Bridging Lightweight and Heavyweight Task Organization: The Role of Tags in Adopting New Task Categories

Christoph Treude, Margaret-Anne Storey
Dept. of Computer Science, University of Victoria
ctreude@uvic.ca, mstorey@uvic.ca

ABSTRACT

In collaborative software development projects, tasks are often used as a mechanism to coordinate and track shared development work. Modern development environments provide explicit support for task management where tasks are typically organized and managed through predefined categories. Although there have been many studies that analyze data available from task management systems, there has been relatively little work on the design of task management tools. In this paper we explore how tagging with freely assigned keywords provides developers with a lightweight mechanism to further categorize and annotate development tasks. We investigate how tags that are frequently used over a long period of time reveal the need for additional predefined categories of keywords in task management tool support. Finally, we suggest future work to explore how integrated lightweight tool features in a development environment may improve software development practices.

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, Annotations, Tags

1. INTRODUCTION AND MOTIVATION

Effective task management is essential to the success of software development projects. In modern development platforms such as IBM's Jazz[1] or Microsoft's Visual Studio Team System[2], tasks are the fundamental mechanism to track and coordinate development progress and workflows. Tasks are the hub for linking development artifacts such as builds or source code, and they typically provide the interface for integration with other systems[3].

Task management systems have traditionally focused on a limited set of categories per task, such as task summary and description, a unique id, the creator and the owner of the task as well as priority and severity. As modern integrated development environments (IDEs) add explicit support for collaboration, fields such as comments and tags have been added to the set of common categories of a development task. While there has been some research on how to improve the quality of development tasks from the perspective of a bug reporter (e.g. [2]), there is surprisingly little work on task management. Categories of development tasks such as priority and severity are often taken for granted and rarely questioned. A notable exception is a study by Herraiz *et al.* [5]. Based on the observation that there are fewer clusters of tasks than severity levels, the authors conclude that there is a need to simplify the category set of a development task.

Development task categories such as priority and status can be seen as heavyweight in the sense that there is a limited and formally defined set of values that can be assigned to each of them. Changing this set of values is often impossible as it affects the entire project and would require updating all tasks in a system. On the other hand, with the shift towards built-in support for collaboration, lightweight collaboration functionalities such as tags and comments have been added as an additional way to categorize tasks. In a recent study on the use of social tags for development tasks in IBM's Jazz environment [16], we found that developers used the tagging field to organize tasks. In particular, we found that tags such as "linux", "windows" or "testing" were used to overcome the lack of a corresponding task category.

In the research proposed here, we examine this phenomenon in more detail. We use mailing list data and quantitative data on the use of tags over time to analyze the relationship between the use of heavyweight and lightweight features. We define heavyweight features as task categories such as priority and severity that have a set of predefined values to choose from; and we define lightweight features as properties that do not have any formal processes attached to their creation or modification, such as tags. In addition, we analyze qualitative data from two interviews.

[1]`https://jazz.net/`

[2]`http://msdn.microsoft.com/teamsystem/`

[3]In this paper, development tasks, bug reports and work items are considered synonyms.

Our preliminary results confirm that there are instances of tags being adopted into categories. Ultimately, this research can inform the design of task management systems, and it can make the role of tags in bridging lightweight and heavyweight task organization more explicit.

The remainder of this paper is structured as follows. Related work is summarized in Section 2. In Section 3, we identify our research questions and describe our research methodology. We report and discuss our preliminary findings in Section 4 and Section 5. Section 6 describes the limitations of our approach before we outline the impact and future work in Section 7.

2. BACKGROUND AND RELATED WORK

Research related to this work can be divided into work on tags in software development and work on task management.

Tags in Software Development.

The popularity of social tagging is closely related to the bottom-up nature of tags: tags do not have to be predefined, every user can choose their own tags, and the number of tags per item is arbitrary. Based on these characteristics, tags are used to classify items in an informal manner, and they stand in contrast to formal top-down classification mechanisms.

