Anticipative Awareness in a Groupware System

Wolfgang Prinz[1,2], Elke Hinrichs[1], Irina Kireyev[2]
(1) Fraunhofer FIT, (2) RWTH Aachen University, Germany

Abstract. This work presents a new approach in the area of awareness in groupware. We have made a first step towards supporting both conventional, past-oriented and anticipative, future-oriented awareness by suggesting a conceptual model for defining expectations of actions in a shared workspace and for an automatic check and notification of fulfillment or non-fulfillment of these expectations. Our approach is people-centric and is based on a real-life mental state: the state of anticipating certain events in a workspace. After having studied previous theoretical and practical research on awareness and having analyzed several current collaborative systems, we developed a conceptual model of expectations in shared workspaces. We then implemented our approach as an add-on package to the shared workspace system BSCW. Having evaluated our system with a group of BSCW users we analyzed their feed-back and came to the conclusion that all test users declared the concept of anticipative awareness useful and the implementation on the whole usable. Based on the comments of the test users we propose several improvements of the implementation.

Keywords: awareness, shared workspaces.

1 Introduction

Current asynchronous collaborative tools support change awareness in an electronic workspace in two ways. Either they store and display a history of events that took place in the past (awareness of past events) or they allow users to register an interest in an event in the future they want to be notified about (awareness of future events). The future event can ideally be defined to encompass any combination of specific artifacts, specific activities and specific members of a workspace. None of the groupware tools we analyzed, however, allows specifying actions to be carried out by workspace members until a date in the future and then automatically verifies and notifies whether they were performed within the deadline or whether they were not performed.

The latter is a frequent requirement in collaborative work as shown by the following typical situations. One team member writes a document and expects the other team members to read it, to review it or to suggest improvements. A teacher asks her students to write an essay and wishes to "remind" late students of the delivery when the deadline approaches. A call for papers is issued and abstracts have been accepted, but the conference committee needs to know who failed to deliver the full paper, rather than who actually did deliver.

We propose to fill the obvious gap by complementing awareness of what happened in the past by awareness of events which will or will not happen in the future — we call this concept anticipative awareness or short: expectations. Expectations allow users the flexibility of defining in detail what action by which participants and on which artifact they are interested to observe and until when.

We decided to implement the concept of anticipative awareness within BSCW [1], a shared workspace system developed by Fraunhofer FIT and OrbiTeam starting in 1995 and in operation ever since, which has a large user community of well over a million users world-wide. We chose BSCW as the base-line system for our implementation for three reasons. Firstly, we wanted to improve the tool we are using in our daily work. Though BSCW features a broad range of awareness functions, it can still be difficult to keep track of the state of work in large shared workspaces with active team members. When deadlines are approaching, it can be quite cumbersome to find out who did not deliver — actually this has to be done manually. Automated notifications of these anticipated (non-) events well ahead of time would be most welcome to many BSCW users. Secondly, the existing awareness features of BSCW made the implementation of anticipative awareness relatively easy. Thirdly, the large number of "real users who are doing

real work" using BSCW offers excellent opportunities for a thorough evaluation and verification of our concept.

After reporting on related work *(final version, not in this abstract)* in the area of awareness, we give a detailed account of our concept of anticipative awareness and how we implemented this concept by integrating it into our shared workspace system BSCW. We then describe how we evaluated our work and "what the users said". This leads us to a critical analysis of what the users liked and what should be improved and how. We conclude with an outlook on our future plans.

2 Concept

Our concept of expectation awareness enables a user to specify his anticipation of future activities on a certain artifact or a group of artifacts in a shared workspace. The users will be informed at a chosen point in time if the expected activities happened or not. Our approach is based on the observation that after performing an activity on an artifact in a workspace, a user usually has a mental picture of what activities on this artifact should be performed next by his collaborators. For example, after uploading a document, the user would probably anticipate that it will be read or modified; after making a poll, he probably waits for others to vote on it. Usually this anticipation also has a certain time limit, within which the activity should take place. We call this mental picture an expectation.

