Knowledge Collaboration by Mining Software Repositories

Thomas Zimmermann
Saarland University, Saarbrücken, Germany
tz@acm.org

Abstract

We will give a short overview on recent approaches to support developers by mining software repositories and outline current and future challenges from which knowledge collaboration can benefit.

1. Introduction

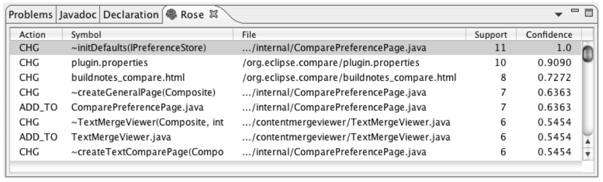
When people collaborate, they communicate and create documents that are shared among each other. In most projects these artifacts are collected and archived in software repositories: For open source projects, communications between developers are stored in *mailing lists*, *newsgroups*, and *personal archives*. Changes to the source code of software are recorded in *version archives* such as CVS. Failures and feature requests are submitted to and discussed in *issue tracking systems* such as Bugzilla. Explicit knowledge such as documentation and design documents is published on websites or wikis.

Recently a new research area evolved that mines software repositories. Although most approaches have focused on understanding software and its evolution so far, software repositories can be leveraged to support developers and their collaboration.

In this paper, we will give a short overview on the state-of-art of mining software repositories with respect to collaboration (Section 2), before we outline ongoing and future challenges from which knowledge collaboration can benefit (Section 3).

2. Supporting Developers

In this section we present several examples how historic data was used to support collaboration among developers. Our overview is not complete since we favored research that actually resulted in tools. For a broader view on mining software repositories we refer to the MSR workshop series [6].



Action	Symbol	File	Support	Confidence
CHG	~initDefaults(IPreferenceStore)	.../internal/ComparePreferencePage.java	11	1.0
CHG	plugin.properties	/org.eclipse.compare/plugin.properties	10	0.9090
CHG	buildnotes_compare.html	/org.eclipse.compare/buildnotes_compare.html	8	0.7272
CHG	~createGeneralPage(Composite)	.../internal/ComparePreferencePage.java	7	0.6363
ADD_TO	ComparePreferencePage.java	.../internal/ComparePreferencePage.java	7	0.6363
CHG	~TextMergeViewer(Composite, int	.../contentmergeviewer/TextMergeViewer.java	6	0.5454
ADD_TO	TextMergeViewer.java	.../contentmergeviewer/TextMergeViewer.java	6	0.5454
CHG	~createTextComparePage(Compo	.../internal/ComparePreferencePage.java	6	0.5454

Figure 1. After an initial change to a method, eROSE recommends related code locations.

Project memory. The Hipikat tool by Cubranic et al. [2] was the first one to combine artifacts from different software repositories such as version archives, bug databases, documentation, and mailing lists. Developers can explicitly query this project *memory* for related artifacts after selecting an initial artifact. Hipikat's recommendations are especially useful for newcomers to a software project.

Guiding developers. The eROSE tool by Zimmermann et al. [10] guides programmers along related changes by mining version archives. When a developer changes f() and other people have changed f() together with g() in the past, eROSE will detect this and suggest *"Programmers who changed function f() also changed function g()"* (see Figure 1). In contrast to Hipikat, eROSE makes recommendations automatically and suggests specific actions (change, add, or delete something).

Software navigation. The NavTracks tool by Singer et al. [7] monitors the *navigation history* of a single developer and use this data to support her future navigation. DeLine et al. [3] extended this work in their Team Tracks tool to multiple developers that share navigation history.

All these tools leverage one or more software repositories to support developers by providing knowledge that is obtained from the past. In the next section, we will outline ongoing research challenges that will further improve knowledge collaboration.

3. Challenges

The research on mining software repositories is currently in an early stage. There are several ongoing challenges that are relevant for knowledge collaboration.

Multiple data sources. Most research focuses only on one data source such as version archives or bug databases. In recent research several software repositories have been combined (starting with Hipikat [2]). This gives additional context to mining. For instance, one can assess changes using bug databases, thus getting a notion of good vs. bad knowledge.

Fine-grained changes. All tools discussed in Section 2 focused only on artifact level such as files, methods, or bug reports. Recently, more fine-grained changes were analyzed [4] and used to identify usage patterns [5] or cross-cutting concerns [1]. Combined with context information this will lead to tools that can assess new changes based on knowledge that is mined from software repositories (think of a self-learning bad smell check across developers).

Collecting new data. Most research analyzed existing software repositories. However, at some point the information available will be exhausted. The NavTracks [7] and Team Tracks [3] tools pioneered a new direction. Instead of taking existing repositories they build their own repositories which are then analyzed. This way, one gets more and better data to turn into knowledge. Related research in this area includes waypointing and social tagging of software as proposed by Storey et al. [8].

Mining across projects. Typically multiple projects are mined at the same time for understanding software evolution. However, when it comes to supporting developer, only single projects were investigated so far. Xie and Pei were the first ones to mine knowledge (usage patterns) across multiple projects [9]. By considering a large amount of projects, one can build a huge knowledge base. The goal will be to improve search engines for source code such as Koders[1] and smoothly integrate them into IDEs.

Although mining software repositories does not explicitly support collaboration, it creates knowledge that helps developers. Since this knowledge is mined from data that comes from different developers, one can think of *implicit* knowledge collaboration: the knowledge is collected in the background and shared among developers.

[1] http://www.koders.com/

4. References

[1] Silvia Breu and Thomas Zimmermann. "Mining Aspects from History." In Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering (ASE 2006), September 2006.

[2] Davor Cubranic, Gail C. Murphy, Janice Singer, Kellogg S. Booth. "Hipikat: A Project Memory for Software Development." In *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 446-465, June 2005.

[3] Robert DeLine, Mary Czerwinski, George G. Robertson. "Easing Program Comprehension by Sharing Navigation Data." In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2005)*, September 2005, Dallas, USA. IEEE Computer Society, pp. 241-248

[4] Beat Fluri and Harald C. Gall. "Classifying Change Types for Qualifying Change Couplings." In *Proceedings of the International Conference on Program Comprehension (ICPC)*, Athens, Greece, June 2006, pp. 35-45.

[5] V. Benjamin Livshits and Thomas Zimmermann. "DynaMine: Finding Common Error Patterns by Mining Software Revision Histories." In *Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/SIGSOFT FSE 2005)*, Lisbon, Portugal, September 2005, pp. 296-305.

[6] International Workshop on Mining Software Repositories 2004-2006, http://msr.uwaterloo.ca/

[7] Janice Singer, Robert Elves, Margaret-Anne Storey. "NavTracks: Supporting Navigation in Software Maintenance." In *Proceedings 21st IEEE International Conference on Software Maintenance (ICSM'05)*, pp. 325-334, September 2005.

[8] Margaret-Anne Storey, Li-Te Cheng, Ian Bull, and Peter Rigby. "Waypointing and social tagging to support program navigation." In *CHI '06: Extended Abstracts on Human Factors in Computing Systems*. Montréal, Québec, Canada, April 2006. ACM Press, New York, NY, pp. 1367-1372.

[9] Tao Xie and Jian Pei. "MAPO: mining API usages from open source repositories." In *Proceedings of the International Workshop on Mining Software Repositories (MSR '06*), Shanghai, China, May 2006. ACM Press, New York, NY, pp. 54-57.

[10] Thomas Zimmermann, Peter Weissgerber, Stephan Diehl, Andreas Zeller. "Mining Version Histories to Guide Software Changes." In *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 429-445, June 2005.