# Codebook: Discovering and Exploiting Relationships in Software Repositories

Andrew Begel, Khoo Yit Phang and Thomas Zimmermann

Presented By : Ranjodh Singh

# Outline

- Introduction
- Contributions
- Inter-Team Coordination Survey
- The Codebook Architecture
- Applications
- Conclusion

# Introduction

- Coordination between software teams is a persistent problem in software engineering.
- Different teams require frequent and effective communication and cooperation to accomplish their tasks.
- Major reason for poor communication is distributed way of development.
- There is a major need of discovering and keeping track of people along with code.

# Contributions

- Conducted inter-team coordination needs survey for a variety of software team roles.
- Developed flexible and customizable framework for mining software repositories.
- Build two applications named Hoozizat and Deep Intellisense to evaluate the framework.
- Microsoft engineers evaluated the usefulness of our applications in satisfying inter-team coordination information needs.

# Inter-Team Coordination Survey

- Conducted a survey with Microsoft engineers.
- 11% of the 1000 invitees which included developers, testers and managers responded to the survey.
- Aimed to learn how software engineers prioritize 31 different needs about inter-team communication.
- These needs were derived from previous studies and interviews with software engineers.

# Inter-Team Coordination Survey

- Needs were organized into 8 categories:
  - Change notifications
  - Finding dependents
  - Finding other people
  - Finding artifacts
  - Awareness of other teams.
  - Artifact history
  - Work Planning
  - Social Networking
- Respondents were asked to pick the most important needs.

# Inter-Team Coordination Survey

- 10 most important needs indicated by engineers are:

| | | |
|---|---|---|
| 1. | Given a feature, API, product or service, finding out who the most relevant engineers (developers, testers, program managers, operations, leads, etc.) are in order to contact them. | 83% |
| 2. | Finding an expert to talk to who knows a lot about a feature, API, product or service. | 67% |
| 3. | Given a feature, API, product or service from another team, getting a list of servers, directories and repositories where the related code, bug reports, work items, specifications, etc. are located. | 64% |
| 4. | Finding out why a recent change was made, e.g., the related bug report/work item, specifications, or conversation threads in discussion lists. | 62% |
| 5. | Being notified that there is a recent change that affects my code or work items. | 60% |

# Inter-Team Coordination Survey

- 10 most important needs indicated by engineers are:

| | | |
|---|---|---|
| 6. | Finding out who might be affected by a change I make to my code/API. | 57% |
| 7. | Finding out who owns some code or has ever worked on it in the past. | 56% |
| 8. | Finding out who owns a specification or knows the most about it. | 56% |
| 9. | Finding out which teams own the feature, product or service I or my team depend on. | 53% |
| 10. | Finding out everyone outside my team who depends on my feature, API, product or service. | 50% |

# Codebook

- A framework for connecting engineers and their work artifacts together.
- It takes care of the processing and optimization that is needed for efficient crawling, analyzing and querying of the data.
- It discovers transitive relationships between people, code, bugs, test cases and specifications by mining any kind of software repository.

# The Codebook Architecture

- Goal
  - To build a framework general enough to answer inter-team coordination information needs directly, without requiring users to conduct in-depth investigations of raw data sources on their own.
- Data Structure used
  - Graph of nodes and edges
  - Nodes represents repository objects (such as people, change sets, work items, files, and source code).
  - Edges represents relationships of the nodes to one another (such as commits, bug assignments, caller)
  - Codebook's algorithm walks the graph from one set of interesting objects to another via the relationship edges and discover which objects are ultimately transitively connected to each other.

# The Codebook Architecture

Steps to create Codebook graph to answer end user's question:

1. A set of crawlers mine objects from various software repositories.
2. Store these objects in database as nodes in graph.
3. Relationships between these objects are derived from structure stored in the database as edges in the graph.
4. Paths defined by regular expressions are written by domain experts.
5. These paths are then compiled by Codebook into state machines and uploaded into the database.
6. The Path Analyzer runs a regular language reachability algorithm on the database to compute and store the start and end nodes for each path recognized by the regular expressions.
7. Finally data is exposed to applications via web services.

# Crawlers

- Source Code Repository Crawler:
  - It start at the first checkin and proceed until the most recent checkin.
  - Changed files are listed and differences are analyzed.
  - Codebook prototype can analyze only C,C++, C# and VBScript code.
  - All of the symbols contained are stored as nodes in a database table, with metadata columns for symbol name, fully qualified name, kind (class, field, method, operator), programming language and nesting depth.
  - Relationships (superclass, calls, references) between source code symbols are stored as edges in DB.