After an expectation is being created, depending on whether its creator decided to notify the involved members, an icon appears in the workspaces of the involved members. In addition a periodic alarm mechanism is set to test the expectation fulfillment at certain time intervals. As long as the expectation is not fulfilled, the periodic fulfillment tests are performed until the expectation end time. At the end time the last fulfillment test is performed and when the expectation is not fulfilled, a non-fulfillment message is sent. If the expectation gets fulfilled, the periodic test mechanism stops and waits for the expectation end time to send the fulfillment notification.

In practice, we model an expectation as an object with six basic attributes: creator, artifact, activity, participants, start date, end date. Many additional attributes can be defined to extend the usability and flexibility of an expectation. For example, a selection of a preferred notification mechanism or an option to inform the involved collaborators of an expectation is thinkable. The values of these attributes will be specified by the user when a new expectation is created.

An expectation changes its state after the deadline. The three basic possible states are initial (before the deadline), fulfilled or non-fulfilled (after the deadline). An expectation itself, as any other object in the system, has operations to manipulate it. We consider as essential operations create, edit, delete, check fulfillment and view fulfillment status.

3 Related Work

Tam and Greenberg [15] define workspace change awareness as the ability of individuals to track the asynchronous changes made to a collaborative document or graphical workspace by other participants over time. In their theoretical framework they list the required information elements for change awareness. The set of questions that the workspace should answer is the following:

- Where have changes been made? — the location of each change
- Who has made the changes? — the author of each change
- What changes were made? — the content of each change
- How were things changed? — the actions that lead of each change
- When did the changes take place? — the time of each change
- Why were the changes made? — the reason for each change

We based the expectation form and notification message in our application on this list of required elements.

Bernheim Brush et al. [13] divide existing asynchronous awareness mechanisms in one or more of the following categories: informational, subscription based and peripheral.

With informational awareness mechanisms, details of the document activity are shown or can be queried from the workspace. The majority of systems concentrate on this type of awareness. ReadWear and EditWear technique [9], Version graph in COOP/Orm [11] and ActivityViewer in Palantir [14] are just some examples of existing practices to provide information on changes in a workspace itself. BSCW already supports this type of awareness with event icons, event history and AwarenessMaps [1], [8].

In subscription-based awareness mechanisms, a user can register an interest to receive notifications on certain events in a workspace. Systems like CVS-watch [2], Coven [5], or Bugzilla [16] originate in the software engineering community, while the CSCW community contributed Khronika [10], Elvin [6] and Nessie [13] to implement user-controllable notification mechanisms.

In peripheral-based awareness mechanisms, often widgets are applied to provide awareness at a glance outside the workspace, for example like a sidebar in [3] or [4]. Peripheral awareness is mostly used in synchronous collaboration. But since the widget will also show changes that have occurred when the user was offline, as soon as the user connects again, it can be valuable also for asynchronous awareness.

The existing collaborative tools use informational, subscription-based or peripheral awareness mechanisms to represent events or activities that took place in the past or that take place in the present. The informational awareness mechanisms show all the events that occur in a workspace. In frequently used workspaces the number of the events that occur is large and hence it is overwhelming and even nearly impossible for the user to view all the events. Subscription based awareness mechanisms filter the events of interest for a user with respect to his awareness profile, but the number of notifications they generate can be also quite large, overwhelming and annoying. Therefore several approaches exist to combine events in an overview presentation [12].

Nevertheless, none of the tools we have analyzed allows a specification of events or actions that should take place in a work-space in the future to automatically verify and notify a user whether they were performed or not.

4 Concept

4.1 Anticipative Awareness

Our concept of expectation awareness enables a user to specify his anticipation of future activities on a certain artifact or a group of artifacts in a shared workspace. The users will be informed at a chosen point in time if the expected activities happened or not. Our approach is based on the observation that after performing an activity on an artifact in a workspace, a user usually has a mental picture of what activities on this artifact should be performed next by his collaborators. For example, after uploading a document, the user would probably anticipate that it will be read or modified; after making a poll, he probably waits for others to vote on it. Usually this anticipation also has a certain time limit, within which the activity should take place. We call this mental picture an expectation. Figure 1 illustrates the process of an expectation.

