Work Item Tagging: Communicating Concerns in Collaborative Software Development

Christoph Treude and Margaret-Anne Storey

Abstract—In collaborative software development projects, work items are used as a mechanism to coordinate tasks and track shared development work. In this paper, we explore how "tagging," a lightweight social computing mechanism, is used to communicate matters of concern in the management of development tasks. We present the results from two empirical studies over 36 and 12 months, respectively, on how tagging has been adopted and what role it plays in the development processes of several professional development projects with more than 1,000 developers in total. Our research shows that the tagging mechanism was eagerly adopted by the teams, and that it has become a significant part of many informal processes. Different kinds of tags are used by various stakeholders to categorize and organize work items. The tags are used to support finding of tasks, articulation work, and information exchange. Implicit and explicit mechanisms have evolved to manage the tag vocabulary. Our findings indicate that lightweight informal tool support, prevalent in the social computing domain, may play an important role in improving team-based software development practices.

Index Terms—Tagging, collaboration, software development, task management, articulation work, work items.

1 Introduction and Motivation

Software development is among the most complicated tasks performed by humans [26]. In a typical software development process, developers perform several different activities: They use numerous 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 lot of time communicating with other members on their team. Most tools used by software developers in their daily work are tailored toward individual developers and hardly support team work. However, software is rarely developed by individuals and the success of software projects largely depends on the effectiveness of communication and coordination within teams [26].

In recent years, academia and industry have started to develop team-aware tools that support communication and cooperation in one way or another. Among those tools, there are comprehensive development environments, such as IBM's Jazz [12], and tools that only focus on certain aspects, such as groupware (e.g., INCOME/STAR [29]). As these tools are brought into the mainstream, the tension of balancing support for formal engineering practices with the informal social aspects of a team becomes obvious. Indeed, a key finding from the Computer Supported Cooperative Work (CSCW) research community is that tools that ignore emergent work practices and social aspects of a tool's use frequently fail (for example, see [17]). Thus, a challenge for the software engineering tool community is to develop tools that support both aspects.

Balancing formal and informal user needs is particularly important for task management in a sociotechnical 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. For example, Jazz has tool support for managing "work items," where a work item is a generalized notion of a development task (see Fig. 1). Work items are assigned to developers, are classified using predefined categories, and may be associated with other work items. Jazz work items also have informal tool support to address social aspects. Specifically, Jazz supports a discussion thread and a lightweight "tagging" mechanism. Using this latter feature, developers can freely associate user-defined keywords with work items.

We report the results from two empirical studies on the practice of tagging work items within Jazz. In our case studies, we examine how industrial software development teams use tags for work items, both for individual use and for collaboration. We gathered data through the inspection of the project repositories and by conducting interviews with developers on different teams. Our main contribution is the identification of different ways in which tags support informal processes in software development by filling the gap between social and technical aspects of artifacts. We explore how tagging supports collaboration in various ways. Furthermore, we examine how tagging was adapted by software developers to suit their needs and we identify potential tool enhancements.

The remainder of this paper is structured as follows: In Section 2, we discuss related work on informal processes in

• *The authors are with the Department of Computer Science, University of Victoria, PO Box 3055, STN CSC, Victoria, BC V8W 3P6, Canada. E-mail: {ctreude, mstorey}@uvic.ca.*

Manuscript received 2 Mar. 2010; revised 2 July 2010; accepted 1 Oct. 2010; published online 18 Oct. 2010.
Recommended for acceptance by J.M. Atlee and P. Inverardi.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSESI-2010-03-0058.
Digital Object Identifier no. 10.1109/TSE.2010.91.

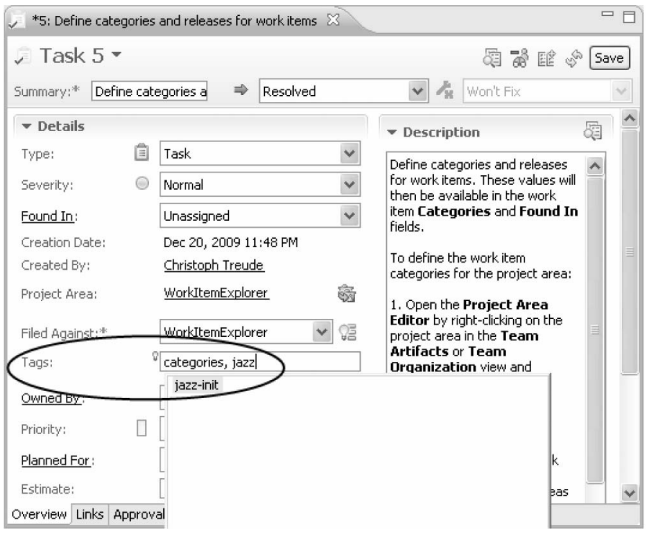

Fig. 1. Work item interface in IBM's Jazz.

software development and provide background on tagging. The tagging feature in IBM's Jazz is introduced in detail in Section 3. Our research questions are presented in Section 4 and Section 5 discusses our methodology. Sections 6 to 9 comprise the main part of this research and describe how tagging of work items has been adopted, what role it plays in software development processes, how it supports collaboration, and how tool support for tagging can be improved. The limitations of our studies are presented in Section 10. Our work is concluded in Section 11.

2 Background and Related Work

Work related to our research can be divided into two main areas: research on the social aspects of software development and research on tagging and its adoption in software engineering. Our work can be interpreted as the intersection of these two areas: using tags to support social aspects in software development.

2.1 Social Aspects in Software Development

As mentioned previously, software development is recognized to be one of the most challenging management tasks performed by humans [26]. The larger systems become and the more complicated the compositions of the developing teams are, the more obstacles there are in the way to 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" [28] must be supported in a software engineering project. According to Gerson and Star [14]: *"Articulation consists of all tasks needed to coordinate a particular task, including scheduling subtasks, 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.

Various challenges related to social aspects in software development have been identified. These include dealing with strategic and cultural issues [22], longer development times when coordinating with remote colleagues [21], dealing with communication breakdowns such as unclear dependencies, circular dependencies, and schedule changes [5], and managing plan failures [34]. In distributed projects, managing implicit knowledge [27], maintaining awareness [18], and leveraging expertise [11] can also impact the success of a project. These many challenges that arise in team-based software development can be addressed by better awareness tools and processes, improved communication practices, implicit and explicit knowledge management, as well as support for articulation work.

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. [8] claim that source code is both a social and a technical artifact and that dependencies not only exist between artifacts but also between developers. In a previous study on source code annotations, Storey et al. [40] report on how annotations are used to document both technical and articulation activities. Grinter [16] also describes how configuration management tools are sometimes co-opted for articulation work, despite the fact that they have significant shortcomings in supporting articulation 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 assumptions about the work flow.

Several researchers have studied how teams use issue tracking systems[1] to support their processes and for managing articulation work. Many of these studies focus on mining and analyzing quantitative data to reveal information about the evolution of the system [24] or to predict future behaviors [2], [30]. Ellis et al. [10] report results from an interview of how developers use Bugzilla, a popular bug tracking system. The motivation for their study was the design of a visualization tool for tasks. One of their main findings was that Bugzilla played a key role in managing the project. Sandusky and Gasser conducted a qualitative analysis of an open source bug repository to describe how negotiation plays a role in coordination activities [35]. Bettenburg et al. also report a study to evaluate the effectiveness of bug reports [3]. By focusing on summaries of bug reports, Ko et al. [25] found that summaries generally describe a software entity or behavior, its inadequacy, and an execution context. They suggest new designs for more structured tasks.

Although researchers have considered how bug repositories and issue tracking systems are used for coordinating work, researchers have not thus far considered how tagging can be used to support informal activities by a team coordinating tasks. De Souza et al. [9] conducted an ethnography with a software development team and found that tools often create a distinction between private and

1. Such systems are also referred to as defect or bug tracking systems.

public aspects of development. To close this gap, several informal practices are adopted in order to manage interdependencies between both perspectives. Similarly through this paper, we wish to consider how tagging is used to bridge the gap between the technical and social aspects of work item management. But first we review related research on tagging and discuss how tagging is currently used in software development.

2.2 Tagging and Software Development

The concept of tagging, as it is currently used, comes from the social computing domain. Social computing technologies, sometimes referred to as Web 2.0, are seeing rapid adoption by emergent communities on the web. Key examples include Facebook,[2] YouTube,[3] as well as community-based recommender systems such as CiteULike,[4] TripAdvisor,[5] and Flickr.[6] Tagging is used by many of these systems and is often referred to as social bookmarking. The success of tags is closely related to their bottom-up nature: 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 way, and they stand in contrast to formal top-down classification mechanisms.

A tagging system consists of three main components [23], [33]: tag users, the tags themselves, and the objects being tagged. In most social tagging systems that have been studied thus far, the items being tagged are often heterogeneous and may come from a very large pool of uncontrolled resources. The typically large number of creators and users of the tags also tend to be from a very large uncontrolled population, with varying levels of expertise. Most systems keep track of who tagged which object, useful metadata which can be used to infer the interests of a particular user as well as count how many times a given tag is assigned to an object (thus providing a way to reinforce the relevance of tags assigned).

Golder and Huberman [15] and Hammond et al. [19] provide overviews of tagging systems and classify the main reasons for user tagging. A common finding across these studies is that users tag to provide information on an artifact (e.g., what an artifact is or to refine a category) and for organizing artifacts. A more detailed study was done with the photo sharing website Flickr [1]. Robu et al. [33] examine data from Delicious,[7] a social bookmarking site, to describe the dynamics of collaborative tagging systems. Their findings indicate that despite the unsupervised tagging by individual users, coherent and rich categorization schemes emerge especially for specialized domains such as complexity science. Heymann et al. analyzed the social tagging of books and found that the tagging system was fairly well consistent, of high quality, and complete [23]. Sen et al. [36] explore how tagging is used by a community using the MovieLens recommender system. Sen et al.'s research questions focus on how personal tendencies and community influence the creation of tags. Part of the success of tagging comes from allowing users to define their own vocabulary [13]. Information retrieval is also enhanced by community tagging [33].

