Continuous Awareness: A Visual Mobile Approach

Conc Chen
Department of Computer Science
University of Texas at Dallas
800 W. Campbell Road
Richardson, Texas, USA
congchen@utdallas.edu

Kang Zhang
∗
School of Software Engineering
Tianjin University
92 Weijin Road
Tianjin, China
kzhang@tju.edu.cn

ABSTRACT
With the proliferation of globally distributed projects, there has been an urgent need for continuous coordination and continuous awareness support. Traditional desktop-based approaches are insufficient for the requirements of continuous awareness. Team Radar Mobile takes a visual mobile approach to continuous awareness by visualizing awareness information on mobile platforms. The concept of continuous awareness and its implementation on multiple platforms are discussed. An experiment has evaluated the visual mobile approach to continuous awareness, and validated the design principles of mobile visualization.

Categories and Subject Descriptors
D.2.2 [Software Engineering]: Design Tools and Techniques; H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces—collaborative computing, computer-supported cooperative work

General Terms
Theory, Design

Keywords
Awareness, visualization, mobile, collaboration

1. INTRODUCTION
Software development is a typical group activity. Such teamwork requires intense collaboration among team members as well as other outside stakeholders. Awareness, “an understanding of the activities of others that provides a context for one’s activities [11]”, is an important contributor to effective team collaboration, as it “aids coordination of tasks and resources, and assists transitions between individual and shared activities [13]”. The last decade has witnessed a proliferation of awareness-based informal approaches to collaboration, in addition to process-based formal approaches [28].

∗Kang Zhang is also affiliated with Department of Computer Science, University of Texas at Dallas, USA

Awareness in collocated teams is usually maintained through direct interactions, such as monitoring each other’s activities, informal conversations, and expert assistance [10]. To foster innovation and competition, more and more software teams are becoming distributed. Team distribution incurs many challenges to collaboration, such as physical, social, and cultural barriers, which consequently obstruct the channel of awareness information [25]. The loss of awareness not only harms team effectiveness and mutual trust [5], but also affects contributors’ willingness and enthusiasm of work [17]. In such a setting, people have to take various alternative approaches to maintain awareness. A number of researchers have argued that the key to promote collaboration in distributed teams is increasing the level of awareness and providing continuous information of ongoing changes [23].

Continuous coordination [28] is a new collaboration paradigm that combines the strengths of traditional formal, process-oriented approaches with those more informal, awareness-based approaches. Continuous coordination promotes the dissemination of continuous awareness information and the integration of heterogeneous tools. We argue that continuous awareness, a continuous acquisition of awareness information across time and space, is essential for continuous coordination in distributed software teams.

Continuous awareness inherently requires multi-platform cooperation. Inspired by the concept of mobile CSCW [27], we call awareness support on desktop platforms desktop awareness, and its support on mobile platforms mobile awareness. In globally distributed teams, awareness need often changes with people’s role, time, and place. Users’ mobility often disables the effectiveness of desktop awareness tools [2], and a mobile awareness tool would complement the insufficiency of continuous awareness support on desktop platforms. Lessons learned from the Computer Supported Cooperative Work (CSCW) community also suggest that an awareness solution can and should “combine the advantages of desktop-based awareness applications – constant, non-disruptive awareness – with the freedom provided with mobile devices” [19]. To our knowledge, however, there have been no mobile awareness tools available in the software engineering community.

Based on our previous work on desktop awareness, Team Radar [5], we developed Team Radar Mobile, a mobile awareness tool that visualizes awareness information on mobile platforms. From now on, we call Team Radar on desktop Team Radar Desktop and the entire system on both desktop and mobile platforms Team Radar. Our work represents the
first attempt at implementing continuous awareness through the cooperation of both desktop and mobile platforms. This paper introduces the concept of continuous awareness, its implementation, and a preliminary evaluation on the visual mobile approach.

The rest of the paper is organized as the following. We first lay out the background of our work, followed by an introduction of the concept of continuous awareness. Then the design of the entire Team Radar system and the design of Team Radar Mobile are discussed. Finally, we report a preliminary evaluation of the visual mobile approach and conclude the paper with future work.