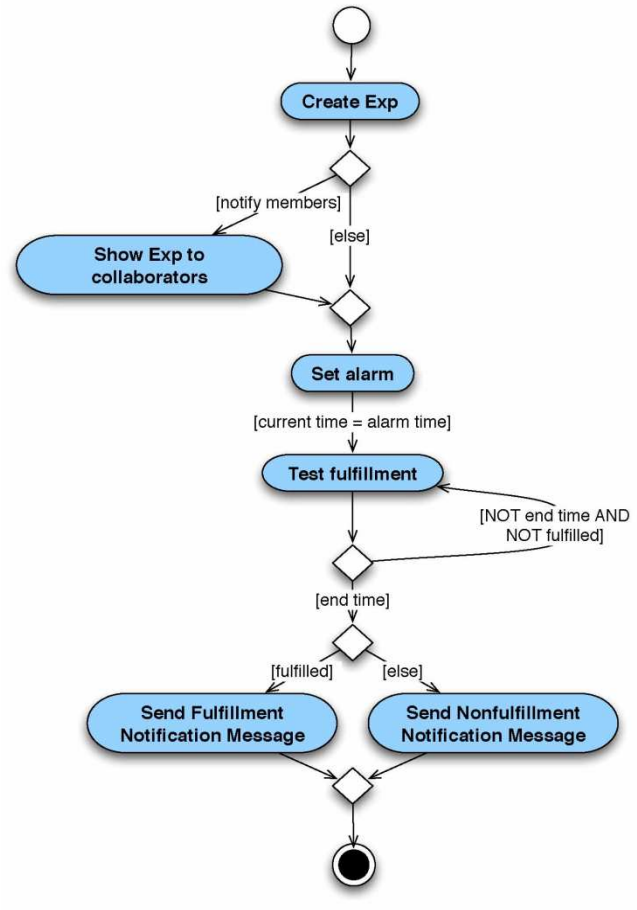

Figure 1: Expectation process diagram

After an expectation is being created, depending on whether its creator decided to notify the involved members, an icon appears in the workspaces of the involved members. In addition a periodic alarm mechanism is set to test the expectation fulfillment at certain time intervals. As long as the expectation is not fulfilled, the periodic fulfillment tests are performed until the expectation end time. At the end time the last fulfillment test is performed and when the expectation is not fulfilled, a non-fulfillment message is sent. If the expectation gets fulfilled, the periodic test mechanism stops and waits for the expectation end time to send the fulfillment notification.

In practice, we model an expectation as an object with six basic attributes: creator, artifact, activity, participants, start date, end date. Many additional attributes can be defined to extend the usability and flexibility of an expectation. For example, a selection of a preferred notification mechanism, or an option to inform the involved collaborators of an expectation is thinkable. The values of these attributes will be specified by the user when a new expectation is created.

An expectation changes its state after the deadline. The three basic possible states are initial (before the deadline), fulfilled or non-fulfilled (after the deadline). An expectation itself, as any other object in the system, has operations to manipulate it. We consider as essential operations create, edit, delete, check fulfillment and view fulfillment status.

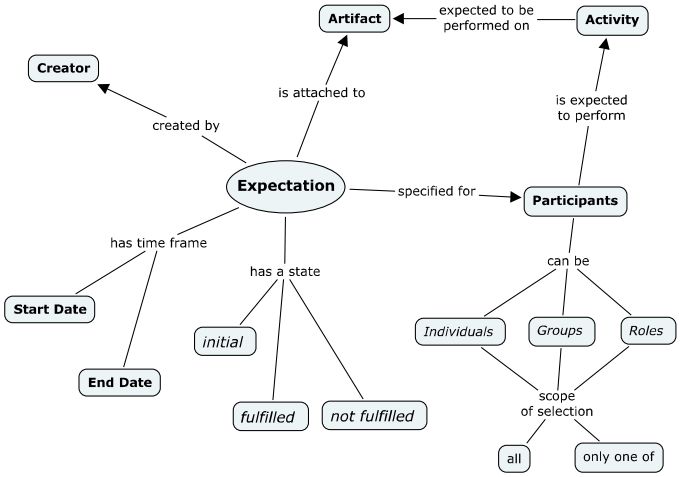

Figure 2: concept map of the expectation concept

In the following we describe the major attributes of an ex-pectation in more detail: artifact, activity and participants.