The introduction of tags into software development raises the question of how the informality of tagging affects the process of developing software and how a typical software development process can take advantage of the characteristics of tags. Tagging is not a new concept to software engineering; however, earlier forms of tagging are not consistent with the social computing notion of tagging today. Many early uses of the word tagging in software engineering systems relied on a preexisting controlled vocabulary. Tags have been used for decades for annotating check-in and branching events in software version control systems, as well as for documenting bugs in bug tracking systems. Also, Brothers' ICICLE was an early exploration of tag-like structures with a limited, controlled vocabulary during code inspection [4].

Due to these inconsistencies on the term tagging, we define a tag as follows: *A tag is a freely chosen keyword or term that is associated with or assigned to a piece of information. In the context of software development, tags are used to annotate resources such as source files or test cases in order to support the process of finding these resources. Multiple tags can be assigned to one resource.* We use the term **tag keyword** to indicate the term that is used (e.g., `usability`), and the term **tag instance** to indicate instances of the tag keyword being applied to one or more resources.

Tagging, as defined here, has not been extensively researched in a software engineering context. Some systems support social bookmarking, for example, Code Snippets[8] and ByteMyCode.[9] They support social tagging of source code, but require the user to post code fragments on public servers before tags can be applied. To aid program comprehension, Hassan and Holt [20] propose annotating static dependencies in source code using sticky notes that contain content recovered from source control systems. A recent tool that intersects social tagging with software development is described by Storey et al. [39]. Their tool Tags for Software Engineering Activities (TagSEA) [38] is a collaborative tool to support software development and uses the ideas of social tagging to support coordination and communication. A case study [37] showed that TagSEA provides the user-defined navigation structures that are lacking in traditional task annotations. In contrast to this bottom-up approach, the tools Concern Graphs [31] and ConcernMapper [32] enable developers to associate parts of source code with high level concerns.

Apart from these studies, there is little research on how the lightweight mechanism of tagging can play a role in supporting informal activities in software development. The research described in this paper examines the current use of tags for task management in software development projects with the aim to identify potential tool enhancements. For a more general discussion of lightweight tool support for work activities, we refer to work by Churchill and Bly [6].

2. http://www.facebook.com/.
3. http://www.youtube.com/.
4. http://www.citeulike.org/.
5. http://www.tripadvisor.com/.
6. http://www.flickr.com/.
7. http://delicious.com/.
8. http://snippets.dzone.com/.
9. http://www.bytemycode.com/.

3 Tagging in Jazz

Jazz is an extensible technology platform that helps teams integrate tasks across the software life cycle. The software development team collaboration tool built on top of Jazz is called Rational Team Concert (RTC). Developers using Jazz organize their work around so-called work items which can be interpreted as development tasks. A typical work item as shown in Fig. 1 consists of a unique number, summary, description, state, work item type, severity, and priority; the component it was filed against, the version it was found in, the creator, and several other details that are optional. The primary way to organize work items in Jazz is to use the category hierarchy. The category for each work item is identified by filling out the *Filed Against* field. A work item can only be in one category and the available categories are defined per project by the development manager. As can be seen in Fig. 1, there is an optional tag field in which developers can insert an arbitrary number of tag instances per work item. The Jazz content assistant suggests tag keywords with a common prefix that have been used before. If a developer adds a tag keyword that has not been used before, a pop-up window appears and asks if this keyword should be added to the vocabulary. Tag instances are public to all members of a project team across all components.

Compared to online media tagging and social bookmarking, tagging of work items in Jazz is different in some important ways. In Jazz, the items being tagged are strictly homogeneous work items that are created by members of the Jazz community. Typically, creators of Jazz work items and tags have some expertise on the underlying software project and would not be classed as casual users. Jazz also has the group concepts of team and project within the community; many tagging systems operate at the level of individual and community only. Another potentially important difference is in terms of the metadata associated with the tag instances and keywords. As mentioned above, most tagging systems record the user(s) that attached a particular tag instance to a resource. In Jazz, the tag instances are added directly to a work item and information on when the tag instance was added and by whom is not easily accessible (only through the work item's history). Tag instances can also be removed from work items. A tag instance can only be attached "once" to a given work item and the creator of the tag instance is not visible in the default view.

Thus, we may expect our findings to be somewhat different from the existing results in this area. Our research questions that explore how the Jazz work item tagging feature supports collaborative software development are listed in the next section.

4 Research Questions

1. How is the social tagging mechanism adopted by developers for annotating work items?
    a. How does the frequency of new tag instances vary over the lifetime of a project?
    b. How many work items are tagged?
    c. How many users tag and how does this number vary over time?
2. What characteristics of tags are prevalent in the tagging of work items?
    a. Which tag keywords are applied more frequently?
    b. What are the different categories of tag keywords that emerge during a project?
3. What role does the tagging feature play in the work practices of individual and team developers?
    a. Are work item tags used for individual and/or collaborative use?
    b. Why do developers tag work items?
    c. How do developers use tags?
    d. How are tags managed?
    e. How does a team reach consensus on the tag vocabulary?

5 Methodology

In the following paragraphs, we outline the setting of our research as well as the three data collection methods we used: inspection of archival data available in repositories, semistructured interviews with software developers, and ethnographic-style observations.

5.1 Research Setting

Our study took place with several professional development teams from IBM.

5.1.1 Case Study 1: Jazz

Our first case study was conducted with the Jazz development team. The team consists of approximately 175 contributors and about 30 functional teams, with some teams acting as subteams of larger teams and some contributors assigned to multiple teams. The team members are located at 15 locations worldwide, primarily in North America and Europe. The developers of the team have been self-hosting their development since early 2006, and they follow the "Eclipse Way" development process [12]. This process, developed by the Eclipse Development Team, is an agile, iteration-based process with a focus on consistent, on-time delivery of quality software through continuous integration, testing, milestones, and incremental planning. At the time of our data collection, the developers were working on the 2.0 release of Rational Team Concert, and they were using the latest milestone builds of RTC for their development.

5.1.2 Case Study 2: Enterprise Infrastructure (EI)

We replicated our study with a large project team[10] of more than 1,000 members working on four interrelated projects. They had been using Jazz for about one year and develop systems mostly for enterprises. They are part of IBM, but not connected to the Jazz development team. The development processes used by these teams range from Scrum to conventional methods. At the time of our study, the teams were using RTC 1.0.

10. The project name is obfuscated for confidentiality reasons.

TABLE 1
Data Extracted from Repositories

case	data object	amount
Jazz	Work items	65,268
	Tag instances applied to work items	27,252
	Tag instances removed from work items	2,452
	Number of unique tag keywords	1,184
EI	Work items	50,826
	Tag instances applied to work items	13,588
	Tag instances removed from work items	758
	Number of unique tag keywords	673

5.2 Data Collection

Our methodology follows a mixed method approach, collecting both quantitative and qualitative data. In order to gather quantitative data on the use of tags in the project, we accessed the repositories of the development teams and extracted all relevant information. The amounts of data extracted for both case studies are shown in Table 1.

Qualitative data were collected through a series of interviews with developers and through ethnographic-style observations. All interviews were semistructured, allowing for follow-up questions and clarifications. Most of the questions were aimed at understanding the details of why and how developers use tags.[11] In total, 12 interviews were conducted: six for each case study. For the Jazz case study, we interviewed the development manager **J-M**, the project administrator **J-A**, one component lead **J-C**, and three developers on one team **J-D1**, **J-D2**, and **J-D3**. For the EI case study, we interviewed one product design lead **EI-L**, a development manager **EI-M**, a release engineer **EI-R**, and three developers who also occasionally take the role of scrum master **EI-D1**, **EI-D2**, and **EI-D3**. All interviews were conducted in-person at an IBM location and lasted about 30 minutes each.

In addition, the first author spent seven months at the Jazz site and two weeks at the EI site as part of an ethnographic study. He frequently had informal discussions with developers regarding their use of tags and the answers in the interviews were mirrored in his observations. The observations were recorded using ethnographic field notes. The quantitative nature of our repository analysis and the qualitative nature of the interviews and observations provided insights for all of our previously posed research questions.

5.3 Data Analysis

We developed a Jazz plug-in to extract the data related to tags from the repositories of all development teams in our studies. The pertinent data we extracted contain all work items, along with their IDs, creators, creation times, owners, summaries, descriptions, priorities, severities, and several other fields. In addition, we extracted the following data for each instance of a developer applying a tag to a work item: the time that the tag instance was created, the tag keyword that was used, the time of creation, and the creator. Instances of tag keywords being removed from a work item were also extracted. We created our ConcernLines tool [41] to help us understand the use of tag keywords over time. ConcernLines supports the cognitive process of understanding how the concerns expressed through tags interrelate by visualizing co-occurring tag keywords over time.

We coded tag keywords to identify the categories of keywords that emerged over the duration of the projects. We coded the tag keywords individually first, and then confirmed our codes in several collaborative coding sessions between the two authors of this paper in which we considered everything we know about each tag keyword before assigning codes to it. The coding was done using the bottom-up inductive technique of Corbin and Strauss [7]. Although there are some findings on the categories of tags for social bookmarking systems (as mentioned earlier), we expected to see very different categories emerge in our study of tagging; thus, we did not start our coding process with an initial set of codes. During the tag coding process, we considered all 12 interviews we conducted, and we also followed up on keywords that we were not able to categorize through e-mail discussions with three of our participants (J-A, EI-R, and EI-D2). In addition, we read summaries and descriptions of the work items that were tagged with particular keywords to confirm our classification. For the Jazz project, we also accessed the project internal mailing lists as well as the documentation available on the project website.[12] Based on these codes, we identified more abstract categories in which we grouped tag keywords using similar codes. Compared to our earlier work on tag usage in the Jazz project [42], we were able to identify additional categories and also to refine the classification.

