conducted a survey at Microsoft to learn about coordination problems between software development teams [1]. We asked engineers—which included software developers, testers, and program managers—to prioritize 31 different information needs around inter-team coordination. Four of the top ten information needs concerned finding the people who own and are responsible for a feature, API, product or service:

1. Given a feature, API, product or service, finding out who the most relevant engineers (developers, testers, program managers, leads, etc.) are in order to contact them.
2. Finding out who owns some code or has ever worked on it in the past.
3. Finding out who owns a specification or knows the most about it.
4. Finding out which teams own the feature, product or service I or my team depend on.

Finding the right person to answer the above questions typically involves searching through various intranet portals and software repositories and is a time-consuming task. Furthermore, engineers have to delve into each search result to locate a related person for an artifact.

To help engineers find people, we built a web-based social search portal called WhoseIsThat [1] based on the Codebook framework. We improve the search experience in two ways: first, we search across multiple repositories (such as TFS and Sharepoint) at once with a single query; second, we return not just a list of artifacts in the results, but also people that can answer questions.

Figure 2 shows an example use of WhoseIsThat. Given some search terms, WhoseIsThat returns a set of related people, work items, code and files from the repositories. In the search results, we augment each **artifact result** with a list of owners, so an engineer can quickly find a person who can answer questions about that artifact *(artifact ownership)*. We augment each **person result** with a list of associates, as this may help an engineer determine the team that person belongs to, or perhaps may help discover another person he might know personally *(associates)*.

Codebook's design was ideal for building the WhoseIsThat tool. First, Codebook crawls multiple repositories and can perform searches across all of them simultaneously. Second, Codebook can return the people, not just their artifacts, by using regular language paths to describe relationships from artifacts to people. We annotated Figure 2 with some of Codebook's regular expressions used by WhoseIsThat. For example to find the owners of a file, Codebook first finds its revisions, from which it gets change sets, which then point to the people who made changes to the file. Codebook comes with a set of regular expression (ownership, associates) which can be customized and extended by domain experts; applications can also additional regular expressions.

At the same time, the design of WhoseIsThat remains flexible. For example, it is possible for WhoseIsThat to include additional relations because Codebook's graphs are typed.

For more information on WhoseIsThat and Codebook, logon to http://research.microsoft.com/en-us/projects/codebook/

Acknowledgements. We thank the FSE reviewers for valuable feedback.

4. REFERENCES

[1] A. Begel, K. Y. Phang, and T. Zimmermann: Codebook: Discovering and Exploiting Relationships in Software Repositories. In *Proceedings of ICSE'2010 – Volume 1*, pp.125-134.

[2] A. Begel and R. DeLine. Codebook: Social networking over code. In *Companion of ICSE'2009*, pp. 263-266.

[3] A. Begel and T. Zimmermann: Keeping up with your Friends: Function Foo, Library Bar.DLL, and Work Item 24. In *Proceedings of Web2SE'2010.*