4.2 Artifact

The artifact attribute refers to a shared object in a workspace, upon which the chosen activity should operate. We distinguish between a single artifact and a container of artifacts. A single artifact is an object that can be uploaded to or created in a workspace, such as a document or a note. A container of artifacts is a collection of artifacts, such as a folder, a blog or a discussion.

For a set of artifacts, users should be able to decide, whether the expectation is fulfilled when the expected activity takes place in all artifacts or in at least one of the artifacts. This selection will often depend on the expected activity itself. It is obvious to expect, for example, that the read operation will take place in all documents in a folder while the modify operation only on at least one of them.

An exceptional activity for this matter is the create activity. The create operation can be expected only in a container, such as a folder. Another consideration of interest is the case of a hierarchical structure of objects. Then it must be decided, whether the activity should take place only in the uppermost level or in the whole hierarchy.

4.3 Activity

Activity is an action or operation expected to be executed on the chosen artifact or on a chosen container of artifacts. The list of possible actions is the operational semantics of a chosen artifact with respect to the chosen participants. In other words, these are the operations available on the chosen artefacts for the chosen participants in accordance with the access rights of the latter.

In case of an artefact containing other artefacts (e.g., a folder) there are two ways to define a possible list of actions: either an action common to all artifacts or an action that is executable only on one of the artifacts.

4.4 Participants

Participants are other members with whom the creator shares the selected object. Similar to an object, a single collaborator can be either a single member or a ’complex’ collaborator, such as a group or a role with a number of members. The list of admissible operations for a certain artifact depends on the collaborator’s rights to perform operations on this artifact.

For members of a group or a role it should be decided, whether the expectation should be fulfilled by a role or by a group. Further one must consider if the expected activity should be performed by all members or by at least one member inside of a role or a group. Again, this definition might depend also on the activity itself, since it is quite natural to expect, for example, that the read operation will be performed by all members of a role or a group while the modify operation by at least one of them.

Finally it should be decided by the user what is the fulfillment condition for the selected set of collaborators (counting a role or a group as one collaborator), i.e. whether all collaborators must fulfill or at least one of them. Figure 3 provides a state diagram of an expectation evaluation.

