Determining Relevancy: How Software Developers Determine Relevant Information in Feeds

Thomas Fritz and Gail C. Murphy
University of British Columbia
{fritz,murphy}@cs.ubc.ca

ABSTRACT

Finding relevant information within the vast amount of information exchanged via feeds is difficult. Previous research into this problem has largely focused on recommending relevant information based on topicality. By not considering individual and situational factors these approaches fall short. Through a formative, interview-based study, we explored how five software developers determined relevancy of items in two kinds of project news feeds. We identified four factors that the developers used to help determine relevancy and found that placement of items in source code and team contexts can ease the determination of relevancy.

Author Keywords
News feeds, relevancy, context, interview study.

ACM Classification Keywords
H.5.3 Group and Organization Interfaces.

General Terms
Human Factors.

INTRODUCTION

Information streams—tweets, status updates from social networking services and other RSS feeds—enable users to stay aware of information that might be relevant to them. Users of such information streams face two problems. First, they must determine which streams contain information of information. Second, in many cases, not all of the information in a stream is relevant, forcing the user to separate the wheat from the chaff. Malone identified this second problem in the context of distribution lists more than twenty years ago [5].

Surprising to us, there appears to be very little formative research about the question of how to determine relevant information in a stream. Instead, the research has focused on how to recommend items within a stream, largely based on topicality; for example, the approach of Chen *et al.* [1] recommends tweets by matching terms recorded in a profile of the user and terms in the tweets. Schamber *et al.* [7] argue that a determination of relevance should go further and should involve understanding how users perceive information relative to their current situation. They suggest the use of more qualitative research methods to better understand how users determine which parts of information are relevant.

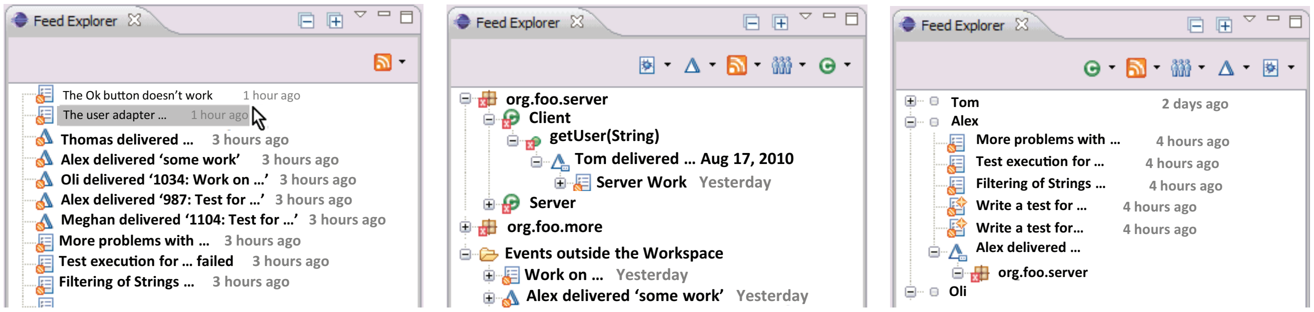
In this paper, we take this direction and present the results of a small formative study that investigated two questions: how does a user determine relevance and can the context of the situation be used to help a user to determine relevance? Our aim was to allow the formation of data-driven hypotheses that can be subjected to more detailed testing. We chose to study a team of software developers because this situation allowed us to study multiple individuals working with similar information items in a similar work environment.

Our formative study involved a team of five developers who use IBM's Rational Team Concert (RTC) environment. RTC includes a default feed reader that provides updates about several information streams, including changes to shared work items and changes to shared code. Over the period of a week, we interviewed each developer two times, presenting each with a subset of the information items in their feed reader and asked them questions related to the relevance of those items. In all, we studied 291 news items.

We found that each user had their own definition of relevance, ranging from an item being relevant because it impacted the individual's work in the present to more generic determinations that the item may help to do one's job in the future. We also found that users use the kind of information stream, the content of an item in the stream, the target of the content, their relationship with who created the item and whether or not the news had been discussed elsewhere to determine relevancy of individual items. Finally, we found that presenting news items in the context of a fragment of a developer's work appears to hold promise for easing relevancy determination. These findings lead to hypotheses that can inform future studies and that can suggest new approaches for automated support for managing information streams.

