Journal of Visual Languages and Computing 24 (2013) 390–401

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Continuous awareness: A visual mobile approach

Cong Chen[a], Wenyuan Tao[b,*], Kang Zhang[b,a]

[a] Department of Computer Science, The University of Texas at Dallas, USA
[b] School of Computer Software, Tianjin University, China

ARTICLE INFO

Available online 1 August 2013

Keywords:
Awareness
Mobile visualization
Continuous awareness
Global software development

ABSTRACT

Facing the challenges of global distribution in software development, Continuous Coordination constitutes a new coordination paradigm that helps break the communication barriers in distributed teams by providing awareness information and integrating heterogeneous tools. Continuous Awareness is an extension of Continuous Coordination emphasizing continuous awareness support across space and time. Traditional desktop-based approaches are insufficient for the requirements of continuous awareness. Team Radar Mobile takes a visual mobile approach to awareness by extending the visualization of awareness information on desktop platforms to 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 found visualization express awareness information more efficiently than the non-visual approach. Our work also provides experience on mobile visualization.

1. Introduction

Software development is a typical group activity. Such teamwork requires intense collaboration among team members as well as other outside stakeholders. To foster innovation and competition, and to minimize cost, more and more software teams are becoming distributed [1]. Geographical distribution incurs many challenges to collaboration, such as physical, social, and cultural barriers, which consequently obstruct the channel of collaboration [2].

Awareness, “an understanding of the activities of others that provides a context for one's activities” [3], is an important contributor to effective team collaboration, as it “aids coordination of tasks and resources, and assists transitions between individual and shared activities” [4]. A number of researchers have argued that the key to promote collaboration in distributed teams is to increase the level of awareness and provide continuous information of ongoing changes [5,2,6].

In software development practice, there exist two major coordination paradigms: formal process-based approaches and informal awareness-based approaches [7]. Formal approaches separate work into multiple independent tasks that are periodically resynchronized. A typical example is software configuration management (SCM) systems, through which developers constantly synchronize local copies to ensure the consistency of the parallel work. Formal approaches are well-disciplined and scalable, but need manual attention and may isolate conflicted local changes. 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 with informal approaches is that they usually do not scale well, because of users' cognitive limitations and lack of process support [7].

Continuous Coordination (CC) [7] is a new coordination paradigm that combines the strengths of formal and informal approaches by retaining the synchronization mechanisms of the formal approaches, but providing developers with a view of each other's relevant activities between formal synchronizations. Fig. 1 illustrates the concept of CC. A key to successful CC system is to integrate

* Corresponding author.
E-mail address: taowenyuan@tju.edu.cn (W. Tao).

1045-926X/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jvlc.2013.07.003

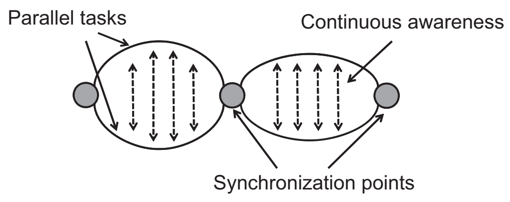


Fig. 1. Continuous awareness for continuous coordination.

heterogeneous (formal and informal) tools seamlessly and provide awareness information continuously. We argue that *Continuous Awareness* (*CA*), a continuous support for awareness information across space and time, is essential for CC in distributed software teams.

CA emphasizes continuous awareness support in two aspects: continuous in the spectrum of awareness types, and continuous across space and time. Software developers often need multiple types of awareness, such as presence awareness, workspace awareness, and social awareness, and these needs may change with the context of the work [6]. Although CA can be partially supported by many existing tools, such as instance-messaging (for presence awareness and communication) and synchronous online editors (e.g., Google Docs, for activity awareness), a more integrated system that is customized for software development is still missing.

CA inherently requires multi-platform cooperation. 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 a person's role, time, and place [8,1]. Users' mobility sometimes disables the effectiveness of desktop awareness tools [9], and a mobile awareness tool would complement the insufficiency of CA 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" [10]. To our knowledge, however, there have been no mobile awareness tools available in the software engineering community and the effectiveness of mobile awareness is unclear.