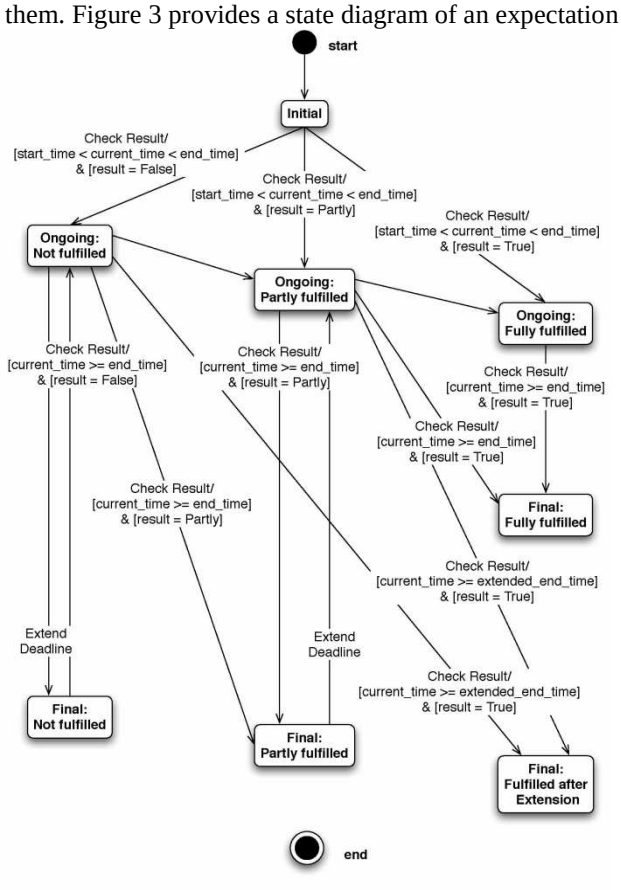

Figure 3: State diagram of an expectation evaluation process

A user can decide whether he wants to make an expectation being visible to the other participants or not. Making an expectation visible, the user implicitly announces to other members what kind of activities he expects them to perform in a workspace. This is intended to establish a better understanding of the expected activities among the members of a project. However, this use of an expectation makes it very similar to another formal method of project management: task assignment.

Therefore, we would like to emphasize the differences between the concept of an expectation and a concept of a task. These are: the intention, the level of necessity, the agreement of the participant assigned.

Intention: A task is a work unit with a sequence of instructions. The main intention of a task is specifying the work needed to be done and assigning it to a certain person or number of persons. The main purpose of an expectation is providing better awareness on anticipated activities in a workspace. Creating an expectation supports the automatic summarization of events that the user is looking forward to occur in a workspace until a certain deadline. Currently this information can be obtained only manually by looking at the history of events.

Level of necessity: Assigning a task is psychologically more demanding than creating an expectation. The one who assigns a task should have the sufficient authority, while anyone in a shared project can have expectations from his colleagues. Tasks have to be performed as a duty, while an expectation is just a lightweight representation for looking forward to a certain action.

Agreement of the assigned participant: The mechanism of a task assignment usually entails the agreement of the participant whom the task is being assigned to. During an expectation creation the collaborators not only do not have to agree to the expectation, but in most cases they do not have to know aboutt an expectation. An expectation is just an expression of what its creator believes should happen in a workspace and an easy means for being informed whether it happens or not.

5 Implementation into a shared workspace system

This chapter describes our implementation of the anticipative awareness concept. First we give an overview of the BSCW system – the baseline system for our implementation. Then we provide a detailed description of the user interaction and user interface of expectations feature.

5.1 Baseline System BSCW

BSCW is a webbased groupware system around the shared workspace metaphor. Shared workspaces are established by groups of people to organize and coordinate their work. A BSCW server (a standard web server extended by the BSCW functionality through the Common Gateway Interface) manages a number of shared workspaces – repositories for shared information, accessible to the members of a group via any normal web browser using the user name/password authentication scheme. In general, a BSCW server manages workspaces for different groups, and users may be members of several workspaces.

A workspace may contain different kinds of information, represented as information objects arranged in a hierarchical order. The objects may be of various types such as folders, URL links to Web pages, documents, like graphics, spreadsheets, text documents, discussion forums, opinion polls, tasks or user specific address books and calendars. The system allows numerous operations – usually depending on the object type – that can be applied to objects, e.g., objects may be renamed, deleted and undeleted, for URL objects the URL reference may be modified, documents may be put under version control, or users may add a comment to a discussion forum. Operations that have been carried out result in events, which are reported to the users by means of event icons so that they are aware of each other's activities in a workspace.

From a system developer's point of view, our main consideration for choosing the BSCW shared workspace system as a baseline system was its modularity and extendibility – new features can be added relatively easily to BSCW by using the packages mechanism. The packages mechanism in BSCW allows switching a certain feature on and off during runtime.

Consequently, the expectation feature was implemented as a BSCW package consisting of various class definition modules, utility routines and user interface modules, XHTML templates and cascading style sheets. The kernel software of BSCW was left unchanged.

Figure 4: Expectation indication in a BSCW workspace

Figure 4 presents the user interface of BSCW. Several object have expectations attached which is indicated by the colored “E” symbols.