The use of tags by software developers raises the question of how the informal nature of tags affects software development processes. Social tagging in software development has not been researched extensively yet. The tool TagSEA (Tags for Software Engineering Activities) described by Storey *et al.* [15] uses the ideas of social tagging to support coordination and communication in software development. TagSEA allows tagging of locations in source code – called waypoints – and artifacts such as files. A case study [14] showed that TagSEA provides the user-defined navigation structures that are lacking in traditional task annotations. As a complementary approach, the tools Concern Graphs [11] and ConcernMapper [12] enable developers to associate parts of source code with high level concerns that are defined as needed by the user. In their research on pre-requirements analysis, Ossher *et al.* [9] explore how tags can be hardened as the requirements emerge.

In our recent empirical study on the use of tags in the work item component of IBM's Jazz [16], we showed that tagging was eagerly adopted by a large project team of 175 developers. Tagging had become a significant part of many informal processes such as life-cycle management and the identification of cross-cutting concerns. In this paper, we focus on one aspect identified in this earlier study: the adoption of new task categories based on tags.

Task Management Systems.

Task management systems play a key role in software development processes, and their use has been studied by several researchers. Many of these studies focus on mining and analyzing quantitative data to reveal information about a system's evolution [7] or to predict future behaviours [1, 10]. Ellis *et al.* [3] report results from interviews of how developers use Bugzilla, a popular task management system. The motivation for their study was to design a visualization tool for tasks. One of their key findings was that Bugzilla played a key role in project management. Sandusky *et al.* conducted a qualitative analysis of an open source task repository to describe how negotiation plays a role in coordination activities [13]. Bettenburg *et al.* also report a study to evaluate the effectiveness of bug reports [2].

Compared to the large number of studies that analyze data available from task management systems, there is little work on the design of task management systems. In a study on bug tracking systems, Just *et al.* [6] revealed several hurdles in reporting and resolving bugs. They present several recommendations for task management systems, including contextual assistance, reminders to add information, and assistance to collect and report crucial information to developers. By focusing on summaries of bug reports, Ko *et al.* [8] found that summaries generally describe a software entity or behaviour, its inadequacy and an execution context. They suggest new designs for more structured tasks. However, there is no previous work on using tags to identify potential task categories in task management systems.

3. RESEARCH METHODOLOGY

This section describes our research methodology by outlining our research questions, the research setting, and our three data collection methods.

Research Questions.

RQ How can tagging play a role in bridging lightweight task management and heavyweight task management?

1. What role do tags play in the adoption of new task categories?
2. How can data on the use of tags help determine the right balance between lightweight and heavyweight task organization?
3. How is software developers' use of tags for task organization different from the tag use of users outside of software development?

In this short paper, we focus on the first subquestion.

Research setting.

Our study took place with the Jazz development team from IBM. The team consists of approximately 150 contributors in about 30 functional teams, with some teams acting as sub-teams of larger teams, and many contributors assigned to multiple teams. The team members are located at 15 locations worldwide, primarily in North America and Europe. The developers have been using Jazz for more than three years and follow the "Eclipse Way" development process [4]. This process, developed by the Eclipse Development team, is an agile, iteration-based process with a focus on consistent, on-time delivery of software through continuous integration, testing, and incremental planning. We used archival data on the use of tags for development tasks, semi-structured interviews, and the project mailing lists in our study.

We focused our analysis on the Jazz development team because it was the first team to use Jazz. Therefore, we were able to access about three years worth of development data and we could examine trends over time. IBM's Jazz was one of the first development environments to add a social tagging field to their task management system.

Table 1: New categories with their possible values, and the corresponding tags

category	allowed possible values	tags
How Found	Unknown, Customer Use, Self Host, Development, Test, Internal	testing, selfhosting/selfhost/self-host
OS	Unknown, Windows, Linux, Mac OS, AIX, Z/OS, i5/OS, Other	windows, linux, mac, aix, ziseries
Client	Unknown, Eclipse client, Visual Studio client, Firefox 3.0, Firefox 3.5, Internet Explorer 7, Internet Explorer 8, Safari 3, Safari 4, All browsers, Command-line interface, Other	eclipse, firefox, ie, safari, browser

Data collection.