Based on our previous work on desktop awareness, Team Radar [11], we have developed a mobile awareness tool called *Team Radar Mobile*. From now on, we call Team Radar on desktop *Team Radar Desktop* and the entire system on both desktop and mobile platforms *Team Radar*. Together with Team Radar Desktop, Team Radar Mobile achieves CA using a *visual mobile approach*, i.e., visualizing awareness information on mobile platforms. Our work represents the first attempt at realizing CA through the cooperation of both desktop and mobile platforms. This paper introduces the concept of CA, its design and implementation on multiple platforms, and a preliminary evaluation on the usability of the visual mobile approach.

The rest of the paper is organized as the following. Section 2 lays out the background of our work, followed by an introduction of continuous awareness in Section 3. The design of the entire Team Radar system and the design of Team Radar Mobile are discussed in Sections 4 and 5 respectively. We report a preliminary evaluation of the visual mobile approach in Section 6, and discuss its implications in Section 7. Finally, Section 8 concludes the paper with future work.

2. Related work

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

Learned from the CSCW community, awareness is now considered an important informal approach to many collaboration problems in software development [6]. Some representative result of such research includes workspace awareness tools (e.g., Palantír [12] and CollabVS [13]) for alleviating merge conflict problems in shared workspace and social awareness tools (e.g., Ariadne [14] and Calefato et al. [15]) for increasing the sense of "teamness" [15]. Most of the existing awareness tools, however, are designed for desktop platforms, and do not meet the need of CA.

The CSCW literature has already demonstrated the value of mobile platforms for awareness support. Presence awareness tools, such as IPAD (portable Inter-Personal Awareness Device) [10] and FriendZone [16], provide availability and proximity information about other mobile users. Context awareness tools (e.g., ContextContacts [17]) give users cues on the context of other users helping determine the best timing and channel of communication [18]. The ConNexus and Awarenex projects [19] present design principles and lessons learned in extending awareness services from desktop to mobile platforms. Papadopoulos [20] studied collaboration in mobile platforms and generalized key requirements for awareness in mobile CSCW. Team Radar Mobile follows several key design principles proposed in the mobile awareness community [19,20], as discussed in Section 5.

The limitations of mobile devices impose many challenges to information visualization. Chittaro [21] suggests 6 steps for designing mobile visualizations. Burigat and Chittaro [22] studied and compared the Overview+Detail visualization on desktop and mobile platforms. Despite the research on graph visualization on desktop computers [23], visualizing graph on mobile devices is seldom addressed. RELT [24,25] visualizes multiple levels of a hierarchical structure on a single view for small screens. Tablorer [26] shows a tree structure using an expandable table format. Team Radar Mobile improves the force-directed layout algorithm on desktop [27] to better utilize screen space for mobile devices.

3. Continuous awareness

Coordinating large-scale collaboration is a difficult task. In software development practice, there exist two major paradigms of coordination: formal process-based approaches and informal awareness-based approaches [7]. Formal approaches rely on predefined process models and tools to ensure synchronization of collaborators' work. The mechanism of formal approaches can be demonstrated by SCM systems.

Developers first check out the artifacts from a SCM repository to get the up-to-date version of the work. Then they can work on their local copies independently. When a 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 [12].

To mitigate the information isolation between formal synchronizations, informal approaches proactively share awareness information on others' activities and local changes, allowing developers to conduct self-coordination and avoid conflicting actions. A major problem with informal approaches is that they usually do not scale well, because of users' cognitive limitations and lack of process support.

Traditional practices treat these two paradigms oppositely. Users usually choose either formal approaches or informal approaches, and cannot benefit from both of them in an integrated environment [7]. Continuous Coordination (CC) breaks this dichotomy and combines elements of both formal and informal approaches [7]. It retains the isolation and synchronization mechanisms of the formal approaches, but also provides informal awareness information between synchronization points. An example of CC can be found in workspace awareness tools such as Palantír [12]. Palantír automatically informs developers of potential conflicts arising from dependency violations in concurrent changes, so that the possibility of merge conflicts can be reduced. Palantír unobtrusively integrates lightweight visualizations (small icons and decorations) into the Eclipse IDE, allowing developers to focus on their main work while getting notified of emerging problems in the SCM workspace.

CC proposes several design principles for implementing coordination systems, including multiple perspectives, unobtrusive integration, combination of social and technical factors, and integrated formal and informal coordination approaches [7]. However, one important principle is not emphasized in the original model: *Continuous Awareness* (*CA*), the continuous acquisition and processing of awareness information across space and time. Existing principles of CC 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 CA is essential for CC in distributed software teams.