# Crawlers

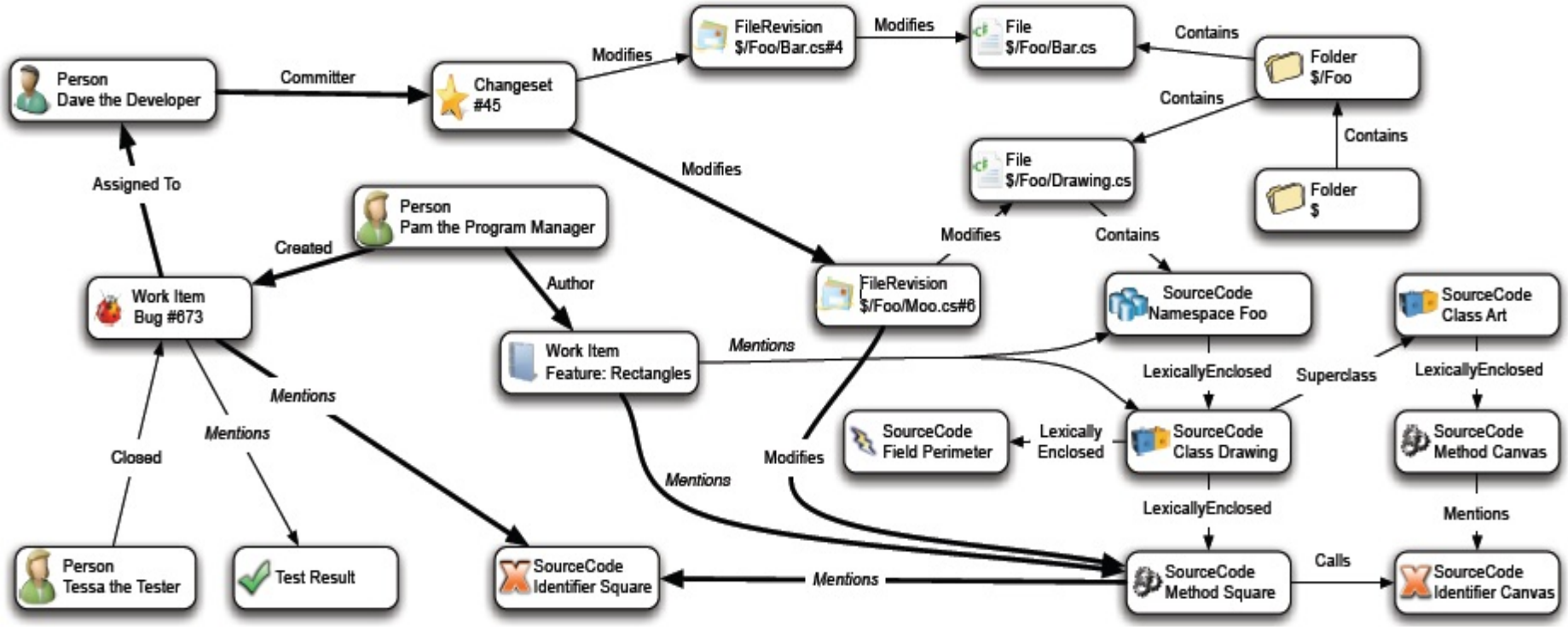
- Employee Directory Crawler:
  - A company's employee directory is crawled.
  - Each person is looked up in the directory and are stored in codebook database.
- Work Item Crawler:
  - The work item database crawler begins at the first work item and proceeds to the most recently created work item.
  - Each revision of work item is processed separately.
  - Work item consist of title, description, a set of people who have changed it.

# Graph Paths

- A graph of related objects is the central component.
- Codebook's graph is complex as there are 9 types of node and 18 edge types.
- Two objects can be related even if they are not directly connected.
- The relations can be represented by regular expressions.
- Ex: *Person Created WorkItem AssignedTo Person.*
- These expressions are written by domain experts.

# Graph Paths

- Possible relationships between objects in Codebook



# Regular Language Reachability

- Codebook computes the set of paths in the graph that conform to the regular expression defined earlier.
- A regular language reachability algorithm is used.
- Unable to return exact set of paths.
- Provides only end points of path and reports the existence of path.

# Search

- Set of keywords are used to search.
- Nodes whose metadata best matches the keywords are returned.
- SQL Server's full text search algorithm is used on nodes and metadata.
- Score generated by the algorithm is combined with domain specific knowledge to derive the ranking function.
- Ex: Individual contributors are ranked higher than managers.