Our methodology followed a mixed method approach, collecting both quantitative and qualitative data to allow for triangulation. To gather quantitative data on the use of tags, we accessed the repositories of the team and extracted 65,268 development tasks with a total of 27,252 instances where a tag was applied to a development task. 1,184 unique keywords were used for these tags. The data stems from the time period from June 2006 to April 2009 and covers the development of the 1.0 release of Jazz as well as most of the development of the 2.0 release of the client component Rational Team Concert. The data used in our earlier paper on tagging [16] was a subset of the data used here.

In addition we examined the mailing lists of the Jazz development project to find out which task categories had been added to the task management system over the period of three years. We also conducted two interviews focusing on the role of tags in the adoption of new task categories. We interviewed one development manager and one developer. One of our researchers spent seven months on-site over the last two years, frequently having informal discussions with developers and observing how they use the task management system. The findings gained from our data collection are mirrored in these observations.

4. PRELIMINARY FINDINGS

Shortly after the 2.0 release of the Jazz client Rational Team Concert in June 2009, three categories were added to the task management system. Table 1 shows the category names as well as the values that can be selected for each of the categories. We analyzed the tag data to find out which of the values that could be selected for the new categories had pre-existed as tags. For most of the values, a corresponding tag existed. These tags are noted in Table 1.

The assumption that the new categories were added because of the tags was confirmed through the interviews: *"Recently we added some new fields on the work item. For the defect, we added a client and so forth. So we're starting to replace some of the tags. Deprecate some of the tags by having fields for them in most cases."* This phenomenon was also described as "training wheels": *"That seems to be almost training wheels – or it seems to be a way of trying out a field for work items. [...] Defects have different fields, platform and other things. And we used to use tags for that, but now it's been promoted. That seems to be what tags really get used for, a cheap way to add fields to work items. I'm not saying that it's bad. But I'm saying that that seems to be how it works out."*

5. DISCUSSION

One of the issues that arises from the adoption of new task categories based on tags is a large number of categories that might become overwhelming. As one of our interviewees pointed out, adding categories that are not always necessary can have disadvantages: *"We're actually at that point right now. There's a client field, I think. There's something that lists how the problem was found. So it lists five web browsers, the CLI, and the Eclipse client. And really, that field should only apply to work items that deal with the web UI. 'Cause it was added for the web UI."* It is hard to imagine that adding a new category would ever be reversed: *"I don't know how easy it would be to remove attributes from work items, though. I think that we'll never lose attributes, we'll only gain them."*

An interesting approach is taken by the way labels are implemented in the issue tracker of Google Code[4]. Their concept of labels is similar to social tagging in other task management systems. However, the Google Code issue tracker goes beyond basic labels to support key-value labels. Key-value labels contain one or more dashes, and the part before the first dash is considered to be a field name while the part after that dash is considered to be the value. For each project, a list of predefined labels and their meaning can be specified. TagSEA [15] offers similar functionality in a lightweight way. To our knowledge, the use of labels and key-value labels in the context of Google Code has not been studied yet, but it might make the transition between tags and task categories even easier.

6. LIMITATIONS

As with any chosen research methodology, there are limitations with our choice of research methods. One limitation lies in the small number of interviewees. However, one of our researchers spent seven months on-site over the last two years frequently having informal discussions with developers regarding their use of tags and the answers in the interviews were mirrored in his observations.

When the team started using Jazz, the tagging feature had not been introduced yet. This might have influenced the specific tagging behaviour. Also, given that the team was the one implementing Jazz, their willingness to adopt new technologies might be above-average. However, we have observed other teams using tags, and we believe that our conclusions also apply for teams not as biased towards using new features in Jazz. As more projects adopt Jazz or other

[4]`http://code.google.com/p/support/wiki/IssueTracker#Labels`

development environments adopt tagging, additional studies should be conducted to gain further insights.

The findings from this case study cannot be generalized beyond the scope of the particular project. However, we were able to find empirical evidence that tagging can be useful in the adoption of new task categories. At this early stage of the research, one of our goals is to explore the questions we need to ask in order to better understand the role of tagging in bridging lightweight task management and heavyweight task management.

7. IMPACT AND FUTURE WORK

We propose to examine the role of tags in bridging heavyweight and lightweight task management for software developers, and we also contribute questions for future research. Our preliminary results confirm that there are instances of tags being adopted into categories. This research can lead to improvements in the structure of tasks in task management systems. Given the central role of task management in software development, it is crucial to optimize this structure. Tags can be used to experiment with categories in order to find the ideal balance between lightweight and heavyweight task organization.