CA emphasizes continuous awareness support in two aspects: continuous in the spectrum of awareness types, and continuous across space and time.

As Omoronyia et al. [6] pointed out, in distributed software teams, developers often work on different tasks and artifacts, at different times, with different colleagues, in different roles, and form different perspectives – awareness need is highly contextual. Unlike traditional paradigms of awareness research, which study single awareness type individually [28], CA requires providing integrated awareness information to meet users' changing needs contextually. Some existing awareness tools have already followed this design paradigm. Holmes and Walker proposed the notion of Developer Specific Awareness (DSA) [29] and realized it in YooHoo [30], by which awareness information about code changes is filtered based on a developer's own code and interests. Another example can be found in the Jazz platform,[1] which incorporates many collaboration features into traditional integrated development environment (IDE). In addition to Jazz's workspace awareness and presence awareness support, Calefato et al. built a Jazz extension to add social awareness service into Jazz [31]. The extendable architecture of Jazz provides a strong support for integrating a broad range of awareness elements.

CA inherently requires its implementation to support multiple platforms. Current awareness systems are restricted to desktop platforms, which are insufficient for distributed collaboration in terms both space and time. 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. In modern offices, employees may work at variable locations or in mobile. Much time can be spent in meetings, visiting customers, or moving between locations. At the same time, spontaneous meeting and group discussions continue to be an important factor in work [10]. Herbsleb et al. [32] and Teasley et al. [33] have both found that unplanned exchange of information can lead to spontaneous collaboration, such as code inspection, pair programming, and problem solving, which is one of the reasons co-located teams perform better than distributed teams [13].

Awareness is an important virtual channel to support such ad-hoc collaboration among members of distributed teams [33], and mobile devices provide unbeatable flexibility than desktop or laptop computers, bringing information, availability, and efficiency to any stage and component of a software process. For instance, it has become a common practice for many project management tools to offer a mobile version (e.g., Outpost[2] and Nozbe[3]), so that practitioners can continue their work anytime, anywhere. A mobile awareness approach could fill the information gap of desktop awareness tools, and help construct a complete CA system.

4. The team radar system

Team Radar is an implementation of CA 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.

4.1. Supporting continuous awareness for distributed teams

People with different roles in a software team have different information needs and communication patterns [8]. Project managers typically have wider range of information need, as they concern about project wide issues, such as progress, budget, schedule, risks, etc. Project

[1] Jazz: http://jazz.net.
[2] Outpost: http://www.outpostapp.com.
[3] Nozbe: http://www.nozbe.com.

Table 1
Awareness information supported by Team Radar.

Information need	Benefit	Benefited party
Project status	Monitoring project progress	Manager
Activeness of developers	Understanding developers' workload	Manager
Collaboration patterns	Understanding team organization and group dynamics	Manager
Others' activities	Understanding work dependency	Developer
	Assisting expert locating and knowledge sharing	
Overlapped work	Reducing merge conflicts	Developer

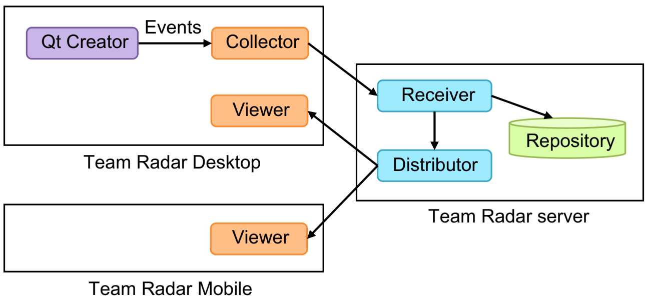


Fig. 2. The architecture and key components of the Team Radar system.

managers often travel, and often meet with other people within and outside the team. On the other hand, developers focus more on finishing their development tasks in time and ensuring the quality of the work. Their communications are usually around technical issues, such as requirements, design decisions, code changes, bug status, etc [34,35]. Developers work on desktop/laptop computers most of the time and communicate mostly with people within the organization. In globally distributed teams, geographical distance imposes more challenges to collaboration. Team members sometimes have to work out of office hours to cope with time difference [8].