We also coded the interviews in collaborative sessions. For some of the research questions such as "Why do developers tag work items?" the answers we considered were mainly the answers to that particular question in our interviews. For other research questions, in particular the ones regarding the collaborative aspects of work item tagging, themes emerged through the assignment of codes to quotes and grouping of codes.

For each interview snippet, sometimes multiple codes would apply (e.g., consensus, externalization). We then grouped the quote segments and extracted the most prominent themes that appeared repeatedly in our interview data. When exploring the interview data, we made use of the tagging data to help us in the interpretation of the quotes. For quotes that were unclear, we would check with Jazz and EI team members on our understanding of their tagging processes. In our analysis of both the tagging keywords and the interviews, our field notes from the ethnographic observations were crucial in helping us make sense of the data.

6 Adoption of Tagging

To answer our first research question on the adoption of tags, we performed an analysis of new tag instances over time, looking at both the number of tag instances that are

11. A list of sample questions we asked in the interviews is available at http://tinyurl.com/WITagging.

12. https://jazz.net/.

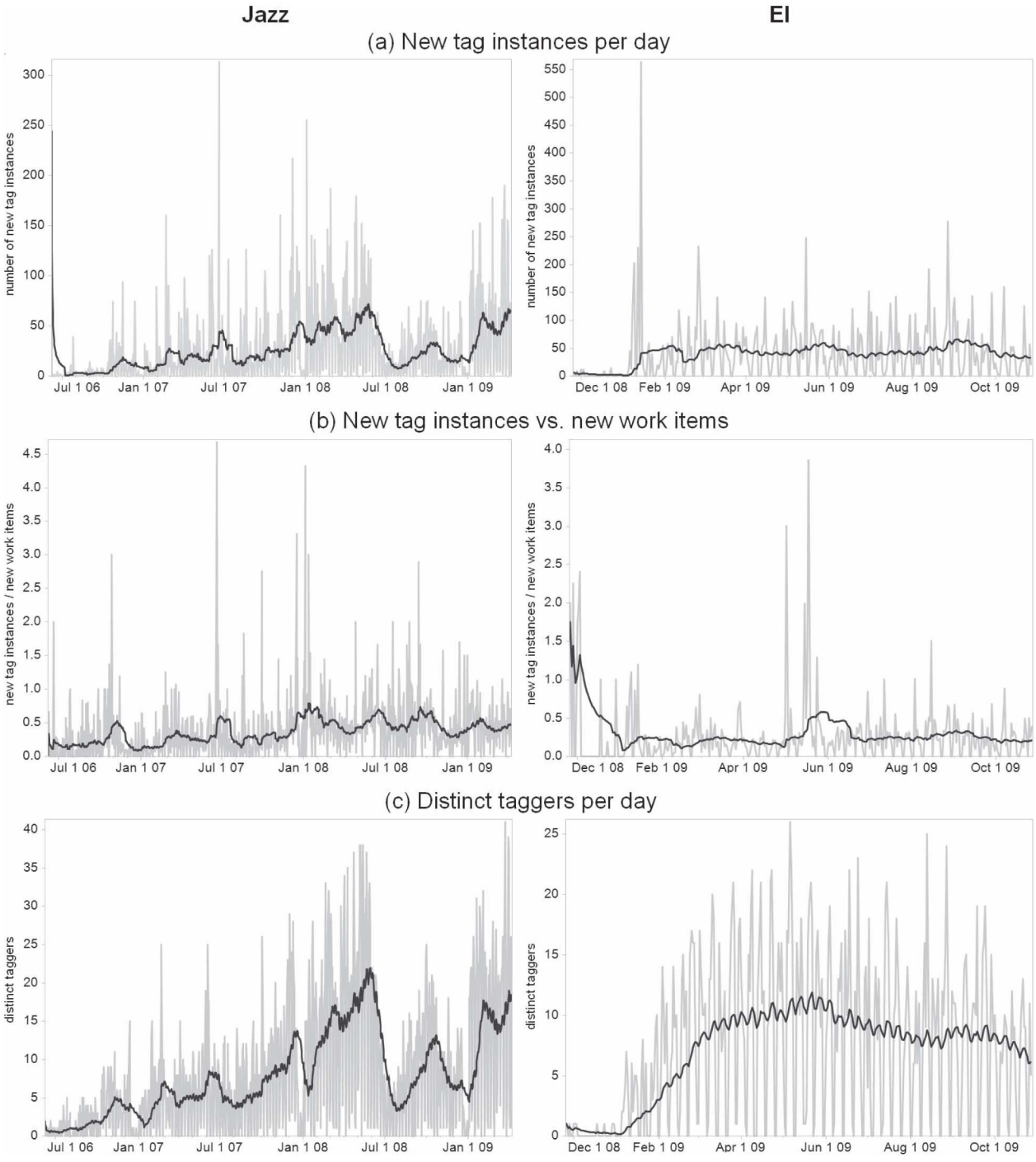

Fig. 2. Graphs showing the rate of new tag instances and distinct taggers over time for Jazz and EI. (a) New tag instances per day. (b) New tag instances versus new work items. (c) Distinct taggers per day.

applied to work items and the number of individuals tagging work items.

6.1 Frequency of New Tag Instances

Fig. 2a shows how the number of tag instances added per day evolves over time in both case studies. The gray line depicts the actual numbers per day; the black line gives the value averaged over the last 30 days at any point of time. The moving average line was added to allow for easier visual interpretation. The graphs are not significantly different when calculating the average for longer or shorter time intervals. For the Jazz project, the number increases until mid-2008, then drops, and increases again toward mid-2009. Both mid-2008 and mid-2009 marked the two major releases of Jazz. Apart from high tagging activity in the beginning, the rolling average of tag instances in the EI case study is stable at around 40 instances per day. Spikes are mostly related to planning activities such as coordinating which work items should be included in a particular release.

To see the extent to which the number of new tag instances depends on the number of new work items, we calculated the ratio of new tag instances to new work items per day as

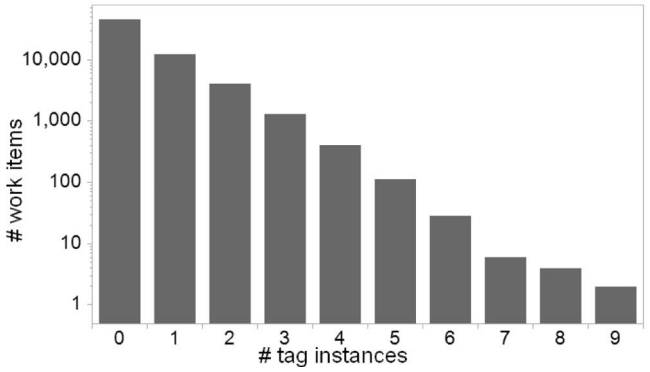

Fig. 3. Distribution of tag instances to work items (log scale) in Jazz.

shown in Fig. 2b.[13] For both case studies, the rate does not change substantially over time, apart from a few spikes.

6.2 Distribution of Tag Instances to Work Items

About 28.5 percent of all work items in Jazz and about 18.8 percent of all work items in EI have been tagged at least once. The distribution of tag instances to work items is shown in Fig. 3 for Jazz. The distribution of tag instances to work items for EI follows the same pattern.

6.3 Number of Tag Users

The number of individuals applying tag instances to work items over time follows a similar pattern as the number of tag instances. As shown in Fig. 2c, there are peaks of up to 40 different individuals applying tag instances on the same day in Jazz, and the only major discontinuities in distinct users per day occur around the Christmas holidays and after the release in mid-2008. For EI, the number of distinct users per day is between 5 and 10 on average, with peaks of up to 25 users.

In Jazz, 360 contributors have applied at least one tag instance to a work item. Out of 299 contributors who owned work items in the last three years, 176 (59 percent) applied at least one tag instance to a work item. In addition, the project has a web portal that allows clients to submit new work items. There were 184 individuals from outside the company applying tag instances through this web portal. However, the main tag users were team members from inside IBM. The top 50 most prolific taggers applied between 125 and up to more than 3,000 tag instances, using about 100 different keywords. In EI, 314 (29 percent) contributors have applied at least one tag instance to a work item. In total, the EI project has 1,082 members. The top 25 most prolific taggers applied between 150 and 800 tag instances, using about 50 different keywords.

These statistics indicate that tags were used continuously after their initial introduction and that software developers found them helpful enough to keep using them over a period of three years. More details on tag usage in support of informal processes and collaboration are given below.

7 Characteristics of Tag Keywords

This section describes the characteristics of the tag keywords that our studies revealed. To address this topic, we looked at our analysis of the tag keywords and instances as our primary data source, but also used interviews (and follow-up discussions) to explain and confirm our findings.

TABLE 2
Tag Keywords with the Most Instances in Jazz

tag keyword	# instances
`polish`	966
`svt`	870
`ux`	668
`tvt`	636
`testing`	565
`globalization`	442
`usability`	441
`maintenancecandidate`	436
`errorhandling`	431
`mustfix`	421

TABLE 3
Tag Keywords with the Most Instances in EI

tag keyword	# instances
external library	601
product X	573
`id`	481
`teamb`	466
`a11y`	433
`docs`	427
product Y	420
component	403
`dev`	357
`conformance`	336

7.1 Most Frequently Applied Tag Keywords

In Jazz, 1,184 different keywords have been applied in the time frame of our case study (May 2006-April 2009). Table 2 shows the 10 most frequently applied tag keywords in Jazz. Details on the use of tag keywords over time are given below (see also Fig. 5).