5.2 Examining the state of an expectation

The creator of an expectation can obtain information about the state of her expectation in several ways: via change of colour in the expectation icon (see Fig. 4), via the expectation state page, and via email notification at the end of the expectation period

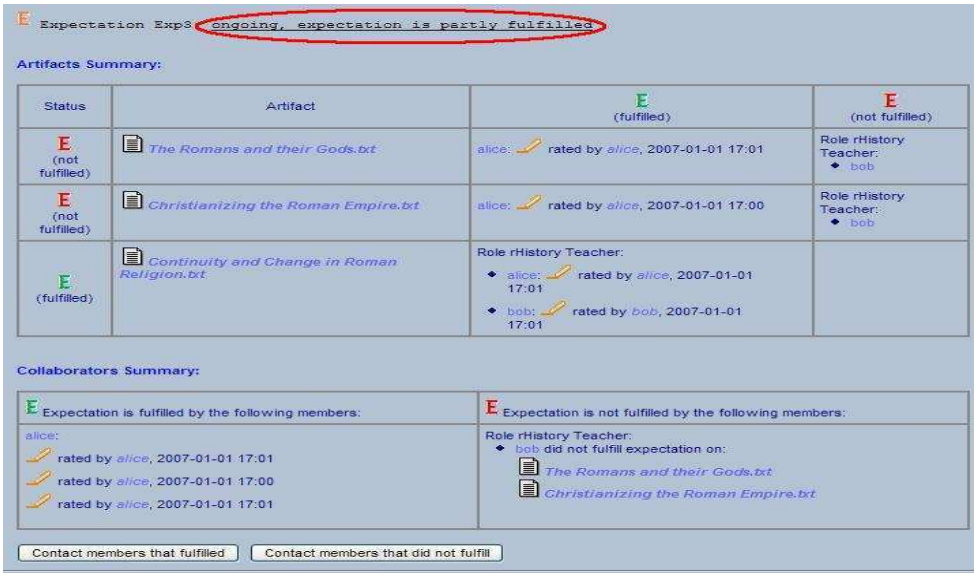

Figure 5: Expectation State Page

Figure 5 demonstrates the state check of an expectation of a rate event. The expectation is on a folder with three documents and two users: Alice and Bob. Alice and Bob hold the role History Teacher. The user expects that all users must rate all documents. The state page shows that the users Alice fulfilled the expectation and the user Bob did not fulfil on two out of three documents. Alice appears in the column of collaborators that fulfilled for each one of the documents and in the summary. Bob rated only one of the documents and thus he appears in the column of the collaborators that fulfilled for this document, but he appears also in the column of collaborators that did not fulfil for other two documents and in the summary. The fulfilment state of the role “History Teacher” is not fulfilled because of its non-fulfilling member Bob. The overall expectation state is is not final, because the end time is not over yet.

6 Evaluation

In this section we present the evaluation of the new expectations features in BSCW. The evaluation consisted of tests carried out by ten representative participants from our target group: experienced BSCW users who are responsible for the management of large international projects using BSCW workspaces with many (>30) members. Each test was held by one person at a time and it took 30-45 minutes.

The evaluation began with a short briefing and a hands-on experiment to get used to the system functionality and use. Afterwards the test persons were asked to perform several tasks on their own. The tasks became more and more elaborated and they covered the whole set of expectations functionality, including the advanced features for creating an expectation on roles and groups. Each participant created or fulfilled five expectations during the test. Out of these, four

were to be created by the test participant in accordance to a given task-scenario. One expectation was prepared in advance by the facilitator. It had to be understood and fulfilled by the participant.

After an expectation was created by the test participant, it was fully fulfilled, partly fulfilled, or not fulfilled at all by the test facilitator. Then the test participant was asked to look at the result and to describe it to the facilitator. Each expectation had to be created on a different artifact in a workspace.

The users were asked to provide comments (thinking aloud) or to ask questions while executing the tasks. These questions and comments, together with usability issues identified during the user tests were collected into a log book.

After each test we conducted a semi-structured interview including questions about the overall impression of the system, whether the expectation concept was clear and easy to learn, estimation of the usefulness of such a feature, possible use cases, and improvements to the user-interface.