RELATED WORK

How users determine relevance of information has been considered in terms of email systems (e.g., [10]). Two of the factors for determining relevance identified in our study overlap with these findings, specifically "content" and "relation with creator". Our work independently finds these factors without presupposing that message characteristics are the only way of exploring relevancy. In addition, our work identifies that the structure of information, such as links between work items and change sets, can also play an important role in relevance determination. Another group of research showed



```
Feed Explorer
The Ok button doesn't work   1 hour ago
The user adapter ...   1 hour ago
Thomas delivered ...   3 hours ago
Alex delivered 'some work'   3 hours ago
Oli delivered '1034: Work on ...'   3 hours ago
Alex delivered '987: Test for ...'   3 hours ago
Meghan delivered '1104: Test for ...'   3 hours ago
More problems with ...   3 hours ago
Test execution for ... failed   3 hours ago
Filtering of Strings ...   3 hours ago
```

```
Feed Explorer
org.foo.server
  Client
    getUser(String)
      Tom delivered ... Aug 17, 2010
        Server Work   Yesterday
  Server
org.foo.more
Events outside the Workspace
  Work on ...   Yesterday
  Alex delivered 'some work'   Yesterday
```

```
Feed Explorer
Tom   2 days ago
Alex
  More problems with ...   4 hours ago
  Test execution for ...   4 hours ago
  Filtering of Strings ...   4 hours ago
  Write a test for ...   4 hours ago
  Write a test for...   4 hours ago
  Alex delivered ...
    org.foo.server
Oli
```

(a) Traditional List-Based Feed Reader (b) Feeds in Source Code Context (c) Feeds in Team Context

Figure 1. Presentation of Feeds.

that users frequently use sorting features to better find relevant information (e.g., [3]). Our study extends this work by investigating how context, such as the source code or the team, can help in this sorting.

A number of researchers have attempted to determine relevancy automatically using recommender-based approaches. Chen *et al.* [1] explore a variety of recommender algorithms, largely based on topic relevance, to find interesting URLs in Twitter feeds. Dabbish *et al.* [2] predict the importance of an e-mail message to a user based on predefined categories. These approaches do not try to understand how users define an item of information as relevant to their current situation.

Other research has looked at the problem of staying aware of relevant information. Palantir [6], for example, provides awareness of changes to the source code in context of a developer's workspace by decorating code in the environment as to whether or not it has been changed by someone else. In contrast, our approach focuses on a feed view of changes and considers the placement of news items in the context of a fragment of source code instead of the entire system. In addition, Palantir only indicates a change, whereas our approach presents the news item, which also provides a rational for the change, such as the work item causing the change. Treude *et al.* [9] studied how software developers use feeds in RTC for accomplishing awareness. They did not consider how developers assess the relevancy of the feeds or the items within, which is our focus in this paper.

INTERVIEW METHODOLOGY

For our study we interviewed a sample of five developers (4 male, 1 female) from a development team within a large corporation. The team was chosen based on their use of an environment that contains project feeds (IBM's RTC) and the access we were granted to their project specific changes and information. The participants had between 4 and 24 years of professional development experience (*M*=10 , *SD*=9) and at least 20 months of experience using RTC (*M*=23, *SD*=7).

We interviewed each participant twice, with four days in between the first and the second interview. In each interview, we presented the participant with a list of 30 items[1] from two default feeds that exist in RTC. One feed, *change set news*, captures changes to the code and the other feed, *work item news*, captures changes to work items, which are similar to bugs, issues or tasks. For each participant, we randomly selected the items to present from the set of all items that had occurred for the participant over the previous three days.We ensured that the presented items represented both feeds and that the participant was not the creator of the item. In each session we asked each participant to review the items and to tell us whether or not the item was relevant, how relevant the item was and why the item was relevant (or not). Participants had the option to read and explore the item and related information as much as desired. We deliberately did not define relevance for the participants to allow investigation of differing ideas of relevance between the participants.

In the first interview session, the news items were presented in a list, based on the interface of traditional feed readers (Figure 1(a)). In the second interview session, we asked the participants to review an additional 30 news items. We first showed these additional items in the context of a fragment of the related source code (Figure 1(b)). If the item was not related to the source code, it was shown in a flat list below the code. We asked the participant whether the context changed his determination of the item's relevance and whether or not the context was helpful. Next, we showed the items in terms of a team context (Figure 1(c)), in which a news item was shown underneath the team member who caused the creation of the news item. We asked the participant the same questions for the team context as for the source code context. We chose to investigate source code and team contexts because developers had earlier described this kind of information as critical to many questions they need to answer [4]. In this paper, we examine whether using this information as context can help a developer determine the relevancy of new information.

Each interview session took between 20 and 60 minutes. All interviews were recorded and the interviewer (first author of this paper) took handwritten notes. Due to the exploratory nature of our study, we parsed our data using an open coding technique to develop and identify categories of data [8].

FEEDS

A news item in the study referred to a change in a work item or a change set. In both cases, a summary of what changed was displayed, e.g. "Oli delivered 1034: Work on ...", for a change set (Figure 1(a)). Hovering over the element in the view presented the details of the change in a popup.