In EI, 673 different keywords have been applied in the time frame of our case study (November 2008-October 2009). Table 3 shows the 10 most frequently applied tag keywords. Italicized keywords are obfuscated for confidentiality reasons.

In both case studies, the distribution of instances to tag keywords has the shape of a "long tail." The long tail distribution has been frequently observed in tagging systems [33], [36].

7.2 Different Kinds of Tag Keywords

Our classification of the tag keywords from both case studies reveals that different kinds of tag keywords exist in the projects. Unlike tagging in other applications such as tagging of photos on Flickr, tags for work items do not necessarily describe the content of the tagged item. EI-R describes: *"If you look at* `dev` *and* `teamb`*, they're not that descriptive. And* `conformance`[14] *wasn't really used for that—it doesn't really describe what the team itself is working on as opposed to the nature of that specific work item."* EI-D2 adds: *"They don't necessarily describe the content of the work item, they might link a bunch of [work items] together in a way that wouldn't be obvious just from looking at one work item, but*

13. Undefined values resulting from a division by 0 on days with no new work items are represented as 0 for simplicity reasons.

14. The keyword conformance was used to indicate work items related to supporting third party systems such as browsers.

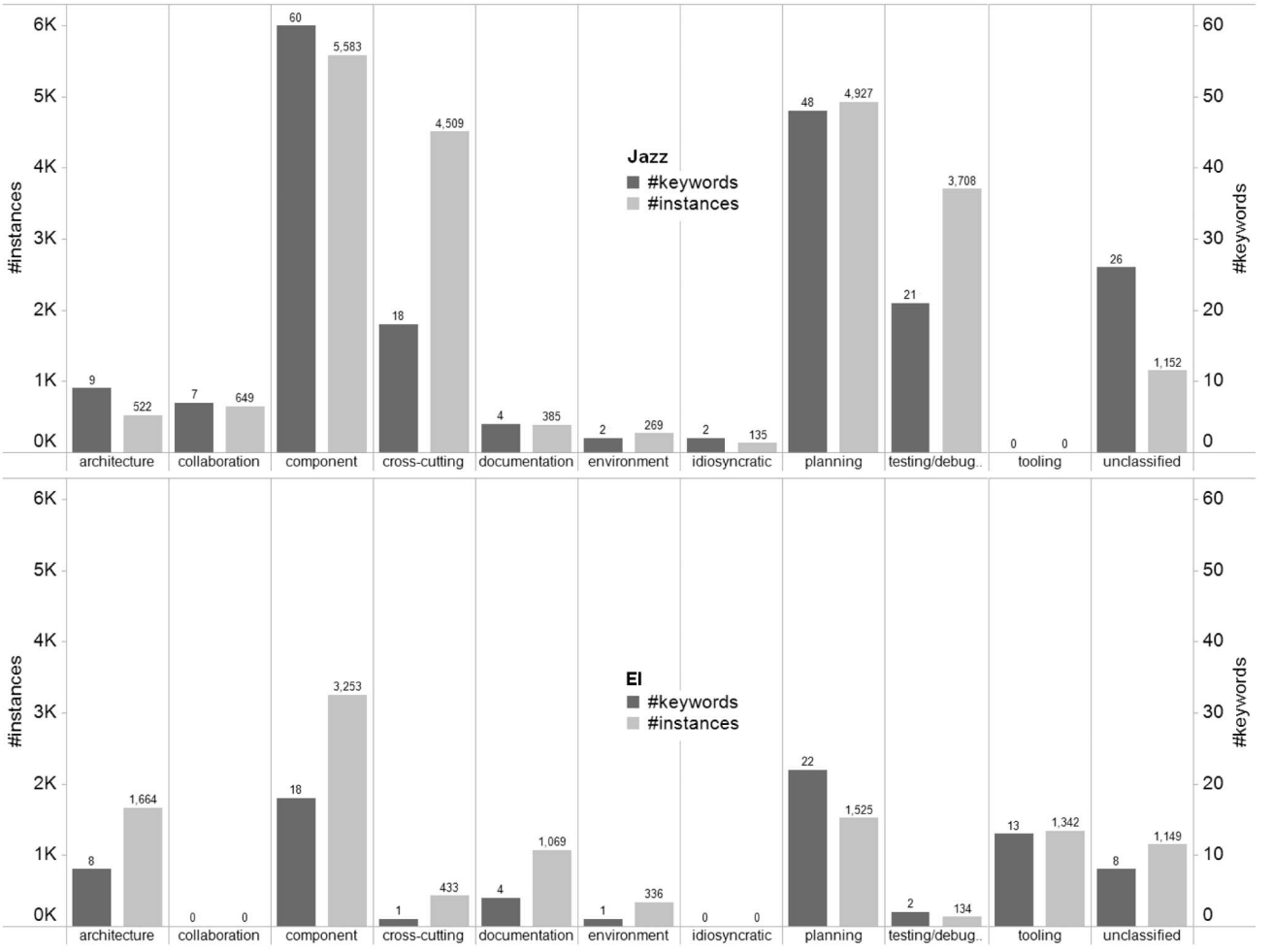

Fig. 4. Number of tag keywords and instances per category (note: different scales for #keywords and #instances).

because I know that they kind of all belong to the same initiative or the same project, they make sense that way. [. . .] Sometimes, it's more metainformation than the actual item itself."

Fig. 4 shows the results of our classification. We classified the keywords accounting for the "heads" of the "long tails," i.e., all keywords that accounted for 80 percent of all tagging instances in the data. Thus, we classified the 197 most used keywords from the Jazz data and the 77 most used keywords from the EI data. The keywords we classified had at least 28 instances each for the Jazz data and at least 34 instances for the EI data.

We identified 10 categories that apply to both case studies as shown in Fig. 4. We assigned one category to each tag keyword. An interesting finding is that despite the ethnographic-style observations, reading the work items, interviews with 12 participants, and follow-up e-mail discussions, there were tag keywords that we were unable to classify. Those keywords are shown in the rightmost column—only 26 for the Jazz project and 8 for the EI project. We describe each category of the classified tag keywords in the following sections.

7.2.1 Architecture

In both data sets, tags are used to mark work items related to architecture. These work items are either about the overall architecture of the product or about integrating the product with other products. The EI teams depend on external libraries more than the Jazz team. The tag keyword most used in the EI project is the acronym for an external library as explained by EI-L: *"That is a company that we were contracting to [. . .]. So, we're using their libraries and their library is called [acronym]."*[15] Other keywords that fall into the architecture category are `third_party`, `vs.net`, and `arch`.

7.2.2 Collaboration

All tagging done in Jazz is inherently collaborative as each developer can see all tag instances applied by other developers and all resources are shared among the entire team. Most tag keywords relate to some kind of technical concern—a component, a requirement, or documentation. However, some of the tag keywords are about collaboration, i.e., they are used to coordinate collaborative processes within the development team or to communicate with other developers.

We found evidence for this in the Jazz data: The keyword `no_code` (and the synonym `non_code`) is used to flag work items that only include changes to messages, images, or JavaDoc, but not source code. Shortly before releases, this keyword is used to communicate to other developers that working on that particular work item will not affect the

15. We do not identify the acronym for confidentiality reasons.

functionality of the system as it does not touch the source code. Another example is `fixready`. This keyword was used by one developer to indicate that the fix for a particular work item was ready, but had not yet been committed due to a missing approval for a related work item.

In the Jazz project, tags are frequently used to coordinate changes between different component teams. J-A explains: *"`Adoption` means that there's a work item that's in our bucket or another bucket for which there's a change set attached that someone in another team has to adopt."*

7.2.3 Component

The tag field is also used to refine the work item categories that Jazz already provides. Compared to the categories, component-specific keywords can be introduced without any effort or official conventions: *"He could've made another heading for each of [the subcategories]. But, for some reason I guess, the tagging was probably more open. I guess when you go and modify something like the spec, something like that; it feels very administrative, [whereas] this tagging is supposed to be more fluid"* (J-D3). Component-specific keywords are usually used to categorize work items and their use depends largely on the presence of other categorization mechanisms. In both data sets, component-related keywords account for the highest number of tag instances.

7.2.4 Crosscutting Requirements

Unlike component-specific tag keywords, crosscutting keywords capture aspects of work items that crosscut the hierarchy of categories for work items, as J-A explains: *"[Crosscutting tags] are orthogonal to categories. They are—that's the beauty of tags—that they are crosscutting. It's not about grouping, when we have grouping, things can only be in one."* Crosscutting tag keywords can be distinguished into functional requirements and nonfunctional requirements. Examples for nonfunctional requirements include keywords such as `performance`, `accessibility`, `scalability`, or `responsiveness`. On the other hand, functional requirements that crosscut several components include `internationalization` and `errorhandling`. The use of tags for crosscutting concerns was more frequent in the Jazz data.