2. RELATED WORK

Our work applies visualization techniques on mobile devices to support continuous awareness. Related research areas include awareness for software development, mobile CSCW, and mobile visualization.

Learned from the CSCW community, awareness is now considered an important approach to many collaboration problems in software development [25]. Some representative work of such research includes workspace awareness tools (e.g., Palantir [29] and Gasper [20]) for alleviating merge conflict problems in shared workspace and social awareness tools (e.g., Ariadne [10] and [5]) for increasing the sense of “teamness” [5].

The CSCW literature has already demonstrated the value of mobile platform for awareness support. Presence awareness tools, such as IPAD (portable Inter-Personal Awareness Device) [19] and FriendZone [3], provide availability and proximity information about other mobile users. Context awareness tools (e.g., ContextContacts [26]) give users cues on the context of other users, which helps determine the best timing and channel of communication [24]. The ConNexus and Awarenex projects [30] present design principles and lessons learned in extending awareness services from desktop to mobile platforms. Papadopoulos [27] studied collaboration in mobile platforms and generalized key requirements for awareness in mobile CSCW.

As the computational power of mobile devices grows, research on mobile visualization has become more popular than before [15]. Chittaro [9] suggests 6 steps in designing mobile visualizations. Burigat et al. [4] studied and compared the Overview+Detail visualization on desktop and mobile platforms.

Our previous work on desktop awareness, Team Radar Desktop [5], studies how visualization can help express fine-grained awareness information more effectively. Inspired by continuous coordination [28], we propose the concept of continuous awareness, and have developed Team Radar Mobile, a mobile awareness tool that works with Team Radar Desktop as a complete continuous awareness system.

3. CONTINUOUS AWARENESS

Coordinating large-scale collaboration is a difficult task. In software development practices, there exist two major coordination paradigms: formal process-based approaches and informal awareness-based approaches [28]. Formal approaches rely on predefined process models and tools to ensure synchronization of collaborators’ work. A typical example is a software configuration management (SCM) system. Developers first check out the artifacts from the SCM repository to get the up-to-date version of the work. This mechanism isolates and protects the local workspace from other collaborators’ interference. When the developer finishes the modification, she checks in the new version so that it can be synchronized with others’ work. Formal approaches are well-disciplined and scalable, but suffer from reconciliation problems: isolated modifications may conflict when merged because developers have no idea of others’ activities and might take conflicting actions. Informal approaches attempt to mediate the isolation of local workspaces by helping users communicate frequently, keep aware of others’ activities, and conduct self-coordination. A major problem of informal approaches is that they usually do not scale well because of the users’ cognitive limitations. Team Radar attempts to ease the cognition of complex awareness information by visual metaphors and visualization techniques.

Traditional practices treat these two paradigms oppositely. Users usually choose either formal approaches or informal approaches, and cannot benefit from both of them. Continuous coordination is a new paradigm that breaks this dichotomy and combines elements of both formal and informal approaches. It retains the isolation and synchronization mechanisms of the formal approaches, but also provides informal awareness information between synchronizations.

Continuous coordination emphasizes several design principles for implementing coordination systems, including multiple perspectives, non-obtrusive integration, combination of social and technical factors, and integrated formal and informal coordination approaches. However, one important principle is missing: continuous awareness, a continuous acquisition of awareness information across time and space. Existing principles of continuous coordination requires seamless integration of multiple perspectives, approaches, and tools. To meet that requirement, awareness information must be acquired and processed continuously to avoid information gaps. We argue that continuous awareness is essential for continuous coordination in distributed software teams. Unlike traditional paradigms of awareness research, which study single awareness type individually, continuous awareness requires the processing of integrated awareness information. In global collaboration and large-scale organizations, awareness need often changes with people’s role, time, and place. A continuous awareness system should provide integrated awareness information to meet users’ changing needs contextually.

