The Role of Emergent Knowledge Structures in Collaborative Software Development

Christoph Treude
Dept. of Computer Science, University of Victoria
PO Box 3055, STN CSC, Victoria, BC, Canada V8W 3P6
ctreude@uvic.ca

ABSTRACT

Many collaboration features in software development tools draw on lightweight technologies such as tagging and wikis. We propose to study the role of emergent knowledge structures created through these features. Using a mixed-methods approach, we investigate which processes emergent knowledge structures support and how tool support can leverage them.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Programming Environments—*Integrated environments*; D.2.9 [**Software Engineering**]: Management—*Programming Teams*

General Terms

Human Factors

Keywords

Collaboration, Software Development, Emergent, Web 2.0

1. INTRODUCTION AND MOTIVATION

In a typical software development process, developers perform numerous activities: they use several tools to develop software artifacts ranging from source code and models to documentation and test scenarios, they use other tools to manage and coordinate their development work, and they spend a large amount of time communicating with other members on their teams.

In recent years, the focus of tools for software developers has shifted towards team-aware tools that support communication and cooperation in one way or another. As these tools are brought into the mainstream, balancing support for formal engineering practices with the social informal aspects of a team becomes essential. A key finding from the Computer Supported Cooperative Work (CSCW) research community is that tools which ignore emergent work practices and social aspects frequently fail [8]. Thus, a challenge for the software engineering community is to develop tools that support both aspects.

Balancing formal and informal user needs is particularly important for task management in a socio-technical system. Tasks are important cogs in the development process machine that need to be carefully aligned with one another, both in what they achieve and in their timing. Since tasks crosscut both technical and social aspects of the development process, how they are managed will have a significant impact on the success of a project. Software development environments typically have explicit tool support for managing tasks. Tasks are assigned to developers, they are classified using predefined categories, and they may be associated with other tasks. Some modern integrated development environments (IDEs) such as IBM's Jazz [5] draw on Web 2.0 mechanisms such as social tagging, commenting, feeds and dashboards to support task organization.

These lightweight mechanisms give rise to socially created **emergent knowledge structures**. We define emergent knowledge structures as *tangible ad hoc artifacts that are created as part of informal individual or collaborative processes in software development but that are not part of the software product.* There are no prescribed processes attached to their creation. Previous studies including our study on tagging [24] have shown that the knowledge stored in these emergent structures adds significant value to artifacts in the software development process, and that it gives rise to collaborative processes around these structures. Furthermore, their tangibility enables developers to refer back to them, unlike oral communication. Social tagging is just one example of the lightweight structures that emerge in collaborative software development environments. Other examples include comments on source code and tasks, wikis, blogging and micro-blogging, but not source code or documentation.

We propose to study the phenomenon of emergent knowledge structures and the use of these structures in the context of collaborative software development.

2. RELATED WORK

Work related to this research can be divided into two categories: Work on social aspects in collaborative software development, and work on lightweight tool support for software developers.

2.1 Social Aspects in Software Development

Software development is recognized to be one of the most challenging management tasks performed by humans [11]. The larger the systems get and the more complicated the

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

compositions of the developing teams are, the more obstacles there are in the way of the release of a software system. Since most software systems are developed by teams, effective coordination and communication are crucial to the success of software projects.

There are at least three strands of research that have considered the impact of social aspects in software development: global software development, open source development and knowledge management. Researchers of these topics recognize that software development processes are more than writing source code, and that "articulation work" [14] must be supported in a software engineering project. According to Gerson and Star [6]: *"Articulation consists of all tasks needed to coordinate a particular task, including scheduling sub tasks, recovering from errors, and assembling resources."* Other examples of articulation work include discussions about design decisions, assigning bug-fixing tasks to developers and deciding on interfaces.

Challenges that arise in collaborative software development include managing implicit knowledge [12], maintaining awareness [9] and leveraging expertise [4]. A key result that has an implication when designing improved tools or processes is that technical artifacts are often intertwined and overloaded with social artifacts during a development project. For example, de Souza *et al.* [2] claim that source code is both a social and a technical artifact and that dependencies do not only exist between artifacts but also between developers. Grinter [7] describes how configuration management tools are sometimes co-opted for articulation work, despite the fact that they have significant shortcomings in supporting this kind of work. She notes insufficient support for individual developers and teams, and reports challenges from a lack of representation of the work itself leading to inappropriate built-in assumptions about the work flow. In a study on source code annotations, Storey *et al.* [20] report that annotations are used to document both technical and articulation activities.