Next, we summarize the difficulties encountered during the tests, the suggestions made by the users and the overall opinion and impressions that the users expressed in the semi-structured interviews.

Difficulties were mainly identified with the expectation creation form and the navigation to the expectation actions. No difficulties were encountered in reading the expectation result or in changing an existing expectation. All the test participants could successfully solve the task.

The struggle with the expectation creation form was due to understanding the meaning of the advanced participant selection for an expectation. The options to select all members or at least one member for a role or a group, often required further explanation. The majority of the participants preferred to select members, one by one, instead of selecting a role or a predefined group. However, in another task, where a specific group had to be selected for an expectation, all participants configured the expectation correctly.

Difficulties with the navigation to expectation actions were mainly due to multiple clicks that were needed to get to the expectation result page. Users expected to see the fulfillment state directly after clicking on an expectation icon

In addition to these usability issues, a number of suggestions were collected that relate to the icon design (improvements for color-blind users), the selection of a deadline (a date is sufficient, a precise time is not needed), and the format of the notification email (more details were required).

All test participants found the expectation concept easy to understand and the expectation result simple to read. All the test participants considered the expectation feature as a very useful extension to the awareness features of the shared workspace system. Some users argued that they would use the feature mostly in large workspaces for awareness, especially to establish anticipative awareness. The majority said that they prefer expectations over tasks since they are less demanding than a task assignment.

Based on this positive and encouraging feedback we have improved the expectation functionality. It is currently being provided at a workspace server for three large European research projects with more than 200 users. After the announcement of the feature to the user group a number of expectations have already been created by the users. Until today we have received very positive feedback about the easiness to monitor the status of an activity and in particular to supervise outstanding activities

7 Lessons learnt and future work

The user tests have shown that the expectation concept is easy to learn, the implementation is responsive and stable and that the expectation result is easy to read. Nevertheless further consideration is needed to improve the expectation icon, the expectation creation form and the navigation to expectation operations. In the following we describe the realized improvements.

The expectation icon now leads directly to the expectation status page instead of the configuration page. This is due to the observation that users are more frequently monitoring the expectation status than updating it. Furthermore it is planned to include a tooltip for the expectation icon that displays further expectation details such as name, event type, creation date, end date.

The selection of single users, roles and groups as participants of an expectation has been simplified by moving the role and group selection into an advanced options window, while the

selection of single members remains on the main page of expectation creation. This enables the creation of standard expectations in one step. The selection of “at least one” or “for all” conditions, on members inside roles and groups and on members including roles and groups, requires further consideration. Since it is also a cognitive challenge to understand these specific set operations, the creation of an easy to use user-interface for this task remains a challenge. To ease the navigation, additional expectation operations have been added to the context menu of an artifact in a shared workspace.

Currently the expectation fulfillment evaluation is triggered by a periodic alarm. Thus it may happen that an expectation has already been fulfilled, although the state has not yet been updated, due to the triggering interval. To increase the responsiveness of the system, it should be triggered by each operation that takes place in a workspace. However this requires changes to the kernel of BSCW which was avoided for the first version of the application.

With respect to the expectation concept, the following extensions are planned. In our concept an expectation can be set either on a single artifact or on all artifacts in a container. Furthermore it is desirable to enable the selection of several artifacts (not necessarily from the same container) for a single expectation. However, we expect that this may cause user-interface challenges. Thus the usefulness of this extension remains to be evaluated.

Vice versa, our expectation concept is currently limited to the creation of an expectation on an entire artifact. But, often expectations refer to specific artifacts elements or parts. Developing a more sophisticated concept of expectations enabling users to describe, how they expect the future contents being cooperatively developed will be the next step.

Finally, we suggest the evaluation of the usefulness of expectations in other domains such as software development, architectural design, web content and knowledge management, e-learning and collaborative problem solving. In this context it will be necessary to extend the presented concept from a single CSCW application (i.e. shared workspace system) to several applications in a cooperative working environment including workflows, instant messaging and application sharing.