Continuous awareness inherently requires its implementation to support multiple platforms. Current awareness systems are restricted to desktop platforms, which are insufficient for distributed collaboration. For example, a globally distributed team often crosses multiple time zones. The continuity of awareness support will be compromised if some team members are not in front of the desktop computer, or even not at work. It has become a common practice for many project management tools to offer a mobile version
Table 1: Awareness Information Supported by Team Radar

<table>
<thead>
<tr>
<th>Information need</th>
<th>Benefit</th>
<th>Benefited party</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project status</td>
<td>Monitoring the progress of the project</td>
<td>Manager</td>
</tr>
<tr>
<td>Activeness of developers</td>
<td>Understanding the workload of developers</td>
<td>Manager</td>
</tr>
<tr>
<td>Patterns of collaboration</td>
<td>Understanding team organization and group dynamics</td>
<td>Manager</td>
</tr>
<tr>
<td>Other developers’ activities</td>
<td>Understanding work dependency</td>
<td>Developer</td>
</tr>
<tr>
<td></td>
<td>Assisting expert locating and knowledge sharing</td>
<td>Developer</td>
</tr>
<tr>
<td>Overlapped work</td>
<td>Reducing merge conflicts</td>
<td>Developer</td>
</tr>
</tbody>
</table>

(e.g., Outpost\(^1\) and Nozbe\(^2\)) so that practitioners can continue their work anytime, anywhere. A mobile approach to awareness support could fill the information gap of desktop awareness tools, and help construct a complete continuous awareness system.

4. THE TEAM RADAR SYSTEM

Team Radar is an implementation of continuous awareness on multiple platforms. Users can keep on using formal coordination mechanisms such as SCM and issue tracking systems, while they can also maintain awareness of other users’ ongoing changes.

Researchers on awareness have studied and collected lists of information needs from project managers and developers\(^22\), some of which are particularly important for distributed software development. Table 1 explains major awareness information supported by Team Radar, as well as its value in practice.

Figure 1 illustrates the architecture of the Team Radar system, where arrows represent information flow. The server side, Team Radar Server, is a central repository and message relay of awareness information. On the client side, Team Radar Desktop is an awareness information monitor and viewer on the desktop, embedded to Qt Creator\(^3\), a C++ Integrated Development Environment (IDE). Team Radar Desktop supports all major desktop operating systems, including Windows, MacOS, and Linux. Team Radar Mobile is a mobile awareness client with the same set of features as Team Radar Desktop, except that it does not capture awareness information. Team Radar Mobile supports two mobile operating systems: Symbian and Android.

Team Radar takes the following steps to disseminate awareness information in the team: capturing, dissemination, analysis, and visualization. Team Radar Desktop monitors and captures events of interest in local workspaces, and sends them to Team Radar Server, which broadcasts them to other registered clients. Team Radar Desktop and Team Radar Mobile use the same technique to analyze the received information, and present it with intuitive visualization. To promote continuous awareness of projects crossing multiple time zones, Team Radar supports offline playback. Users can download and replay awareness events stored in the repository specified by event type, development phase, developer, time span, etc. A set of analytical tools can mine the underlying patterns of collaboration, allowing managers to inspect daily activities, monitor the progress and analyze collaboration issues.

Figure 2 is a screenshot of the animated visualization. Team Radar adopts a tree structure\(^7\) to present the directory structure of a project. The tree is laid out aesthetically and automatically by a force-directed layout algorithm\(^14\) to fully utilize the screen space. Non-leaf nodes represent directories and are connected to the tree by edges. Leaf nodes denote files colored by their types. Each online developer is shown as an icon. When a developer starts making changes to a file, her icon will fly close to the corresponding tree node indicating the artifact she is working on. When an icon moves, its afterimage stays, and a light trail shows its track. The accompanying tag shows the developer’s current working mode (coding, debugging, etc.).

\(^1\)Outpost: [http://www.outpostapp.com](http://www.outpostapp.com)
\(^2\)Nozbe: [http://www.nozbe.com](http://www.nozbe.com)
When conflicting changes to the same artifacts occur, anclamation mark will give developers an early warning of the potential merge conflict. All local events are stored in the central repository as event scripts for users to retrieve and replay offline. A video demo of the visualization is available at [http://www.utdallas.edu/~congchen/Projects/TeamRadar/teamradar.html](http://www.utdallas.edu/~congchen/Projects/TeamRadar/teamradar.html).