2.2 Lightweight Tool Support

Emergent knowledge structures are often created through lightweight tools such as wikis, blogs, tags and feeds. Research on how these tools support social development processes is not yet far advanced. After the success of Web 2.0 [16] and its lightweight mechanisms, software developers and researchers have started to ask how these mechanisms can be transferred into software development. For the social aspects of team-based development practices, informal collaboration and an *"architecture of participation"* [17] are promising. As developers [11] and designers [28] are lacking informal tools in their work, researchers and industry have started to explore how Web 2.0 and the mechanisms associated with it can be adapted to software development.

Websites such as delicious[1], facebook[2] and wikipedia[3] have found counterparts in software development tools. Dogear [15] introduces the idea of social bookmarking for large enterprises. Codebook [1] uses the approach of facebook for software development. Developers can become friends with work artifacts, keep track of dependencies and discover connections using a web interface. Wikis have also been adopted by software developers [13]. Related to the use of Web 2.0 mechanisms by software developers is the recent trend of moving the IDE into the browser [18, 27]. A prominent example is the Bespin initiative from Mozilla Labs[4].

Annotating artifacts is one of the mechanisms associated with Web 2.0. On a fine grained level, task annotations can be used in programming languages to annotate source code. These annotations evolve to have different meanings and depend on team dynamics. They support informal individual, team and community processes in software development [21]. The tool TagSEA [19] extends the idea of annotations beyond source code to other artifacts such as files and offers syntax for social tagging of source code. Storey *et al.* found that the added semantic information improves the value of annotations and supports reminding and refinding [22].

We have also been investigating how tagging is used to bridge the gap between technical and social aspects of managing development tasks [24]. Our research showed that the tagging mechanism was eagerly adopted and adapted by a large development team of 175 developers, and that it had become a significant part of many informal processes. Our findings indicate that lightweight informal tool support, prevalent in the social computing domain, may play an important role in improving collaborative software development practices. In addition, the aggregation of tags enabled new insights into software development processes such as the co-occurrence of concerns over time [23].

The concept of emergent knowledge structures is related to stigmergy, a mechanism of spontaneous, indirect coordination between agents or actions. Stigmergy refers to the idea that a trace left in the environment by an action stimulates the performance of a subsequent action, by the same or a different agent [3]. Studying emergent knowledge structures goes beyond stigmergy by looking not only at the relationship between actions, but also at the knowledge that accumulates from socially constructed, lightweight structures.

To date, there is no comprehensive work on the role and interplay of emergent knowledge structures. In this research, we propose to study the role of emergent structures in collaborative software development.

3. RESEARCH QUESTIONS

We ask two main research questions with regard to the role of emergent knowledge structures in collaborative software development. Specifically we will explore the following subquestions:

RQ1 Which software development processes do emergent knowledge structures support?

- (a) How are emergent knowledge structures adopted and adapted?
- (b) Why are emergent knowledge structures used and which software development roles do they support?
- (c) Which individual and collaborative processes do emergent knowledge structures support?
- (d) How does the use of emergent knowledge structures evolve over the life cycle of a project?
- (e) What is the impact of emergent knowledge structures on development practices?

[1] http://delicious.com/
[2] http://www.facebook.com/
[3] http://www.wikipedia.org/
[4] https://bespin.mozilla.com/

(f) What knowledge do emergent knowledge structures add to a software development project?

RQ2 How can tools for software developers leverage the knowledge from emergent knowledge structures?

(a) How is the information derived from emergent knowledge structures useful?

(b) How can tool support for emergent knowledge structures be improved?

4. METHODOLOGY AND EVALUATION

To answer the first research question RQ1, we propose to study software development teams and analyze their use of relevant features of the IDE. We will study all features in state-of-the-art IDEs that support the creation of lightweight structures. To date, these features include tags for different kinds of artifacts such as tasks and build results, task annotations, comments on source code, check-ins and tasks in task management systems, awareness tools such as feeds and blogs, and communication features such as email and chat. For these features, we will study how they are being used and what role they play in software development processes.

Our proposed methodology is based on a mixed-methods approach. Through mining the software repositories of development teams we will extract quantitative data on emergent knowledge structures, in particular on tags, comments, feeds, dashboards, email threads and chat logs. In addition, through interviews, surveys and ethnography-style observations on-site, we will add qualitative data in order to understand why developers generate emergent knowledge structures and how they are used. We will also analyze the lightweight structures to help determine what kinds of knowledge they capture.