7.2.5 Documentation

In both projects, tags are used to identify work items that are documentation related. In the Jazz data, there are keywords for `doc` and `documentation` (synonyms) and keywords that help with the compilation of the "new and noteworthy" for each release. The keywords relating to documentation in the EI data include `id` (information delivery), `docs`, and `samples`. An example that shows how powerful the tagging mechanism is for tasks that require some metadata but that do not require a formal process is given by J-D1: *"I went through a bunch of things that were tagged with `faqable` or `faq` or something like that, so then when I was done in order to see what I've done, I tagged it with `included_in_faq`."*

7.2.6 Environment

Software products have to be adjusted to work on particular browsers or particular operating systems. In both data sets, a small number of tag keywords has been used to indicate work items that are related to compatibility. In the Jazz data, these keywords include `linux` and `ziseries`. Developers in EI use the keyword `conformance` as EI-L explains: *"So, `conformance` would be tasks related to supporting particular things like say, a particular browser, or a particular database vendor, or a particular app. So, product managers would come up with a required conformance."*

7.2.7 Idiosyncratic

Among the tag keywords that we classified, two keywords in the Jazz data set stood out as being idiosyncratic. They are neither related to a milestone nor are they used to organize work items according to components or crosscutting concerns. They are used for various reasons and are usually only used by very few developers. The two idiosyncratic keywords in the Jazz data are `selfhosting` and `rfe`. `Selfhosting` was used in the beginning of the project when Jazz started to be self-hosted. When the entire project became self-hosted, the keyword became unnecessary, but at the time, it was the easiest way of marking work items that related to the self-hosting aspect in particular. The other keyword is `rfe`, which is used to flag work items that were created by the support team for official customer requirements. Again, this is not something that could be expressed through other work item features, such as priority and severity, but nevertheless was important to record.

7.2.8 Planning

Tags are used heavily for planning purposes in both case studies. Keywords such as `beta2candidate` and `committed-sprint-8` show whether a certain work item is a candidate to be included in an upcoming release or whether it has been committed toward that release already. It is interesting to note that the number of instances per tag keyword for planning tags is much higher in the Jazz data. Since the EI teams mostly use Scrum, their "releases" are a lot more frequent—and include fewer work items.

In both case studies, one particular keyword was used to flag work items that definitely had to be included in a particular release. In Jazz, the keyword used for these work items is `mustfix`; in the EI data, it is called `mandatory`. J-M explains: *"I have to say easily, the most used and the most useful tag has been the `mustfix` tag, right. Especially when we'll sort of—we're working on some effort and there's limited resources, limited time, and it's like OK, do we really need to fix this thing or not, right. Irregardless of all the other fields in the work item that tell you information about that defect, right. Bottom line is, do we need to fix it or not. And to be honest, this `mustfix` tag is usually set by the development manager, right, based on discussion with other people. And interestingly, sometimes you see in the work item, they'll sort of argue, well not argue, but they'll ask, they'll say does this really have to be `mustfix`, right."*

Work items are annotated with planning tag keywords extensively only during a specific period of time as they are related to a milestone in the development process and usually have the name of this release in their name, e.g., `beta2candidate`. In Fig. 5, a screenshot of our ConcernLines tool [41] shows the time lines of the most used tag keywords in the Jazz project. Color is used to show the

Fig. 5. Screenshot of the ConcernLines tool: Time line lengths of most used tag keywords in Jazz.

intensity of tag use on a particular day.[16] Planning tag keywords such as `beta2candidate` have a relatively short time line, whereas other keywords such as `polish` and `ux` (user experience) have been used throughout the entire project. Compared to the other kinds of keywords, planning related tags are transient.

7.2.9 Testing/Debugging

Tags are also used to coordinate the testing process. In the Jazz data, keywords such as System Verification Test (`svt`), Translation Verification Test (`tvt`), and Function Verification Test (`fvt`) are prominent, and other keywords such as `buildstatus` are used to indicate how a certain bug was found; as described by J-D1: "*`Buildstatus` is flagging work items that I've created while I [was a release engineer] that have something to do with the current status of the build. So, if it's broken and I'm complaining, I flag it with `buildstatus`.*" Keywords such as `include_in_testplan` help coordinate the testing process, and `review` is used for work items that contain reviewing work rather than development work.

Testing and debugging plays a less prominent role in the EI tag data. The only two keywords that fit into this category from the EI project are quality control (`qc`) and `testing`.

7.2.10 Tooling

The developers in our EI case study switched to using Jazz in the middle of their projects and they did not start using all features of Jazz right away. Work item tags were helpful in that situation as they allow the developers to create processes based on ad hoc artifacts. The keywords that we classified as tooling are almost exclusively annotations that could have been made to the work items in Jazz through different tool features. For example, there were at least 42 instances for each of the keywords `defect`, `enhancement`, and `bug`. Tag keywords were also used instead of team areas, as explained by EI-L: "*[Our project] has four teams: team a, team b, team c, and team d. And each of those, I think we've started off the practice of adding those items and tagging them. And I think that might have dropped off for some of the other teams, but the team b, it looks like they stuck with it.*" `Teamb` is the fourth most used keyword in the EI data. In contrast, tooling-related tag keywords have not been used by the Jazz team.

16. Colors are shown as shades of gray in this paper.

8 The Role of Tagging

In this section, we discuss the findings for our third research question on the role of the tagging feature in the work practices of individuals and teams. To answer this question, we look primarily at our data from the interviews and from the ethnography-style observations. We use the analysis of the archival data to further explain and confirm our findings from the interviews.

8.1 Tagging Audience: Self, Team, and Community

Although tagging in Jazz is by design a collaborative feature, as is the case in most tagging systems, it is difficult to verify if tagging is done to service collaborative or individual needs. In web-based tagging systems, collaboration is supported [1], [36]. In the previous section, we saw that at least one category of tag keywords attached to work items is explicitly about collaboration, e.g., `non_code` (communication) and `fixready` (coordination). For other categories of tag keywords, our interviews reveal that these support social as well as individual activities. This category of tag keywords is not as evident in systems such as Flickr because the focus in these systems is on navigation and categorization, rather than on supporting articulation work which is a crucial aspect of collaborative development.

Some developers predominantly tag to service pressing **individual needs**, but also will use them to support collaboration within the team. As J-D1 says: "*I primarily create them for myself. But with the candidates, for example, that's obviously for someone else's consumption.*" However, other developers see tagging as more of a **team activity** than

an individual one, as J-D3 says: *"I don't personally tag work items for myself that much. But I know when I was doing the testing for the [component] that [J-D1] wrote and [he] had basically a tag for each command, so I would follow—I would add his tags, like his conventions, cause I figured it would be easier for him, right. And I'm already on the work item when I'm creating it, so—I would add those as I was creating them."* We also saw another instance of where a new team creates a list of the tags already used in a project. Furthermore, J-A discusses how tagging is not just for the team: *"It's a bit for everybody, for the team as a whole and for people on the outside."*

From the interviews, we were able to glean that tag instances are added by feature owners, team managers, and also other stakeholders such as release managers. For example, this quote from EI-R demonstrates how an agreed-upon keyword called `mandatory` was used to coordinate work across various team roles: *"We had a lot of features—and we had to determine which ones are mandatory for the release. And they went through an exercise through all the features, and for the ones that are mandatory, they actually use the tag `mandatory`. So product management, development mostly, and as well the execution team."* This same developer further goes on to discuss how tags may be used to support the articulation work of breaking a task into subtasks and to support communication from management about high level features: *"It's mostly the management, but as well, there is nothing that says development won't do it. They are the ones mostly who break some of the stories or work items into smaller items, and they will tag it appropriately if needed. But mostly the tags are at the high level, like features and so on, which is mostly feature owner, the scrum master, or the manager of the component."*

From our analysis of the archival data, we can see that in addition to team members, 184 **community** members also tag work items in the case of Jazz. However, the smaller participation by the general community members sets apart the kind of tagging done here from the tagging performed in Flickr and Delicious.

8.2 Tagging Motivation: Categorization and Organization

The predominant reason for the use of work item tags is **categorization**. As J-C put it: *"Mainly as a kind of categorization. [...] Tags are useful for identifying crosscutting concerns like performance or accessibility or scalability or responsiveness, things like that, or testing."* While the Jazz interface already provides an opportunity to categorize work items (see the *Filed Against* field in Fig. 1), tags are more flexible. The category tree can be altered, but this would change the available categories for the entire team and does not work for crosscutting concerns, as EI-D2 notes: *"If we're organizing a team and we have a bunch of work items that will span different projects or different logical organizations, tag them all together, so that I can make one query for all those types of things. [...] I find it works well for things that don't quite fit into a nice tree hierarchy."* J-D3 also identified this disadvantage of the top-down classification: *"The problem is its very administrative-side feeling, which is fine, except it's not as flexible to just ad hoc make things."*

The developers we interviewed also recognize the benefits of tagging over a strict categorization scheme (which is available for Jazz work items but is strictly controlled). This one quote from EI-D2 captures how tags are useful for capturing varying **viewpoints**: *"No, the category tree you couldn't make it perfect, because my definition of perfect would be different from your definition of perfect and even then, I don't think in a tree. So, I have a hard time when browsing a large piece of information where I have to know how somebody else would categorize it in order to find it. [...] Different companies and different users and even people within this company are going to use it—and different teams—are going to use it differently. So, I don't think you can build sort of a one size fits all type model for it."*

Tags are also seen as a way to **organize** work items. J-D1 responded: *"[I use tags] because I feel like [work items] should be organized. I feel like they're there and so I should use them. [...] I don't know if they do organize work items, but it makes me feel like I'm doing something when I associate a tag with it. In theory, I'd like to believe that tags draw work items that have a similar area together. That's my hope."* The organization achieved through tags is different from other categorization mechanisms as described by EI-D2: *"It allows me to kind of organize the work items in a way that is very free form and flexible and so I can write queries that fit what I'm looking for, rather than the 1,700 different fields that there are and trying to figure that out. [...] The tag stuff is nice and free form, and allows me to think the way I want to think."*

On the other hand, developers are not forced to use the tagging feature of work items. For example, EI-D3 explains: *"I didn't find it useful. It didn't give me any additional information."*

8.3 Tags in Use: Finding Work Items, Articulation Work, and Information Exchange

The main use case for tags is **finding** work items later as described by J-A: *"I use [tags] to categorize things basically, so I can have queries and find things. I'm afraid of losing work items if they aren't [tagged]."*