5. VISUALIZING AWARENESS ON MOBILE PLATFORMS

Though Team Radar Mobile has almost the same set of features as Team Radar Desktop, the visualization has been fully optimized for mobile platforms. Figure 3 presents the UI and visualization of Team Radar Mobile.

5.1 Visual Metaphors

We believe that metaphor is a key factor of successful information visualization [6]. In order to create a virtual presence environment that promotes users’ cognition and interests, as well as to increase information density, Team Radar adopts several metaphors in its visualization based on the afterimage technique.

- Afterimage, or visual aftereffect, is an optical illusion that refers to an image continuing to stay in one’s vision after the original image is removed. Neural biologists now generally agree that aftereffects are not mere by-products of “fatiguing neurons”, but reflect neural strategies for optimizing information perception [31]. There is also evidence that afterimage stimulates eyes to track motion smoothly [18]. Afterimage is a critical technique to implement our metaphors. We argue that the afterimage technique, which embodies past and present information in our visualization, helps to stimulate the user’s interests and engagement.

- Radar is an important component of battlefield awareness, referring to knowledge of everything occurring on the battlefield [12]. On a typical radar screen, positions of targets are displayed as moving blips, typically with light trails showing their courses and directions. Similarly, Team Radar alerts developers where others were and are working on. We use the radar metaphor to create a notion that monitoring software team is just like observing a radar screen. In Team Radar, the tree layout mimics the polar coordinates of a radar system, icons simulate the blips of radar targets, and more interestingly, when an icon moves, its light trail shows the afterimage of the course.

- Memory metaphor refers to a common sense that the older the memory is, the more blurred the image appears in the mind [1]. As mentioned above, when the icon flies to a new position, the afterimage of the icon and the light trail remains on the screen and blurs out through time, mimicking a passing memory. The afterimage eventually disappears, and how long this process takes is customizable by the user, depending on how much past information the user intends to observe.

- “Picking apples”. When an apple is being picked from a tree, the tree branch will bend over and bounce back. This metaphor inspires the animation for developer icon attaching and detaching a tree node. When a developer icon flies to a tree node, the tree branch connecting the node will be elongated towards the developer and restore when the developer leaves.

The afterimage technique, the radar metaphor, and the memory metaphor together create an illusory environment that allows users to traverse between past and present. The “picking apples” metaphor and the flying icons create a notion that the project is a living organization.

5.2 Optimization Techniques

Team Radar Mobile shares most features with Team Radar Desktop. However, making a mobile version of an application is not a simple reimplementation of its desktop counterpart. It involves much attention to the specific requirements for mobile platforms. The characteristics of mobile platforms, such as smaller screen, limited computation power and battery life, and users’ intermittent focus (contrasted to more continuous focus on desktop computers) [30] pose special requirements for mobile application design.

Team Radar Mobile is designed to meet the following requirements for mobile visualization. Other general optimizations for visualization are discussed in our previous publication [3].

1. Keep style consistent with Team Radar Desktop.
2. Performance is tuned for mobile devices.
3. Screen space is better utilized.
4. Support touch UI.
5. Suit mobile users’ intermittent usage.
6. Preserve power consumption.

To ease the development for multiple platforms as well as to keep a consistent style, the entire Team Radar system is developed with C++ and [Qt](http://qt.nokia.com/). Qt is a cross-platform application and UI framework. It ensures the application has a consistent behavior across multiple platforms while remaining a native look and feel for each platform. The same set of metaphors further ensures the users can perceive the same visual effects.