To meet the need of CA in distributed teams, Team Radar is designed to support different roles and different usage patterns. Team Radar consists of a desktop client and a mobile client. The desktop client can be used by any roles in a team who need unobtrusive continuous awareness of the team dynamics. The mobile client is especially useful for people who often work out of office or office hours (e.g., project managers), and also promotes ad-hoc collaboration. Team Radar supports both online and offline mode. The online mode allows users to capture the most up-to-date awareness information. To promote CA for projects crossing multiple time zones, Team Radar supports offline mode, in which awareness information can be downloaded upon request and played asynchronously. Moreover, push notification on Team Radar Mobile ensures that developers will not miss any important updates or collaboration opportunities. Table 1 summarizes major awareness information types supported by Team Radar, as well as their value in practice.

4.2. Architecture and features

Fig. 2 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,[4] a C++ IDE. Team Radar Desktop supports all major desktop operating systems, including Windows, Mac OS, 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. A set of analytical tools can mine the underlying patterns of collaboration, allowing managers to inspect daily activities, monitor progress, and analyze collaboration issues.

Fig. 3 is a screenshot of the animated visualization. Team Radar uses a tree [36] to present the directory structure of a project. The tree is laid out automatically and aesthetically by a force-directed layout algorithm [27]. Non-leaf nodes represent directories and are connected to the tree by edges. Leaf nodes denote files and are 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

[4] Qt Creator: http://qt-project.org/downloads.

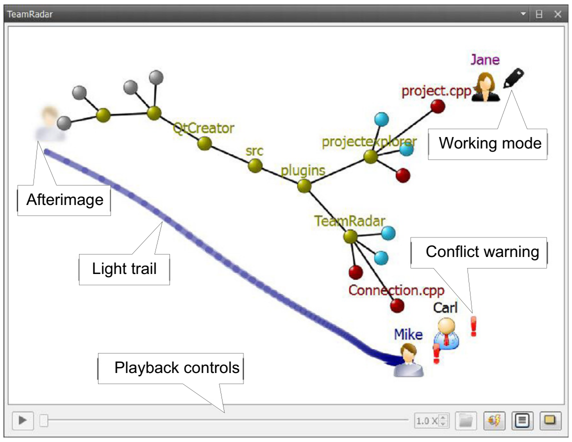


Fig. 3. The visualization on Team Radar Desktop.

afterimage stays, and a light trail shows its track. The accompanying tag shows the developer's current working mode (coding, debugging, etc.). When conflicting changes to the same artifacts occur, an exclamation 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. As time intervals between events could be very long, in the offline mode, users are allowed to adjust the playback speed and navigate to certain event through the playback controls. Video demos of the visualization on different platforms are available at http://www.utdallas.edu/~cxc094020/Projects/ContinuousAwareness.

4.3. *Visual metaphors*

We believe that metaphor is a key factor of successful information visualization [37]. 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. The metaphors are consistently implemented in all the desktop and mobile platforms.

- 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 [38]. There is also evidence that afterimage stimulates eyes to track motion smoothly [39]. 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 users' interests and engagement.
- Radar is an important device for battlefield awareness, providing knowledge of everything occurring on the battlefield [40]. On a typical radar screen, positions of targets are displayed as moving blips, 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 a memory is, the more blurred the image appears in the mind [41]. As mentioned above, when an icon flies to a new position, the afterimages of the icon and the light trail remain on the screen and blur 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 a developer icon attaching to and detaching from 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 restored 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.

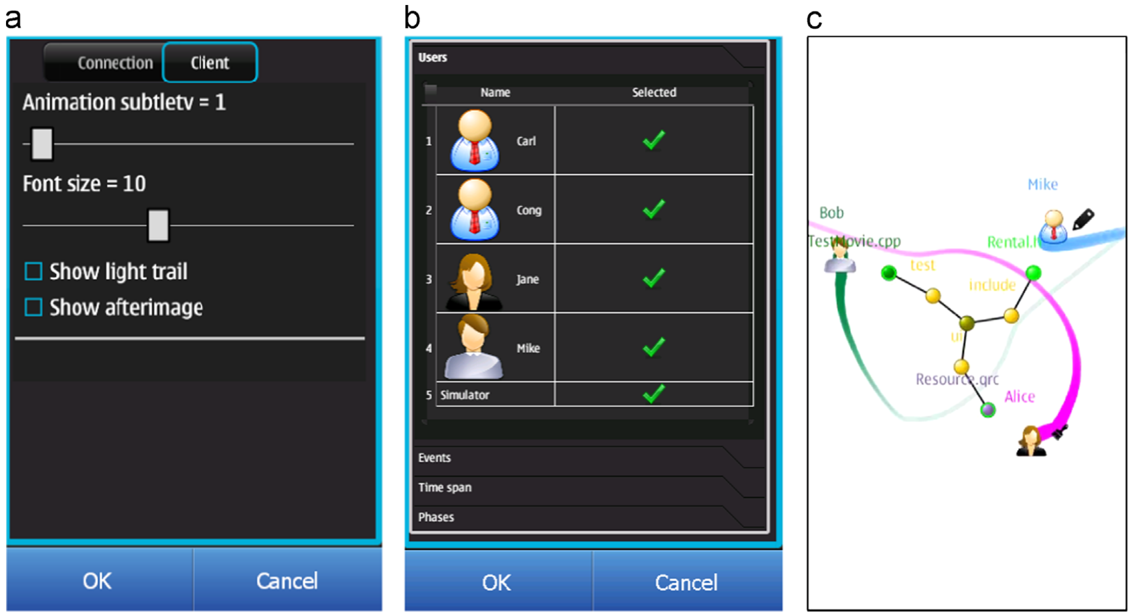
5. Visualizing awareness on mobile platforms