8 Conclusion

This work presents a new approach in the area of awareness in groupware. We have made a first step towards supporting both conventional, past-oriented and anticipative, future-oriented awareness by suggesting a conceptual model for defining expectations of actions in a shared workspace and for an automatic check and notification of fulfillment or non-fulfillment of these expectations. Our approach is people-centric and is based on a real-life mental state: the state of anticipating certain events in a workspace. After having studied previous theoretical and practical research on awareness and having analyzed several current collaborative systems, we developed a conceptual model of expectations in shared workspaces. We then implemented our approach as an add-on package to the shared workspace system BSCW. Having evaluated our system with a group of BSCW users we analyzed their feed-back and came to the conclusion that all test users declared the concept of anticipative awareness useful and the implementation on the whole usable. Based on the comments of the test users we propose several improvements of the implementation.

References

1. Appelt, W. WWW Based Collaboration with the BSCW System. SOFSEM ’99 conference (Milovy, Czech Republic). Springer Lecture Notes in Computer Science 1725.
2. Berliner, B. CVS II: Parallelizing software development. In Proceedings of the USENIX Winter 1990 Technical Conference, 1990.
3. Bernheim-Brush, A. J., Bargeron, D., et al. Notification for Shared Annotation of Digital Documents. In Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves. ACM Press, 2002.
4. Cadiz, J., Venolia, G., et al. Designing and Deploying an Information Awareness Interface. In Proceedings of the Conference on Computer Supported Cooperative Work, (CSCW 2002), (New Orleans, USA), ACM Press, 2002.
5. Chu-Carroll, M. C., Sprenkle, S. Coven: Brewing better collaboration through software configuration management. ACM SIGSOFT Software Engineering Notes, 25: 88-97, 2000.

6. Fitzpatrick, G., Mansfield, T., et al. Augmenting the Workaday World with Elvin. In Proceedings of the Sixth Conference on Computer Supported Cooperative Work. (ECSCW '99). Copenhagen, Kluwer Academic Publishers, 1999.
7. Fraunhofer FIT and OrbiTeam. BSCW – Basic Support for Cooperative Work. http://bscw.fit.fraunhofer.de/, 2007.
8. Gräther, W., Prinz, W. Visualizing Activity in Shared Information Spaces. In Human-computer interaction. J. Julie: 1096-1100, 2003.
9. Hill, W. C., Hollan, J. D., et al. Edit Wear and Read Wear. In Proceedings of the SIGCHI conference on Human factors in computing systems, 1992.
10. Lövstrand, L. Being selectively aware with the Khronika System. In Proceedings of the 2nd European Conference on Computer Supported Cooperative Work, Amsterdam, Kluwer Academic Publishers, 1991.
11. Magnusson, B., Asklund, U. Fine Grained Version Control of Configurations. In COOP/Orm. Lecture Notes In Computer Science; Proceedings of theSCM-6 Workshop on System Configuration Management, Springer, 1996.
12. Pankoke-Babatz, U., Prinz, W., et al. Stories about Asynchronous Awareness. Cooperative Systems Design - Scenario-Based Design of Collaborative Systems. F. Darses, R. Dieng, C. Simone and M. Zacklad, IOS Press: 23-38, 2004
13. Prinz, W. NESSIE: An Awareness Environment for Cooperative Settings. In Proceedings of the Sixth Conference on Computer Supported Cooperative Work, (ECSCW'99), Copenhagen, Kluwer Academic Publishers, 1999.
14. Ripley, R. M., Sarma, A., et al. Using Visualizations to Analyze Workspace Activity and Discern Software Project Evolution, in UCI ISR Technical Report. http://www.isr.uci.edu/tech-reports.html, 2006
15. Tam, J., Greenberg, S. A framework for asynchronous change awareness in collaborative documents and workspaces. International Journal of Human-Computer Studies (64): 583-598, 2006.
16. The Mozilla Organization, Bugzilla Bug Tracking System, http://www.bugzilla.org/, 2007.