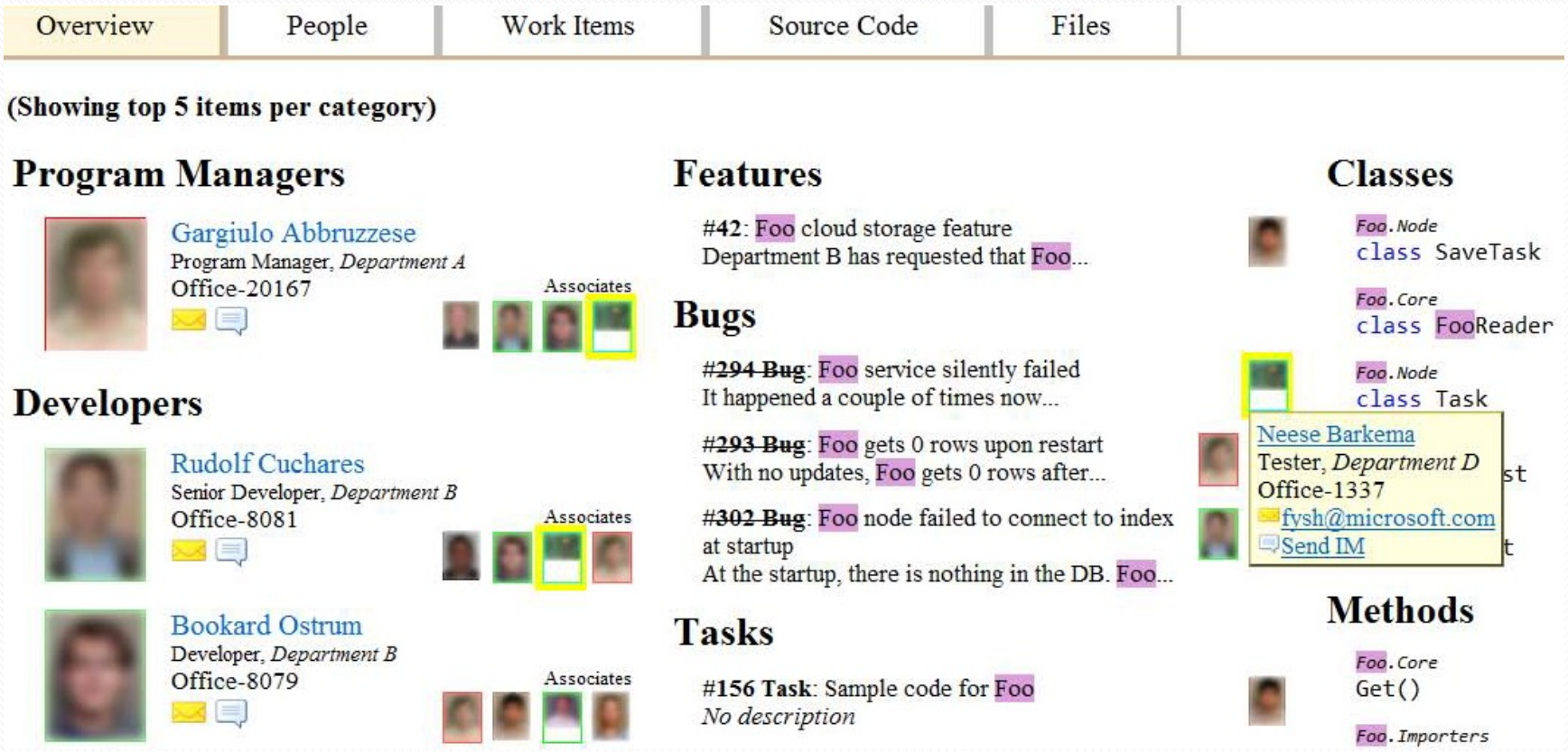
# Web Services

- Show the graph, data contained within and paths to the front-end clients.
- Path regular expressions are uploaded to the system by developers to connect the application to the Codebook web service.

# Hoozizat: Finding People with Codebook

- A web search portal Codebook application to help engineers to address their information needs.
- 6 Microsoft engineers were consulted prior to building this application.
- It crawls multiple repositories and can perform searches across all of them simultaneously.
- It presents results in a web based interface.

# Hoozizat: Finding People with Codebook



# Hoozizat: Finding People with Codebook

- Evaluation:
  - 14 engineers  interviewed to evaluate the utility and user interface design.
  - Participants did 5-6 searches and found that it is returning right results.
  - Inaccurate text allusions results in false positives i.e. codebook returns work items which should not have been in list.

# Deep Intellisense

- Helps in code investigation.
- Displays a reverse chronologically sorted list of events showing everything that has happened to source code symbol in the past.
- Five developers and testers were consulted to understand their needs around code investigation.
- Evaluated with the help of 3 projects and received positive feedback.

# Conclusion

- Addresses the problem of inter-team coordination with Codebook, a framework for connecting engineers and their work artifacts together.
- A survey of software engineers at Microsoft who helped us prioritize the most important information needs around coordination that should be addressed by new tools.
- Built two front-end applications, Hoozizat and Deep Intellisense.

# Discussion

- What further changes can be introduced in Codebook framework to increase the accuracy of results?
- Can size of the company affects the performance of Codebook?
- What are the limitations of the paper?