We take several measures to optimize the performance on mobile devices. First, the visual effects are customizable. Users can adjust the frame rate of the animation and other parameters according to their hardware performance. This measure allows the application to support low-end hardware without compromising the basic visual experience. Second, the force-directed graph layout algorithm is customized for our specific context. Though aesthetically appealing and flexible, the classic force-directed layout algorithm does not scale well, with the worst running time of \( O(|V|^3) \), \(|V|\) being the number of vertices [14]. Since the graph in Team Radar is a hierarchical tree, we utilize the local nature of the sub-trees and develop a simplified multi-scale force-directed
layout algorithm \cite{14}, which takes into account only siblings in the same sub-tree and ancestors when relocating a node. According to our experiment, the performance of the algorithm is greatly improved for large trees. The following is the pseudo code of the improved layout algorithm.

\begin{verbatim}
customized-force-directed-layout() {
    while (!converged()) {
        Queue q;
        q.enqueue(root);
        while (!q.empty()) {
            t = q.dequeue();
            for (all ancestor nodes n of t) {
                pull(t, n);
                push(t, n);
            }
            for (all sibling nodes n of t) {
                push(t, n);
            }
            q.enqueue(t.children());
        }
    }
}
\end{verbatim}

The scalability and readability of a visualization is often affected by excessive information. In Team Radar, the project tree is expanded on demand. Because developers’ behavior also exhibits certain local nature \cite{21}: no matter how the project scales, one developer usually works on a small subset of the artifacts, there is no need to expand the entire tree. Initially, Team Radar only loads the root of the tree. When a user opens a file, Team Rader will automatically expand the nodes along the path from the root to the file, and keep other nodes folded. Expand-on-demand keeps the screen clean and significantly improves the scalability and performance of the system by showing a minimal subset of the nodes.

Team Radar Mobile fully supports multi-touch. Users can pinch to zoom, pan to move the screen, tap to select an object, tap and hold to enter and leave full screen mode, and double tap to enter or leave the camera mode. The camera mode helps minimize the number of gestures needed to focus on a specific developer. Double tapping a developer icon enters the camera mode, in which the screen will be zoomed in to focus on the developer and the developer will always stay in the center. Double tapping an empty space leaves the camera mode, and the screen will be zoomed out to display the entire content.

As contrasted to desktop computers, mobile devices are only used intermittently. Team Radar Mobile supports push notification and offline playback to avoid unnecessary interruptions. The user only needs to open Team Radar Mobile and download the events offline when she receives a push notification on an update on the server. Offline playback allows users in different time zones to work together. Push notification and offline playback also help reduce power consumption.
Table 2: Experiment Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Tested features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Find the most conflicted file</td>
<td>Conflict warning, Work dependency</td>
</tr>
<tr>
<td>2. Find who conflicted with Mike</td>
<td>Work dependency, Project progress</td>
</tr>
<tr>
<td>3. Find the most active developer</td>
<td>Expert locating, Developer activeness</td>
</tr>
<tr>
<td>4. Tell what phase the project is most likely at (UI design, coding, or testing)</td>
<td>Project progress</td>
</tr>
<tr>
<td>5. List the files Mike has edited</td>
<td></td>
</tr>
<tr>
<td>6. Tell who has edited rental.cpp</td>
<td></td>
</tr>
</tbody>
</table>

6. EVALUATION

The effectiveness and efficiency of the visual mobile approach to continuous awareness are tested by a small group of users. Participants were asked to finish several awareness related tasks and answer corresponding questions.

6.1 Experiment Setup

To test the effectiveness and efficiency of Team Radar Mobile for expressing awareness information, we measured and compared user performance on a series of awareness related tasks with and without Team Radar Mobile. There are two configurations of the system. Configuration 1 disables the visualization module on Team Radar Mobile. The awareness events received on the mobile client are displayed textually. Configuration 2 enables the visualization module and allows participants to use the visualization to help answer the questions.

The awareness events are from our previous experiment for Team Radar Desktop. The system had already recorded a set of awareness events while developers were working on several programming tasks. In this experiment, the participants were asked to download those events, review them, and perform several task based on what they have viewed. There are 55 events, and the visualization lasts 47 seconds. The tasks are design to cover all the features of Team Radar Mobile (see Table 1 for details). Participants were allowed up to 180 seconds for each task. Table 2 lists the tasks.