Though Team Radar Mobile has almost the same set of features as Team Radar Desktop, the visualization has been fully optimized for mobile platforms. Fig. 4 presents the visualization of Team Radar Mobile running on Symbian.

Team Radar Mobile shares most features with Team Radar Desktop. Making a mobile version of an application, however, 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) [19] impose special requirements for mobile application design.

Team Radar Mobile is designed to meet the following requirements for mobile visualization we have identified [20]. In additional to general optimizations for visualization discussed in our previous publication [11], we have innovated several techniques for mobile visualization, including a new multi-level force-directed layout algorithm considering screen boundary and shape.

1. Maximize the performance for mobile devices.
2. Better utilize screen space.
3. Support touch UI.
4. Suit mobile users' intermittent use.
5. Minimize power consumption.
6. Keep style consistent with Team Radar Desktop.



Animation subtletv = 1

Font size = 10

☐ Show light trail

☐ Show afterimage

Users

	Name	Selected
1	Carl	✓
2	Cong	✓
3	Jane	✓
4	Mike	✓
5	Simulator	✓

Events

Time span

Phases

Fig. 4. The UI and visualization of Team Radar Mobile. (a) Customization dialog. (b) Download events offline. (c) Visualization.

5.1. Optimizing performance

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 the performance of their hardware. This measure allows the application to support low-end hardware without compromising the basic visual experience. Second, the force-directed graph layout algorithm [27] is customized for our specific context. Though aesthetically appealing and flexible, the classic force-directed layout algorithm does not scale well, with the worse running time of $O(|V|^2|E|)$ for each iteration, $|V|$ being the number of vertices, and $|E|$ the number of edges [27]. Since the graph in Team Radar is essentially a tree, we utilize the local nature of the sub-trees and developed a simplified multi-scale force-directed layout algorithm [42], 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. Algorithm 1 shows the pseudo code of the improved layout algorithm, which performs a breadth-first traverse of the tree, and for each node, applies repulsive forces (from its ancestors and siblings) and attractive forces (from its edges) to it.

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 [43]: no matter how a 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 Radar 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.

Algorithm 1. A multi-scale tree layout algorithm.

```
procedure LAYOUT(Tree $T$)
    create a queue $Q$
    $Q.enqueue(T.root())$
    while not $converged()$ do
        $n \leftarrow Q.dequeue()$
        $rf \leftarrow (0,0)$            // repulsive force
        for all $a \in n.ancestors()$ do
            $rf \leftarrow rf + repulse(n,a)$
        end for
        for all $s \in n.siblings()$ do
            $rf \leftarrow rf + repulse(n,s)$
        end for
        $af \leftarrow (0,0)$            // attractive force
        for all $e \in n.edges()$ do
            if $e$ connects $n.parent()$ then
                $af \leftarrow af + attract(n,e)$
            end if
        end for
        $p \leftarrow t.pos() + rf + af$            // combined force
        $n.setPos(p)$
    end while
end procedure
```