In future work, we plan to investigate this phenomenon in other development contexts. In particular, studying how developers use Google Code's approach with simple labels and key-value labels will lead to new insights with regard to task organization. We also plan to study why some popular tags are not adopted into categories and how new task categories are used compared to the corresponding tags. Another interesting avenue for future research is the application of machine learning techniques to help decide on the right balance between lightweight and heavyweight approaches to task organization.

Acknowledgments

We wish to thank the team that granted us access to their repositories and conducted interviews with us. This research is supported by a fellowship from IBM. We also appreciate comments from Lars Grammel and Nancy Songtaweesin that helped improve the paper.

8. REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *ICSE '06: Proc. of the 28th Intl. Conf. on Software Engineering*, pages 361–370, New York, NY, USA, 2006. ACM.

[2] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In *SIGSOFT '08/FSE-16: Proc. of the 16th SIGSOFT Intl. Symposium on Foundations of Software Engineering*, pages 308–318, New York, NY, USA, 2008. ACM.

[3] J. B. Ellis, S. Wahid, C. Danis, and W. A. Kellogg. Task and social visualization in software development: evaluation of a prototype. In *CHI '07: Proc. of the SIGCHI Conf. on Human factors in computing systems*, pages 577–586, New York, NY, USA, 2007. ACM.

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

[5] I. Herraiz, D. M. German, J. M. Gonzalez-Barahona, and G. Robles. Towards a simplification of the bug report form in eclipse. In *MSR '08: Proc. of the Intl. working Conf. on Mining software repositories*, pages 145–148, New York, NY, USA, 2008. ACM.

[6] S. Just, R. Premraj, and T. Zimmermann. Towards the next generation of bug tracking systems. In *VLHCC '08: Proc. of the Symp. on Visual Languages and Human-Centric Computing*, pages 82–85, Washington, DC, USA, 2008. IEEE.

[7] H. Kagdi, J. I. Maletic, and B. Sharif. Mining software repositories for traceability links. In *ICPC '07: Proc. of the 15th Intl. Conf. on Program Comprehension*, pages 145–154, Washington, DC, USA, 2007. IEEE.

[8] A. J. Ko, B. A. Myers, and D. H. Chau. A linguistic analysis of how people describe software problems. In *VLHCC '06: Proc. of the Visual Languages and Human-Centric Computing*, pages 127–134, Washington, DC, USA, 2006. IEEE.

[9] H. Ossher, D. Amid, A. Anaby-Tavor, R. Bellamy, M. Callery, M. Desmond, J. De Vries, A. Fisher, S. Krasikov, I. Simmonds, and C. Swart. Using tagging to identify and organize concerns during pre-requirements analysis. In *EA '09: Proc. of the ICSE Workshop on Aspect-Oriented Requirements Engineering and Architecture Design*, pages 25–30, Washington, DC, USA, 2009. IEEE.

[10] T. Ostrand and E. Weyuker. A tool for mining defect-tracking systems to predict fault-prone files. *IEE Seminar Digests*, 2004(917):85–89, 2004.

[11] M. P. Robillard and G. C. Murphy. Concern graphs: finding and describing concerns using structural program dependencies. In *ICSE '02: Proc. of the 24th Intl. Conf. on Software Engineering*, pages 406–416, New York, NY, USA, 2002. ACM.

[12] M. P. Robillard and F. Weigand-Warr. Concernmapper: simple view-based separation of scattered concerns. In *eclipse '05: Proc. of the OOPSLA workshop on Eclipse technology eXchange*, pages 65–69, New York, NY, USA, 2005. ACM.

[13] R. J. Sandusky and L. Gasser. Negotiation and the coordination of information and activity in distributed software problem management. In *GROUP '05: Proc. of the Intl. Conf. on Supporting group work*, pages 187–196, New York, NY, USA, 2005. ACM.

[14] 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 20th anniversary Conf. on Computer supported cooperative work*, pages 195–198, New York, NY, USA, 2006. ACM.

[15] 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.

[16] 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, USA, 2009. IEEE.