6.2 Process

The subjects of the study were recruited as volunteers. The test used 8 graduate students from the Department of Computer Science, University of Texas at Dallas and 6 professionals in software industry. Their professional profiles were gathered before the test to ensure they have the required skills and experience. We used a between subjects design for the test, i.e., the subjects were divided into two groups, each for one configuration. Each group contains 4 students and 3 professionals. Group 1 used configuration 1 and group 2 used configuration 2.

The evaluation was conducted in three phases. First, there was a training session for the participants to get familiar with the system. Second, a pilot study involving a small group of participants was conducted to refine the tasks, ensuring the experiment environment is functional and the tasks can be completed within reasonable time. Any technical problems arose in the pilot study were solved before the test. Finally, we proceeded with the test and compared their performance.

6.3 Result

The performance of participants on the tasks are measured by the average correctness and completion time (in seconds) of their answers to the questions associated with the tasks. Those tasks that were not finished within the allowed time were considered taking 180 seconds. As shown in Table 3, group 2 outperformed group 1 for most of the tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Configuration 1</th>
<th>Configuration 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>57% 152</td>
<td>86% 127</td>
</tr>
<tr>
<td>2</td>
<td>71% 171</td>
<td>100% 89</td>
</tr>
<tr>
<td>3</td>
<td>100% 102</td>
<td>86% 53</td>
</tr>
<tr>
<td>4</td>
<td>43% 167</td>
<td>57% 114</td>
</tr>
<tr>
<td>5</td>
<td>86% 144</td>
<td>86% 134</td>
</tr>
<tr>
<td>6</td>
<td>100% 87</td>
<td>100% 64</td>
</tr>
</tbody>
</table>

As we had expected, the visual approach outperforms traditional approach the most for those tasks that require a comprehensive understanding of the project, such as task 1, 2, and 4. Many participants could get the answer right after reviewing the animation. Users without the visual assist, however, often have to traverse the events again and again to find out the required information.

For tasks that only need to scan the event script once (e.g., task 5 and 6), Team Radar Mobile did not improve the correctness on them, as they are already straight-forward. But visualization is still able to improve the speed on those tasks.

One participant in group 2 miscounted the second active developer as the most active one, contributing to the loss of visual approach for task 3.

6.4 Discussion

Though the visualization Team Radar Mobile provides significantly improves the effectiveness and efficiency of the awareness system, the participants still pointed out some possible improvement of current system.

The analytical features of the current version of Team Radar
are still primitive. In some cases (e.g., task 4), though the visualization has encoded the information visually, the viewers still could not get the answer directly. Some participants suggest it will be simpler if the information is analyzed automatically and the result is shown directly.

Team Radar applies the memory metaphor (i.e., afterimage and light trail) to help present both past and present information at the same time. However, the historical information embodied in afterimage is limited. Users can configure how much past information to be shown by adjusting how long afterimage stays. Keeping the afterimage and light trail for too long, however, may pollute the screen. For example, in task 5, the participants were able to find out the recent files Mike has edited from the light trail of Mike’s icon. But the light trail only shows the recent several seconds’ history. The viewer still need to review the entire animation to get the complete answer.

Team Radar uses some simple heuristics to provide the high-level awareness information. Some of them (e.g., editing files means being active) are found impractical for complex projects.

7. CONCLUSIONS
This paper has presented a research on continuous awareness, including its concept, design, implementation, and evaluation. Continuous coordination aims to combine the strengths of informal awareness-based approaches with those of formal process-based approaches. Continuous awareness is an extension of this paradigm on awareness support, addressing the need of continuous and integrated awareness information. We have implemented a continuous awareness system by the cooperation of desktop and mobile platforms. The visual mobile approach complements the insufficiency of desktop platforms regarding continuous awareness support. The design of Team Radar Mobile follows the principles of mobile visualization, and is aided with several visual metaphors and physiological theories. The effectiveness of this approach is validated with a controlled experiment. Future work includes implementing more analytical features and a comprehensive evaluation of the entire Team Radar system.

8. REFERENCES