Our preliminary work has indicated that the knowledge created through lightweight structures such as tags is sometimes hard to access, partly because the structures are usually created to support *ad hoc* processes rather than as part of a formal process. Therefore, in our second research question RQ2 we aim to explore more advanced search and navigation tools to help software developers use already existing knowledge structures more effectively. One of our initial ideas is inspired by CTSearch [10], a tag cloud based search tool for semi-structured data that was developed to query clinic trial data. Figure 1 shows an early prototype of our adaptation of CTSearch. In addition to standard search results, facets are displayed on the left hand side. We will integrate the ideas we gained from implementing and studying the use of ConcernLines [23], a tool we developed as part of our research on tagging in software development. ConcernLines aggregates data obtained from lightweight annotations and enables new insights into development processes over time.

Through user studies with our partners in industry, we will evaluate if the additional information enhances the process of finding relevant tasks. We will collect quantitative data, such as time to find a certain task, as well as qualitative data obtained through interviews and surveys with tool users.

5. EXPECTED CONTRIBUTIONS

With our research, we aim to assist managers and developers in their decisions about using informal collaboration tools in their software development projects. This research is timely as IDEs such as IBM's Jazz continue to adopt informal collaboration functionalities and as some IDEs are moving some or all of their functionality to the browser. Also, software projects are adopting informal collaboration mechanisms such as micro-blogging outside of the IDE[5].

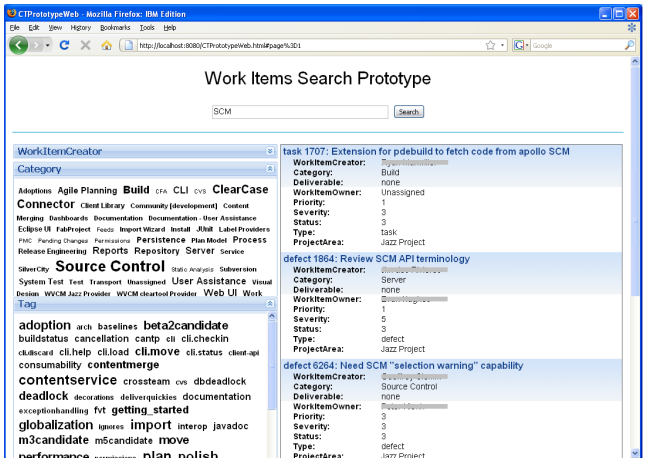

Figure 1: Work Item Search Prototype

We expect three main contributions:

- Identifying the software development processes that are supported by emergent knowledge structures.
- Characterizing the knowledge that emergent knowledge structures add to a software project.
- Improving tool support for emergent knowledge structures.

6. PROGRESS TO DATE

We have already conducted two studies to answer our first main research question RQ1, and we have developed the tool ConcernLines [23] towards answering RQ2. We have also conducted a systematic literature review on studies on collaboration in software development.

In our first study, we showed that social tags for tasks in a large software development project of 175 developers were eagerly adopted and adapted. In particular, lifecycle management, identification of cross-cutting concerns, categorization of work and several idiosyncratic processes were supported by tags. This study was published at ICSE 2009 [24] and follow-up ideas on this work will appear at the NIER track of ICSE 2010 [26]. In a second study, we examined the role of dashboards and feeds in collaborative software development. We found that dashboards became pivotal to task prioritization in critical project phases and that they stirred competition while feeds were used for short term planning. This study will appear at ICSE 2010 [25].

At ICSE 2010, we are also organizing a workshop on Web 2.0 in Software Engineering (Web2SE). The findings from this workshop will broaden our knowledge on the role of informal collaboration mechanisms in software development. We are currently in the process of developing tools that will help us answer our second research question.

[5]E.g., `http://wiki.eclipse.org/Twitter`

We will continue to present our results to the community through venues such as ICSE and FSE.

Acknowledgments

I would like to thank my advisor Dr. Margaret-Anne Storey for supporting this work. This research is funded by a fellowship from IBM.

7. REFERENCES

[1] A. Begel and R. DeLine. Codebook: Social networking over code. In *ICSE Companion '09: 31st Intl. Conf. on Software Engineering - Companion Volume*, pages 263–266, Washington, DC, 2009. IEEE.

[2] C. de Souza, J. Froehlich, and P. Dourish. Seeking the source: Software source code as a social and technical artifact. In *GROUP '05: Proc. of the 2005 Intl. ACM SIGGROUP Conf. on Supporting group work*, pages 197–206, New York, NY, USA, 2005. ACM.

[3] M. Elliott. Stigmergic collaboration: The evolution of group work. *M/C Journal*, 9(2), 2006.

[4] S. Faraj and L. Sproull. Coordinating expertise in software development teams. *Management Science*, 46(12):1554–1568, 2000.

[5] R. Frost. Jazz and the Eclipse way of collaboration. *IEEE Software*, 24(6):114–117, 2007.

[6] E. M. Gerson and S. L. Star. Analyzing due process in the workplace. *ACM Trans. Inf. Syst.*, 4(3):257–270, 1986.