In particular, the developers in charge of assigning work items to other developers use the tagging feature. In the agile processes followed by the EI team, this kind of **articulation work** is the job of the scrum master. This role rotates between developers. EI-D1 describes the process: *"I have [...] to organize their stories, their tasks, etc. Trying to find things that fall into their work category. I need to do searches on them to pull that stuff out of the backlog to propose it to the team. And [in] doing so, this is the type of queries that I would use the tags for. [...] A lot of times you can get that information through queries of just the title or the description, but I also like to do the tags, cause if somebody actually did use them, then it'd be good. [...] So I don't go searching for those things. I don't go out trying to specifically enter these things I should say. But when I'm creating queries and such like that, I do look to see what's available and I will use that."* In this case, tags help for exploring the work items: *"They have led me on paths to find—or think about other things to search for. So, for example, if I do a [...] search on it, and a [certain tag appears, then I] read up on that particular item, figure out what it's involved in, and then from that, I can do other queries that find items in my component that will base on that."* Similarly, the release engineer EI-R uses tags as well: *"I rely completely on using these [accessibility] tags [...] to pick up on these requests. [..] And we know when the request comes, through the queries."*

In the Jazz project, the administrator J-A searches for tag keywords: *"For nonfunctional things, like usability—Okay, we got two weeks to do some things, what are the usability related enhancements that we have, for example. So, we have a set of work items, that are enhancements, and some are usability related. So, I look at those, I go 'Oh, these are easy. Let's try and fix these four.' For example, during the polish phase of the 1.0 release, we had two weeks to polish. [...] So, we had tags like* `polish` *and* `usability` *and I use that to kind of guide what work items we could work on."*

But tags are also used for queries by developers who are not in charge of assigning bugs. For example, J-C describes: *"I used them the other day, trying to search for [...] a list of bugs against [a related product]. Thinking that, you know, I was probably a good boy and had tagged any [product] related issues with the [product] tag and did a search for [product] tags and that actually found very few. [Laughs] Cause I hadn't been a very good boy and tagging my Jazz work items with the [product] tag."*

A specific case of tag use in queries are the dashboards in Jazz that are displayed and configured using the web interface. Dashboards are intended to provide information at a glance and to allow easy navigation to more complete information. By default, each project and each team within a Jazz project have their own dashboard, and an individual dashboard is created for each developer when they first open their web interface. A dashboard consists of several viewlets. Viewlets are rectangular widgets displaying information about some aspect of a project. Developers can add viewlets to their dashboards and configure the viewlets using different parameters.[17] J-A reports: *"We have a lot of dashboards that are tag-based, like the test teams, favorite bugs, and so on. Having a tag lets us have those [dashboard] viewlets that otherwise—it would be really hard to describe a query that says, 'show me all the work items that were added by the test team'."* J-M introduced a particular keyword to ensure visibility in the project's dashboard: *"There's things that we specifically track. I introduced this* `tracking` *tag, so there's a* `tracking` *tag that we put on certain kinds of work items which actually then show up in a dashboard. [...] There's something where we just want to raise the visibility."* As J-A describes, tags increase awareness: *"On my team anyway, if I think it's related to one of those characteristics like performance, I tag it because I want to be aware of it and I want to have a query that shows me what they are."*

Many of the tag categories discussed previously (such as component and crosscutting requirements) are to add information to work items. From the interviews, we were able to discern that these tag keywords also played an awareness role in **informing** others and being informed about work items: *"My use for tags mostly is just to get an idea of what other work items are about. [...] I think the PMC [Project Management Committee] likes to put like* `1.1candidate` *or* `0.6candidate`*, so like that—that rarely impacts me, but I see it and then I'm just a little bit more aware, so that okay, sure, someone finds that important in that respect"* (J-D3).

17. For more details on the use of dashboards in Jazz, see [43].

8.4 Tag Management: Keeping Track of Tags, Removing Tags, and Tag Structures

In our interviews, we learned how teams of developers developed mechanisms or informal processes to help manage the tags they use on a project.

For example, EI-D2 discussed how he **externalized** lists of tag keywords to help in maintaining some consistency around tags used: *"I have been keeping a list of tags so that I don't, you know, create duplicates, things that are spelled alike or that sound alike, or two different ways of referring to the same thing."* Externalizing lists of tag keywords was also used as a mechanism by a new team to learn which tags were used in a project: *"There was a new team that joined and they were like, what tags should we use? We don't actually have them written down anywhere. So, they went and they did compile a list—I don't know where they put it, they put it somewhere on the wiki—right, of the tags. Well, they asked me and I said, here's [...] off the top of my head [...] about 5 or 6 that we use a lot, right"* (J-M).

We discussed how tag instances are removed based on our analysis of the archival data. Here, we see that developers consider team members and may be reluctant to do so when faced with possibilities for **removing tags**: *"I don't believe I ever have [removed tags] and I'd probably be reluctant to. And it would—if I were to—it would probably have to be something that I own or am deeply involved with, right. Because, again, if my presumption is, if they're more like a supplementary thing, if a tag is wrong, I'm not going to actively go out and say, no that's wrong and fix it, I'm going to let whoever owns it make that call"* (J-D3). We see that the developers consider owners of work items should manage and remove tag instances on those work items: *"I would only [remove tags] if I attached the tag to the work item or if I owned the work item. I wouldn't want to mess with—I don't know—someone else's categorization"* (J-D1).

Tag instances are not removed often however, since work items do not show up in queries anymore by default once they have been closed. As EI-R describes: *"We use [the tags], then we just forget about them."* EI-D2 explains: *"Once I tag something, usually what happens is I close the work item and it'll remain tagged because it's still part of whatever that is. [...] I mean unless I made a mistake or mistyped something, I can't think of a time where I actually went in and said, 'Oh this a stupid tag' and removed a pile."*

There are only two scenarios that were mentioned by our interviewees where tag instances would get removed from work items. The first one is about the status of the work item as described by J-A: *"I'm tagging these things for candidates, [and then I find out] they're not [candidates] and I untag them all."* The other example is given by J-C: *"If I don't like the way somebody tagged it or there's a better tag for one that they used, I'd remove it and add another one. But that doesn't actually happen terribly often."* This issue of removing tag instances is not something that occurs in Flickr or Delicious because the tag instances are not directly attached to the resources as they are in Jazz.

J-M discussed the need to manage the complexity of **tag structures** that were emerging in the tagging vocabulary. Indeed, from the archival data, we could see that some component tag keywords as well as some testing keywords

had refinement keywords associated with them (e.g., `testing.performance`). J-M discussed both of these structures and specified a need to keep on top of how that structure was being decided about in e-mail discussions: *"I wasn't involved in the discussion, I was watching the e-mail about it and then they decided that, no they were just going to use the tags instead, and then listed all the 10 different tags and so in fact I was thinking about that this morning, OK, I need to get in on that discussion, saying, no, I don't really like it that way."*

8.5 Vocabulary Consensus: Explicit, Implicit, and Tolerance

An important aspect in understanding the advantages and disadvantages of tagging systems, especially in terms of representing knowledge and task coordination, is if the community using the system will converge on a common vocabulary that is useful.

We found that sometimes consensus is reached in an **explicit** way, e.g., through e-mails, meetings, and wikis, and that an explicit consensus is particularly important for the planning tag keywords. We saw an example of this above in terms of the `mandatory` tag keyword. Several more cases on how planning tag keywords were agreed on emerged in the interviews. Two examples include: *"I usually tag with a tag whose name I've been told. [...] Or an e-mail that's sent out. So, `m1candidate`, for example"* (J-D1). *"I already know this—he had set a convention, so I was following the convention. Other than that, the other tags, I kinda see or pay attention to, is during milestone releases or candidate releases"* (J-D3). In addition to planning tag keywords, we observed that component keywords were explicitly agreed on as well, for example, as explained by J-C: *"Either myself or the team lead would establish a tag for the area, so—in the [component] team I established one called `workspaceeditor` and, you know, I would use that consistently and other people would start using it as well once they saw it."*

When consensus is reached in an explicit means, we also heard how the list of available tag keywords is externalized either in a wiki or list to encourage their use, but at the same time not insist that they are used as described by EI-D2: *"Well, I guess that I keep the central list of—and that list is in a publicly available area. So, I assume other people do look at it and I've had a few questions on it so I think other people look at it but I can't tell you whether everyone does and everyone follows it—and frankly I wouldn't want it to be that hard and fast a rule. It's supposed to be kind of a loose system, so I keep that for my own convenience."*

Although some keywords are agreed on in an explicit way, we also observed that tag keywords are frequently agreed on through an **implicit** means. This interview quote from EI-R demonstrates how both explicit and implicit mechanisms for consensus occur: *"Maybe the only one that at least I know of has a convention is the one that I put because I make it tight to the integration build that we are in, so we know where the approvals went. But other than that, if it's legal or other than that, there's not really conventions—just a word that means something, and whoever is basically working on these work items, knows what it means."*

Implicit awareness of tag options can occur through watching work item feeds: *"You just sort of see it happening in the work item's feeds"* (J-C), or through content assist: *"There's always the concern about when you're creating tags, are people going to create more tags. I mean at least the good thing right now is that when you create a tag it tells you that you're creating a new tag"* (J-M). Consensus can also be achieved through the context and existing knowledge underlying the use of the tag keywords: *"If you're part of the organization—it would be odd that somebody would use a tag within the organization that means nothing to somebody in the organization. [...] All of the tags that I have seen that are there, I'm aware of what the acronym means, what the word means, in my context"* (EI-D1).

To further facilitate implicit understanding, the developers will take steps to make sure that the tag keywords they create will be comprehensible to other team members: *"I make sure I don't use too many acronyms, so that people can understand what the tag means. Apart from that, I try to make it somewhat descriptive, so [...] I wouldn't use `perf`, instead use `performance`"* (J-A). There is also trust that this process will work: *"Anything that makes sense, intuitively to me, I just go ahead and tag it. Hopefully what makes sense intuitively to me will make sense to other people as well"* (EI-D2).