Algorithm 2. A multi-scale tree layout algorithm with boundary constraint and distortion.

```
procedure LAYOUTWITHBOUNDARY (Tree $T$, Rectangle $r$)
    create a queue $Q$
    $Q.enqueue(T.root())$
    while not $converged()$ do
        $n \leftarrow Q.dequeue()$
        $rf \leftarrow (0,0)$            // repulsive force
        for all $a \in n.ancestors()$ do
            $rf \leftarrow rf + repulse(n,a)$
        end for
        for all $s \in n.siblings()$ do
```

```
11:         $rf \leftarrow rf + repulse(n, s)$
12:       end for
13:       $af \leftarrow (0, 0)$              // attractive force
14:       for all $e \in n.edges()$ do
15:         if $e$ connects $n.parent()$ then
16:           $af \leftarrow af + repulse(n, e)$
17:         end if
18:       end for
19:       $rf \leftarrow Distortion(rf, r)$          // apply distortion
20:       $p \leftarrow t.pos() + rf + af$          // combined force
21:       $p \leftarrow Boundary(p, r)$           // apply boundary
22:       $n.setPos(p)$
23:     end while
24: end procedure
25: procedure BOUNDARY(Position $p$, Rectangle $r$)
26:     $w \leftarrow r.width()$
27:     $h \leftarrow r.height()$
28:     $p.x \leftarrow min(w/2, max(-w/2, p.x))$
29:     $p.y \leftarrow min(h/2, max(h/2, p.y))$
30:     return $p$
31: end procedure
32: procedure DISTORTION(Force $f$, Rectangle $r$)
33:     $distortion \leftarrow r.width()/r.height()$          // aspect ratio
34:     $f.x \leftarrow f.x * distortion$
35:     return $f$
36: end procedure
```

5.2. Better utilizing screen space

Another challenge to the tree layout algorithm is its screen utilization. Unlike desktop platforms, where screen utilization may not be a critical issue, mobile devices have much smaller screens, and cannot afford wasting screen space. In order to ensure visual consistency, Team Radar Mobile reuses the node-link presentation for project structure from Team Radar Desktop. Traditional tree layout algorithms, such as force-directed algorithms, usually result in a symmetric layout, which does not fit rectangular small screens. Fig. 5a illustrates the effect of Algorithm 1 on a (fully expanded) project structure displayed on a small screen. The tree nodes are evenly distributed without considering screen boundary or screen shape. The user has to zoom-out to see the entire tree on the screen, leading to more waste of screen space.

Despite the large body of research on graph drawing [23], only a few of them address visualizing tree structures on small screens [24–26]. To improve its screen utilization, we apply two techniques to Algorithm 1, considering screen boundary and screen shape respectively. We first use Fruchterman's approach [44] to confine the graph within the screen boundary. A screen border acts as a wall to stop the component of an escaping node's displacement normal to the wall. Fig. 5b shows the effect of the boundary constraint. The nodes that are placed out of the screen boundary in Fig. 5a are now squeezed into the screen and placed along the screen edges. Given a rectangular screen in portrait orientation, the area close to the left and right edges may become too crowded, leaving the top and bottom portions of the screen underused. We then add a distortion to the layout according to the aspect ratio and orientation of the screen. With the distortion, the nodes are more likely to be placed along the longer dimension of the screen. As illustrated by Fig. 5c, with the boundary constraint and the distortion, the nodes are more evenly distributed and fill in most of the screen. In case the graph is too large to fit in the screen or the user wants to see the details of a large graph, the boundary constraint will be temporarily disabled to allow the user to zoom in and out. Algorithm 2 shows the pseudo code of the boundary constraint and distortion.

5.3. Supporting touch UI

Team Radar Mobile fully supports multi-touch. Users can pinch to zoom, swipe 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.

5.4. Supporting intermittent use and reducing power consumption

As contrasted to desktop computers, mobile devices are only used intermittently [19]. Team Radar Mobile supports push notification and offline playback to avoid unnecessary focus and obtrusive interruptions. Users only need to open Team Radar Mobile and download the events offline upon receiving 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.

5.5. Keeping consistent style

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.[5] Qt is a cross-platform application and UI framework. It ensures the application to have a consistent behavior across multiple platforms while remaining a native look and feel for each platform. The same set of metaphors and visual presentations further ensures the users to perceive the same visual effects.

6. Evaluation

The effectiveness and efficiency of the visual mobile approach to presenting awareness information have been evaluated by a controlled experiment involving a small group of users (subjects). The subjects were asked to finish several awareness perception tasks and answer corresponding questions.

[5] Qt: http://qt-project.org/.