[7] R. E. Grinter. Supporting articulation work using software configuration management systems. *Computer Supported Cooperative Work*, 5(4):447–465, 1996.

[8] J. Grudin. Groupware and social dynamics: eight challenges for developers. *Commun. ACM*, 37(1):92–105, 1994.

[9] C. Gutwin, R. Penner, and K. Schneider. Group awareness in distributed software development. In *CSCW '04: Proc. of the 2004 ACM Conf. on Computer supported cooperative work*, pages 72–81, New York, NY, USA, 2004. ACM.

[10] M.-E. Hernandez, S. M. Falconer, M.-A. Storey, S. Carini, and I. Sim. Synchronized tag clouds for exploring semi-structured clinical trial data. In *CASCON '08: Proc. of the 2008 Conf. of the center for advanced studies on collaborative research*, pages 42–56, New York, NY, USA, 2008. ACM.

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

[12] T. D. LaToza, G. Venolia, and R. DeLine. Maintaining mental models: A study of developer work habits. In *ICSE '06: Proc. of the 28th Intl. Conf. on Software Engineering*, pages 492–501, New York, NY, USA, 2006. ACM.

[13] P. Louridas. Using wikis in software development. *IEEE Software*, 23(2):88–91, 2006.

[14] P. Mi and W. Scacchi. Modeling articulation work in software engineering processes. In *Proc. of the 1st Intl. Conf. on the Software Process*, pages 188–201, 1991.

[15] D. R. Millen, J. Feinberg, and B. Kerr. Dogear: Social bookmarking in the enterprise. In *CHI '06: Proc. of the Conf. on Human Factors in computing systems*, pages 111–120, New York, 2006. ACM.

[16] S. Murugesan. Understanding Web 2.0. *IT Professional*, 9(4):34–41, 2007.

[17] T. O'Reilly. What is Web 2.0: Design patterns and business models for the next generation of software, 2005. http://oreilly.com/web2/archive/what-is-web-20.html.

[18] W. Ryan. Web-based java integrated development environment. BEng thesis, University of Edinburgh, 2007.

[19] M.-A. Storey, L.-T. Cheng, I. Bull, and P. Rigby. Shared waypoints and social tagging to support collaboration in software development. In *CSCW '06: Proc. of the 2006 20th anniversary Conf. on Computer supported cooperative work*, pages 195–198, New York, 2006. ACM.

[20] M.-A. Storey, J. Ryall, R. I. Bull, D. Myers, and J. Singer. Todo or to bug: Exploring how task annotations play a role in the work practices of software developers. In *ICSE '08: Proc. of the 30th Intl. Conf. on Software Engineering*, Washington, DC, USA, 2008. IEEE Computer Society.

[21] M.-A. Storey, J. Ryall, R. I. Bull, D. Myers, and J. Singer. Todo or to bug: Exploring how task annotations play a role in the work practices of software developers. In *ICSE '08: Proc. of the 30th Intl. Conf. on Software Engineering*, pages 251–260, New York, 2008. ACM.

[22] M.-A. Storey, J. Ryall, J. Singer, D. Myers, L.-T. Cheng, and M. Muller. How software developers use tagging to support reminding and refinding. *IEEE Trans. on Software Engineering*, 35(4):470–483, 2009.

[23] C. Treude and M.-A. Storey. Concernlines: A timeline view of co-occurring concerns. In *ICSE '09: Proc. of the 31st Intl. Conf. on Software Engineering*, pages 575–578, Washington, DC, 2009. IEEE.

[24] C. Treude and M.-A. Storey. How tagging helps bridge the gap between social and technical aspects in software development. In *ICSE '09: Proc. of the 31st Intl. Conf. on Software Engineering*, pages 12–22, Washington, DC, 2009. IEEE.

[25] C. Treude and M.-A. Storey. Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds. In *ICSE '10: Proc. of the 32nd Intl. Conf. on Software Engineering*, New York, 2010. ACM. To appear.

[26] C. Treude and M.-A. Storey. Bridging lightweight and heavyweight task organization: The role of tags in adopting new task categories. In *ICSE '10: Proc. of the 32nd Intl. Conf. on Software Engineering – NIER track*, New York, 2010. ACM. To appear.

[27] A. van Deursen. What your IDE could do once you understand your code. Joint keynote for the WSE'09 and VISSOFT 2009 conferences, 2009. Edmonton, Canada.

[28] J. Wu, T. C. N. Graham, and P. W. Smith. A study of collaboration in software design. In *ISESE '03: Proc. of the Intl. Symposium on Empirical Software Engineering*, page 304, Washington, DC, 2003. IEEE.