This last quote also captures that there is not a large concern if tag keywords are misunderstood, and that there is **tolerance** for some variation in keywords. This is important because there may be some concern that ambiguity of tag keywords and the use of synonyms may lead to problems in using tagging systems [33] but our interviews do not reveal such issues: *"And then, after having seen something for so many times—not that I would necessarily know if there's a difference between, you know, slight variations in tag naming, but if I've got an idea, I'll do it. And then again, I wouldn't be too afraid if I got it wrong, cause someone will just change it, right"* (J-D3).

Tables 4 and 5 show the most shared tag keywords for both projects.

TABLE 4
Most Frequently Shared Tag Keywords in Jazz

tag keyword	#instances	#distinct users
`performance`	413	46
`globalization`	442	45
`tvt`	636	45
`polish`	966	43
`maintenancecandidate`	436	40
`no_code`	197	40
`errorhandling`	431	38
`usability`	441	38
`beta2candidate`	308	35
`rc4candidate`	133	33
`ux`	668	33

9 Discussion: Implications on Tool Design

One of the goals of our research is to contribute to the development of tool support for collaborative software development, especially with regard to tagging. In this section, we discuss how tool support for tagging could be broadened for other social and technical artifacts, and how such tool support can be improved.

While tags have already been adopted by the software developers in our study, we propose there are still areas

TABLE 5
Most Frequently Shared Tag Keywords in EI

tag keyword	#instances	#distinct users
id	481	30
ally	433	24
docs	427	24
mandatory	185	21
conformance	336	20
defect	132	17
product X	113	15
third_party	300	15
product Y	79	14
all	205	13
teamb	466	13

where tool support for tagging can be improved. However, the eagerness with which tags have been adopted and the experiences of our interviewees suggest that the lightweight nature of tags has to remain intact. EI-D2 explains: *"Part of the beauty of it, I think, is that you only have to enter, you know, a word or two, and that's it. [...] I think keeping it light is good. It's actually probably the best about it, frankly. [...] The minute you start making me tag stuff, I'm going to resent it. [...] I find that the tags are a good way to reduce the amount of time that I'm looking for particular things—then it's worth the investment, but the minute I have to do something, then it just becomes a drag."*

Therefore, enhancements of tool support should recognize the current benefits of tags and the main theme of any changes to tool support should be to help developers use tags. We suggest the following tool enhancements:

- Using the same lightweight approach as with tags for work items, **a tag property could be added to other kinds of artifacts**, especially source files, test cases, and requirement documents. Tags can also be implemented on a fine-grained level, e.g., for methods and fields. This would enable tagging across different types of content and thus would further support collaborative organization of artifacts. Similar ideas have been successfully tried and tested in TagSEA [39]. Tagging for builds has recently been added to Jazz, but the tagging systems are treated entirely separately. It is not possible to get all work items and builds tagged with the same keyword through a single query.
- Display **tag authors** along with the tag instances. During our interviews, we showed our interviewees a list of tag keywords that were used on their work items but that they did not apply themselves. We discovered that our participants used the tag authors to understand the keywords. Adding the author information of tag instances is not obtrusive as the information is collected anyway and could just be displayed on mouse over.
- Apart from author information, the only metadata property that should be added to tags is an **optional description**. For tag keywords that do not have an obvious meaning such as `adoption` or `buildstatus`, a short description would increase the usefulness as there are tag keywords in the vocabulary that may be unfamiliar to some developers. When a new keyword is introduced to the vocabulary, a dialog could ask for an optional description instead of just notifying developers that they are about to introduce a new keyword.
- Current tools do not offer any **management** for tags on work items. Useful functions would be changing all tag instances with a particular keyword, e.g., to fix spelling mistakes. For synonymous keywords such as `doc` and `documentation`, folding would be beneficial. Similar refactoring mechanisms have been implemented in TagSEA [39].
- To increase understanding of how tags are used and which tags are suitable for work item search, a way of **externalizing tags** should be added. Information about tag keywords and instances, their authors, the corresponding work items, and the time of the tag instance creation is available in the system, but this information is not used yet by the work item tooling. The explicit mechanisms and externalization activity for reaching tag consensus among all users are not activities that are likely to occur with users of Flickr or Delicious. Ames and Naaman report that two of their participants coordinated tags for photos on Flickr with others in order to facilitate later search and retrieval [1]. However, the consensus described here in the context of Jazz goes beyond that and includes all members of a project.
- Once the tag vocabulary is analyzed, **tags for incoming work items could be suggested**. Strong candidates for suggestions are tag keywords that have extensively been used in the near past such as planning related keywords and keywords that have been applied to work items in the same category.

When tags were initially introduced in Jazz, several additional features were suggested by developers. However, over the time of more than three years of tagging activity, developers adapted to the tagging tool support as it was initially implemented, J-A says: *"In the beginning, I thought that we should do a lot, have private tags, have more metadata with them. In the hindsight, their simplicity is kind of interesting."* Therefore, it is important to not introduce barriers and focus tool enhancements on metadata that can be automatically collected such as author names.

However, for a very small subset of five to 10 keywords, namely, the ones indicating the operating system and the ones indicating how a bug was found, more formal tool support has recently been added to the Jazz project. We investigate how tag keywords that are frequently used over a long period of time reveal the need for additional predefined categories of keywords in task management tool support in a recent short paper [44].

10 Limitations

As with any chosen research methodology, there are limitations with our choice of research methods. When the Jazz team started on their project, the tagging feature for work items had not been introduced yet. This might have influenced the specific tagging behavior. Also, the Jazz developers might be biased toward their own tool and their usage pattern might be different from other developers.

However, we were able to confirm our findings with other development teams that are much larger than the Jazz team. There might still be bias, as all development teams in our study were part of IBM. However, the team members of the EI case study are not related to the Jazz project and work in a completely different domain.

In our analysis, we only analyzed the "head" of the "long tail" of the tag distribution that accounts for 80 percent of all tag instances. Although the remaining tag keywords may not have been used frequently, there may have been very important categories in there that should be examined. However, a fine-grained analysis of all keywords is beyond the scope of this paper. An alternative could have been to code a random sample of the tag keywords. We decided to focus on the most frequently used ones, but considered all tag keywords mentioned during our interviews.

Our interpretation of the tag categories could have potential errors. We addressed this issue by asking about specific tag keywords and instances in our interviews, by reading the summaries and descriptions of the corresponding work items, through follow-up e-mails to our participants, and by searching the project websites and pertinent mailing lists. Potential errors are also offset by the first author's ethnographic-style observations that were conducted on the Jazz site for seven months and at the EI site for two weeks. Compared to our earlier work on how work item tags are used in the Jazz project [42], we were able to identify additional categories and also to refine the classification. These enhancements are based on additional data from the Jazz project, entirely new data from the EI project, and an additional eight interviews.

IBM's Jazz is still new and it is one of the first software development environments supporting tags for development tasks. Thus, we were only able to get data from Jazz users. As more projects adopt Jazz or other development environments adopt tagging, additional studies should be conducted to gain further insights into the use of tags in software development.

11 Conclusions and Future Work

The main contributions of this paper are the identification of the various ways in which tagging supports informal processes in software development as well as concrete suggestions for tool improvements.

While there are many formal processes in place for technical artifacts, managing social artifacts and articulation work is only supported by informal processes if there is any process at all. Informal processes are usually carried out via communication mechanisms. In order to understand software development as a whole and in order to provide appropriate tool support, we have to understand both the technical and the social aspects of software development. Tags are one way to look at the informal side of software development in a team setting. Through understanding how developers use tags in their daily work, we can extend our knowledge on informal aspects of software development and furthermore understand how a social computing technology, such as tagging, is adapted by software developers.

Our research has shown how the social computing mechanism of tagging has been adopted and adapted by two large software development teams. Not only is tagging used to support informal processes within the teams, it has also been adapted to the specific needs of software developers. Different kinds of tags have emerged over the duration of a software project for processes that require metadata but are not formalized, ranging from architecture and planning to collaboration and testing. The main advantages of using tags in software development are their flexibility and their lightweight, bottom-up nature. While fields such as *Operating System*, *Milestone*, or *Crosscutting Concern* could be part of fixed schemata, this would add overhead for work item creators and owners. Tags add the same functionality without implying administrative changes.

With the shift to team-based software development and the corresponding increasing importance of articulation work, informal processes, and communication mechanisms, social computing mechanisms such as tagging may play an important role beyond work items. They may be used to organize, manage, and categorize software artifacts in general in an informal and collaborative way. Future work lies in the examination of the benefits of social computing mechanisms in other areas of software development.

Collaborative tagging implies an underlying social structure. We are currently exploring which social networks emerge in software development between authors of work items, owners of work items, and tag authors. This will increase our understanding of team dynamics in software development and may ultimately result in better collaborative software development tool support.

Acknowledgments

The authors would like to thank the teams that granted them access to their repositories and conducted interviews with them. This research is supported by a fellowship from IBM and funding from NSERC. The authors also appreciate the comments from Lars Grammel, Nancy Songtaweesin, Jamie Starke, and the anonymous reviewers that helped improve the paper.

References

[1] M. Ames and M. Naaman, "Why We Tag: Motivations for Annotation in Mobile and Online Media," *Proc. SIGCHI Conf. Human Factors in Computing Systems,* pp. 971-980, 2007.

[2] J. Anvik, L. Hiew, and G.C. Murphy, "Who Should Fix This Bug?" *Proc. 28th Int'l Conf. Software Eng.,* pp. 361-370, 2006.

[3] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What Makes a Good Bug Report?" *Proc. 16th ACM SIGSOFT Int'l Symp. Foundations of Software Eng.,* pp. 308-318, 2008.