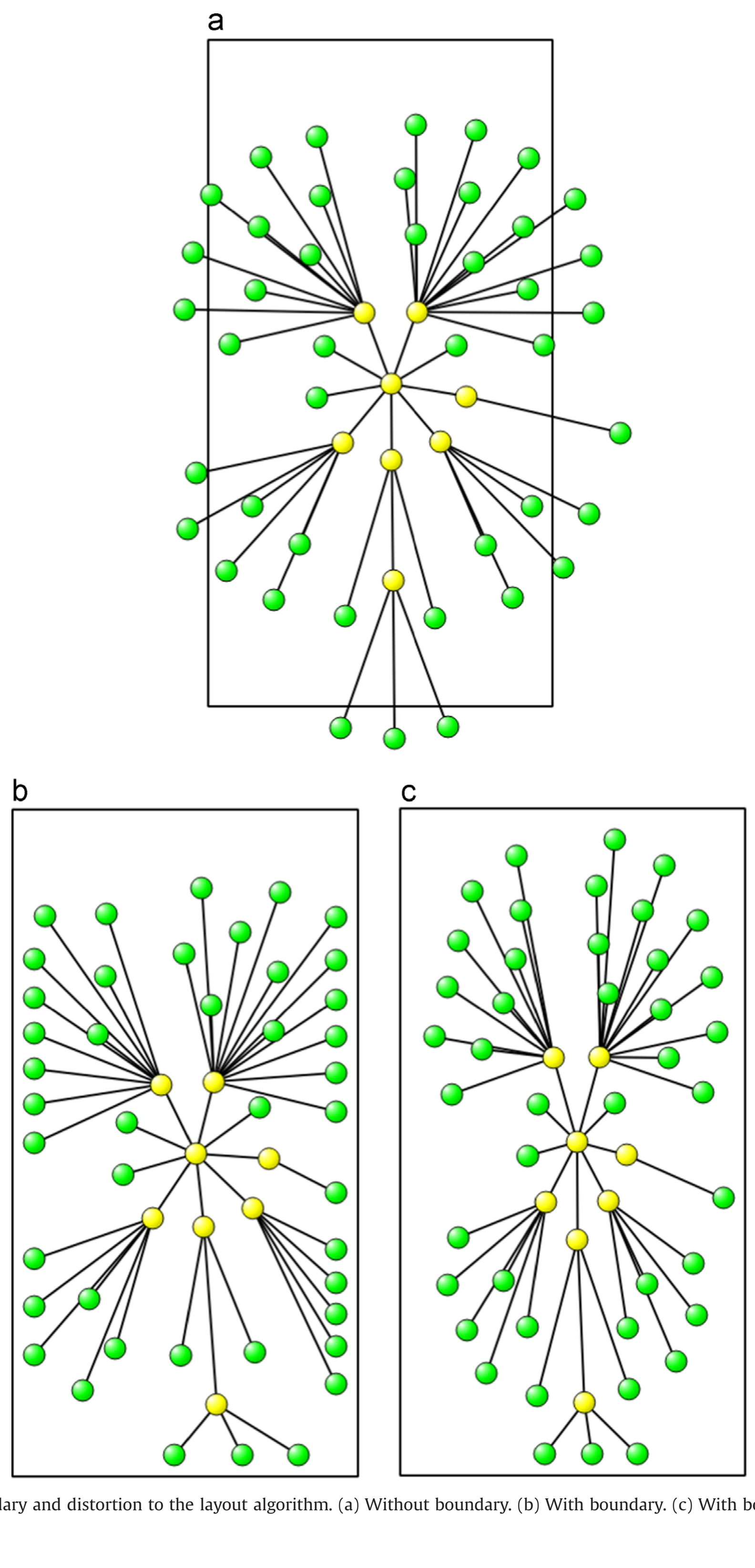


Fig. 5. Applying boundary and distortion to the layout algorithm. (a) Without boundary. (b) With boundary. (c) With boundary and distortion.

6.1. Research questions and hypotheses

The experiment intends to answer the following research questions.