[1]In one case, only 21 news items were available.

During the study period, a mean of 50 (*SD*=31) items per day were available to the participant in their feed reader, created by a mean of 5 (*SD*=2) authors. Most, 46 of 50 (*SD*=32), items were about work items; the remaining 4 of 50 (*SD*=3) were about change sets.

In total, our study gathered feedback on 291 news items. Overall participants perceived 79 of the 291 (27%) news items as relevant, 24 (8%) as somewhat relevant and 188 (65%) as not relevant. Participants perceived more of the change set news items as relevant compared to work item news; participants perceived 54 of the 81 (67%) change set news items as relevant or somewhat relevant, whereas only 49 of the 210 (23%) work item news were considered relevant or somewhat relevant. Since all participants worked on the same team, many of the news items overlapped.

In the second set of interviews, we placed the items into the source code context (Figure 1(b)). Of the 150 news items considered in these interviews, 37 (25%) were related to the source code on which the team was working. All of the news items could be related to team members and could thus be shown in the context of the team member that authored the item (Figure 1(c)). More than the five developers in the study were referred to by news items so the team context involved eleven team members.

Although all participants knew of the feed reader in RTC, only one participant described using the view on a regular basis. Others just used it occasionally or very rarely. Instead, they often used other, more specialized views available within RTC to monitor changes. In particular, all participants stated that they regularly monitor changes to the source code, the information summarized in the change set news, using a view called the Pending Changes View.

USER-ORIENTED VIEW OF RELEVANCE

Each participant defined relevancy of a news item differently: a news item is relevant when it impacts the code that I am familiar with (P1), when the information is useful to me now or in the foreseeable future (P2), when it's in an area in which I am an expert of, interested in and almost are lacking expertise (P3), when it helps me to do my job at some point (P4) and if it is in an area that I see myself as a keeper or knowledgable in or critical to the project (P5).

Classification of the detailed reasons given by participants when determining relevancy suggests nine broad categories (Figure 2). Some of the categorized reasons were used mainly for identifying an item as relevant. For example, in 16 cases, participants described that the item being about a change set is why the item was relevant. Other categories are about why an item was irrelevant, such as an item being not from someone on the participant's team. Finally, some of the categorizes were used to identify items as relevant as well as irrelevant.

As described below, participants did not apply consistent criteria for determining relevancy. Although the perception of relevance is individual and situation-based, we have identified four dimensions along which participants determined the relevancy of an item: the content of the item, the target of the content, the relation with the creator and the previous interaction with the topic of the news item.

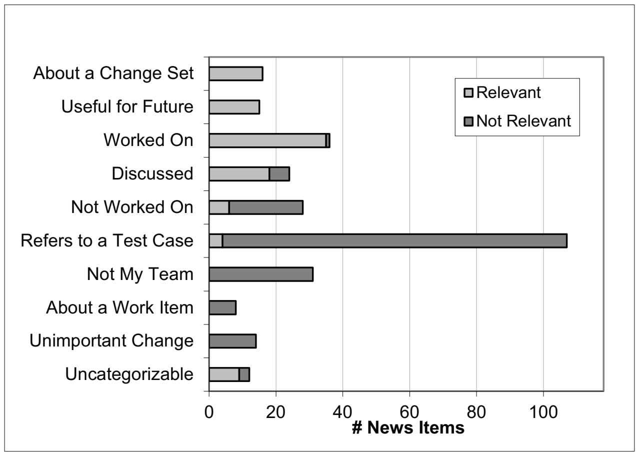


Figure 2. Categorized Reasons Developers Mentioned for Identifying a News Item as Relevant or Irrelevant.

Content

All participants generally rated change set news as more relevant than work item news. One participant even went as far as saying that all change set news for the project are relevant but that work item news are not relevant as too many items appear and much of the information was available for the team on a physical white board (P4). This content-based approach is well-supported by today's tools that support feed subscription based on content. However, as only 67% of all change set news and 23% of work item news were considered as relevant, this statement is not well supported by the data from all participants. Furthermore, this same participant (P4) admitted that because he generally ignores work item news, he also misses questions asked of him through work items, suggesting that a content-based approach has significant drawbacks.

Other participants considered the type of the content, as opposed to the type of the feed, to determine relevance. For example, most participants stated that work item news concerning test cases were not relevant. Other factors did override these general trends. For instance, the participants described that the state of the project affects the relevancy of work item news referring to test cases. In some project states, such as when the system fails to run, work item news related to testing is more likely to be deemed relevant.

This trend of using the type of an item was also seen in other work item news deemed irrelevant, such as items on the addition of a work item to a feature (P1, P2) or on the change of a subscriber to a work item (P5).