[4] L. Brothers, V. Sembugamoorthy, and M. Muller, "ICICLE: Groupware for Code Inspection," *Proc. ACM Conf. Computer-Supported Cooperative Work,* pp. 169-181, 1990.

[5] M. Cataldo, M. Bass, J.D. Herbsleb, and L. Bass, "On Coordination Mechanisms in Global Software Development," *Proc. Int'l Conf. Global Software Eng.,* pp. 71-80, 2007.

[6] E.F. Churchill and S. Bly, "It's All in the Words: Supporting Work Activities with Lightweight Tools," *Proc. Int'l ACM SIGGROUP Conf. Supporting Group Work,* pp. 40-49, 1999.

[7] J.M. Corbin and A. Strauss, "Grounded Theory Research: Procedures, Canons, and Evaluative Criteria," *Qualitative Sociology,* vol. 13, no. 1, pp. 3-21, 1990.

[8] C. de Souza, J. Froehlich, and P. Dourish, "Seeking the Source: Software Source Code as a Social and Technical Artifact," *Proc. Int'l ACM SIGGROUP Conf. Supporting Group Work,* pp. 197-206, 2005.

[9] C.R.B. de Souza, D. Redmiles, and P. Dourish, "'Breaking the Code', Moving between Private and Public Work in Collaborative Software Development," *Proc. Int'l ACM SIGGROUP Conf. Supporting Group Work,* pp. 105-114, 2003.

[10] J.B. Ellis, S. Wahid, C. Danis, and W.A. Kellogg, "Task and Social Visualization in Software Development: Evaluation of a Prototype," *Proc. SIGCHI Conf. Human Factors in Computing Systems,* pp. 577-586, 2007.

[11] S. Faraj and L. Sproull, "Coordinating Expertise in Software Development Teams," *Management Science,* vol. 46, no. 12, pp. 1554-1568, 2000.

[12] R. Frost, "Jazz and the Eclipse Way of Collaboration," *IEEE Software,* vol. 24, no. 6, pp. 114-117, Nov./Dec. 2007.

[13] G.W. Furnas, T.K. Landauer, L.M. Gomez, and S.T. Dumais, "The Vocabulary Problem in Human-System Communication," *Comm. ACM,* vol. 30, no. 11, pp. 964-971, 1987.

[14] E.M. Gerson and S.L. Star, "Analyzing Due Process in the Workplace," *ACM Trans. Information Systems,* vol. 4, no. 3, pp. 257-270, 1986.

[15] S. Golder and B.A. Huberman, "Usage Patterns of Collaborative Tagging Systems," *J. Information Science,* vol. 32, no. 2, pp. 198-208, 2006.

[16] R.E. Grinter, "Supporting Articulation Work Using Software Configuration Management Systems," *Computer Supported Cooperative Work,* vol. 5, no. 4, pp. 447-465, 1996.

[17] J. Grudin, "Groupware and Social Dynamics: Eight Challenges for Developers," *Comm. ACM,* vol. 37, no. 1, pp. 92-105, 1994.

[18] C. Gutwin, R. Penner, and K. Schneider, "Group Awareness in Distributed Software Development," *Proc. ACM Conf. Computer Supported Cooperative Work,* pp. 72-81, 2004.

[19] T. Hammond, T. Hannay, B. Lund, and J. Scott, "Social Bookmarking Tools (I): A General Review," *DLib Magazine,* vol. 11, no. 4, pp. 1-23, http://www.dlib.org/dlib/april05/hammond/04hammond.html, 2005.

[20] A.E. Hassan and R.C. Holt, "Using Development History Sticky Notes to Understand Software Architecture," *Proc. 12th Int'l Workshop Program Comprehension,* pp. 183-192, 2004.

[21] J.D. Herbsleb, A. Mockus, T.A. Finholt, and R.E. Grinter, "An Empirical Study of Global Software Development: Distance and Speed," *Proc. 23rd Int'l Conf. Software Eng.,* pp. 81-90, 2001.

[22] J.D. Herbsleb and D. Moitra, "Guest Editors' Introduction: Global Software Development," *IEEE Software,* vol. 18, no. 2, pp. 16-20, Mar./Apr. 2001.

[23] P. Heymann, A. Paepcke, and H. Garcia-Molina, "Tagging Human Knowledge," *Proc. Third Int'l Conf. Web Search and Data Mining,* pp. 51-60, 2010.

[24] H. Kagdi, J.I. Maletic, and B. Sharif, "Mining Software Repositories for Traceability Links," *Proc. 15th IEEE Int'l Conf. Program Comprehension,* pp. 145-154, 2007.

[25] A.J. Ko, B.A. Myers, and D.H. Chau, "A Linguistic Analysis of How People Describe Software Problems," *Proc. Visual Languages and Human-Centric Computing,* pp. 127-134, 2006.

[26] R.E. Kraut and L.A. Streeter, "Coordination in Software Development," *Comm. ACM,* vol. 38, no. 3, pp. 69-81, 1995.

[27] T.D. LaToza, G. Venolia, and R. DeLine, "Maintaining Mental Models: A Study of Developer Work Habits," *Proc. 28th Int'l Conf. Software Eng.,* pp. 492-501, 2006.

[28] P. Mi and W. Scacchi, "Modeling Articulation Work in Software Engineering Processes," *Proc. First Int'l Conf. Software Process,* pp. 188-201, 1991.

[29] A. Oberweis, T. Wendel, and W. Stucky, "Teamwork Coordination in a Distributed Software Development Environment," *Proc. GI Jahrestagung,* pp. 423-429, citeseer.ist.psu.edu/oberweis94teamwork.html, 1994.

[30] T. Ostrand and E. Weyuker, "A Tool for Mining Defect-Tracking Systems to Predict Fault-Prone Files," *IEE Seminar Digests,* vol. 2004, no. 917, pp. 85-89, 2004.

[31] M.P. Robillard and G.C. Murphy, "Concern Graphs: Finding and Describing Concerns Using Structural Program Dependencies," *Proc. 24th Int'l Conf. Software Eng.,* pp. 406-416, 2002.

[32] M.P. Robillard and F. Weigand-Warr, "Concernmapper: Simple View-Based Separation of Scattered Concerns," *Proc. OOPSLA Workshop Eclipse Technology eXchange,* pp. 65-69, 2005.

[33] V. Robu, H. Halpin, and H. Shepherd, "Emergence of Consensus and Shared Vocabularies in Collaborative Tagging Systems," *ACM Trans. Web,* vol. 3, no. 4, pp. 1-34, 2009.

[34] K. Rönkkö, Y. Dittrich, and D. Randall, "When Plans Do Not Work Out: How Plans Are Used in Software Development Projects," *Computer Supported Cooperative Work,* vol. 14, no. 5, pp. 433-468, 2005.

[35] R.J. Sandusky and L. Gasser, "Negotiation and the Coordination of Information and Activity in Distributed Software Problem Management," *Proc. Int'l ACM SIGGROUP Conf. Supporting Group Work,* pp. 187-196, 2005.

[36] S. Sen, S.K. Lam, A.M. Rashid, D. Cosley, D. Frankowski, J. Osterhouse, F.M. Harper, and J. Riedl, "Tagging, Communities, Vocabulary, Evolution," *Proc. 20th Anniversary Conf. Computer Supported Cooperative Work,* pp. 181-190, 2006.

[37] M.-A. Storey, L.-T. Cheng, J. Singer, M. Muller, D. Myers, and J. Ryall, "How Programmers Can Turn Comments into Waypoints for Code Navigation," *Proc. IEEE Int'l Conf. Software Maintenance,* pp. 265-274, 2007.

[38] 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. Software Eng.,* vol. 35, no. 4, pp. 470-483, July/Aug. 2009.

[39] M.-A. Storey, L.-T. Cheng, I. Bull, and P. Rigby, "Shared Waypoints and Social Tagging to Support Collaboration in Software Development," *Proc. 20th Anniversary Conf. Computer Supported Cooperative Work,* pp. 195-198, 2006.

[40] 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," *Proc. 30th Int'l Conf. Software Eng.,* pp. 251-260, 2008.

[41] C. Treude and M.-A. Storey, "ConcernLines: A Timeline View of Co-Occurring Concerns," *Proc. 31st Int'l Conf. Software Eng.,* pp. 575-578, 2009.

[42] C. Treude and M.-A. Storey, "How Tagging Helps Bridge the Gap between Social and Technical Aspects in Software Development," *Proc. 31st Int'l Conf. Software Eng.,* pp. 12-22, 2009.

[43] C. Treude and M.-A. Storey, "Awareness 2.0: Staying Aware of Projects, Developers and Tasks Using Dashboards and Feeds," *Proc. 32nd Int'l Conf. Software Eng.,* pp. 365-374, 2010.

[44] C. Treude and M.-A. Storey, "Bridging Lightweight and Heavyweight Task Organization: The Role of Tags in Adopting New Task Categories," *Proc. 32nd Int'l Conf. Software Eng.,* pp. 231-234, 2010.

Christoph Treude is working toward the PhD degree in computer science at the University of Victoria and is an organizer of the workshop on Web 2.0 for Software Engineering (Web2SE). In his PhD research, he is exploring the role of emergent knowledge structures in collaborative software development. He has already studied the use of tags, dashboards, feeds, and a community portal by professional software developers using IBM's Jazz.

Margaret-Anne Storey is a professor of computer science at the University of Victoria, a Canada research chair in human computer interaction for software engineering, and a principal investigator for the US National Center for Biomedical Ontology. Her research goal is to understand how technology can help people explore, understand, and share complex information and knowledge. She applies and evaluates techniques from knowledge engineering, social software, and visual interface design to applications such as collaborative software development, program comprehension, biomedical ontology development, and learning in web-based environments.