Target of Content

The target of the content of a news item, for instance, the code to which a news item referred, was often mentioned as a reason for an item being relevant or not. One participant stated, "This is not relevant since I have not worked on that [part of the code] for a while" (P1). Another stated, "[this is relevant as] the area is related to what I worked on" (P5). As with the content, the relevance based on the target of the content is also very dependent on the situation and the individual. For instance, (P2) stated that even though an item is not in an area he worked on, it could be useful for the future and thus perceived as relevant. Tools could be built to help automate this factor of relevance determination; tracking data about which code a developer touches and changes could be used to filter information streams to only show items on which a developer does (or does not) work.

Relation with Creator

In most cases, an item created by a developer not on the direct team of the participant was perceived as less relevant (P1, P2, P4, P5). However, when the participant worked with a developer from a different team on a regular basis due to project dependencies, the item was perceived as highly relevant (P3). Some participants stated that a combination of factors may make an item relevant; for instance, if an item was created by a certain teammate and touched an area of related work it was relevant (P1). Tools could be built to support the use of combinations of factors in automating determining relevance of items.

Previous Interaction

Whether or not the news had been discussed in another setting had an impact on the perceived relevance of the corresponding news item when it appeared in the feed. Some stated that work item news were not interesting because the news had already been discussed in the planning sessions (P4). Others stated that seeing a news item related to a previously discussed issue was relevant as it confirms what is known (P3). One participant (P5) had mixed views and identified one item as relevant and another one as irrelevant based on having previously discussed these items. Providing support to automatically determine if news had been seen in another setting appears much more challenging. A better characterization of the impact of this factor would be helpful to ascertain in future studies.

CONTEXT TO REDUCE INFORMATION OVERLOAD

Although participants stated that their knowledge about the part of the code to which the news item related was used in determining relevance, only 37 of 150 (25%) were easily determined to relate to the code on which the participants worked. In 20 of the 37 (54%) cases, participants considered the source code context to be helpful. In 9 of 37 (24%) cases the participants referred to it as being somewhat helpful and in 8 (22%) cases as being not helpful. In particular, when the short form of the summary used to identify the news item was too generic, such as "delivered 2 change sets", the context provided a significant benefit to the participant in explaining the item.

When asked about source code context in general, three out of the five participants considered the placement of news items in context helpful (P1,P2,P5). One of the three stated that he "had problems with the [news items] out of context, but [for] the ones within, [he could determine relevancy] pretty quick" (P1). Another participant preferred the team context, stating that the source code context provided too much information but that the team context allows skimming over people quickly (P3). Three more participants (P1,P2,P5) conceived the team context as helpful, in particular in situations in which you do not have to work with a lot of source code, such as quality control.

CONCLUSION

A better understanding of how users determine relevance of items from feeds is needed to build better tool support to help users deal with the vast amount of information flowing to them in this form. Our study of how developers determine the relevancy of items in project-related feeds resulted in a description of four factors that impact relevancy. Only one of these factors, the type of information stream, is well-supported by today's tooling. Other factors such as the target of the content and the relationship with the creator appear plausible to develop. The fourth factor, previous interaction, requires additional study to better understand its impact in the determination of relevancy. Our study also shows that the placement of information stream items into user-oriented context, even as fragments, can be helpful in determining relevance. Further study is needed to understand how these factors generalize into other settings.

REFERENCES

1. J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. In *Proc. of CHI '10*.
2. L. A. Dabbish, R. E. Kraut, S. Fussell, and S. Kiesler. Understanding email use: predicting action on a message. In *Proc. of CHI '05*.
3. N. Ducheneaut and V. Bellotti. E-mail as habitat: an exploration of embedded personal information management. *interactions*, 2001.
4. T. Fritz and G. C. Murphy. Using information fragments to answer the questions developers ask. In *Proc. of ICSE'10*.
5. T. W. Malone, K. R. Grant, F. A. Turbak, S. A. Brobst, and M. D. Cohen. Intelligent information-sharing systems. *Commun. ACM*, 30(5), 1987.
6. A. Sarma, Z. Noroozi, and A. van der Hoek. Palantír: raising awareness among configuration management workspaces. In *Proc. of ICSE '03*.
7. L. Schamber, M. B. Eisenberg, and M. S. Nilan. A re-examination of relevance: toward a dynamic, situational definition. *Information Processing & Management*, 26, 1990.
8. A. C. Strauss and J. Corbin. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage Publications, Inc, 2nd edition, 1990.
9. C. Treude and M.-A. Storey. Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. In *Proc. of ICSE '10*.
10. G. D. Venolia, L. Dabbish, J. Cadiz, and A. Gupta. Supporting email workflow. Technical report, 2001.