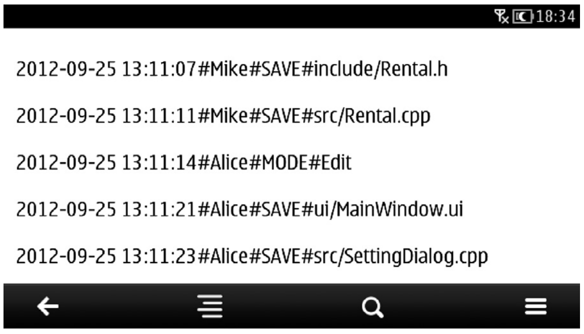
Q1: Does the visualization on Team Radar Mobile increase the correctness of the answers to the awareness perception questions, compared to non-visual approaches?
Q2: Does the visualization on Team Radar Mobile reduce the time needed for the awareness perception tasks, compared to non-visual approaches?

The hypotheses associated with the research questions are:

H1: The visualization on Team Radar Mobile increases the correctness of the answers to the awareness perception questions.
H2: The visualization on Team Radar Mobile reduces the time needed for the awareness perception tasks.

6.2. Variables

The purpose of the experiment is to compare the performance of the visualization on Team Radar Mobile with the non-visual approach for presenting awareness information. The independent variable is the configuration of the awareness system. It has two values: Team Radar Mobile without visualization and Team Radar Mobile with the visualization. The dependent variables are correctness of the answers to the tasks and their completion time, measuring the effectiveness and efficiency of the system respectively.



```
2012-09-25 13:11:07#Mike#SAVE#include/Rental.h

2012-09-25 13:11:11#Mike#SAVE#src/Rental.cpp

2012-09-25 13:11:14#Alice#MODE#Edit

2012-09-25 13:11:21#Alice#SAVE#ui/MainWindow.ui

2012-09-25 13:11:23#Alice#SAVE#src/SettingDialog.cpp
```

Fig. 6. The textual presentation of awareness events.

6.3. Experiment setup

To test the hypotheses, we measured and compared user performance on a series of awareness perception tasks with and without using visualization on 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, allowing the subjects to use the visualization to help answer the questions. Figs. 4c and 6 show the screenshots of the visual and textual presentations of the events used for two groups respectively.

The awareness events were from our ongoing experiment for Team Radar Desktop. Team Radar Server had already recorded a set of awareness events while developers were working on a software project with Team Radar Desktop. The project was adapted from an example in Fowler et al. [45], a canonical reference on refactoring. The example demonstrates how a professional developer would improve the design of existing software code through a series of refactoring. Each refactoring may include several steps. Developers were asked to perform some of the representative steps, involving renaming, moving methods, extracting methods, creating classes, etc. The changes to the code and developers' interactions with the IDE were recorded by Team Radar Desktop as event scripts and saved in Team Radar Server.

In this experiment for Team Radar Mobile, the subjects were asked to download previously stored events (see Fig. 4b), review them (Figs. 4c and 6), and answer several questions based on what they have viewed. There are 55 events, and the visualization lasts 47 seconds (with playback speed set to 2 times). Subjects were allowed up to 180 seconds for each task. Table 2 lists the tasks.

The tasks and relevant questions cover all the information aspects of Team Radar (see Table 1 for details), and test all major visual features of Team Radar Mobile, including UI, visualization on small screen, and performance. The

Table 2
Experiment tasks.

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

experiment was conducted using Nokia C7 devices with a 3.5″ touchscreen and 640 × 360 resolution.

6.4. *Process*

The subjects of the study were recruited as volunteers. The test used 8 graduate students from the Department of Computer Science, The University of Texas at Dallas and 6 professionals in software industry. Their professional profiles were gathered before the test to ensure that 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 consists of randomly selected 4 students and 3 professionals. Group 1 used configuration 1 and group 2 used configuration 2.

The evaluation was conducted in three phases. First, a pilot study involving a small group of subjects (different from the experimental study subjects) was conducted to refine the tasks, ensuring that 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. Then, there was a training session for the subjects to get familiar with the system. Finally, we proceeded with the test and compared subjects’ performance.

6.5. *Results*

The performance of the groups are measured by the average correctness (%) and completion time (in seconds) of the subjects in each group per task. As shown in Table 3, group 2 outperformed group 1 for most of the tasks. We performed paired t-test on the data shown in Table 3 to examine whether the improvements are significant. As a prerequisite, we first conducted Shapiro-Wilk normality test to ensure that the data follows normal distribution. Using paired t-test, we find that the mean correctness of group 2 is 12.7% higher than that of group 1 with mean difference being 9.67%. The two-tailed p-value is 0.232 and is not significant at the 0.95 confidence level. We hypothesize that the tasks are relatively simple so that there is not much room for improvement of correctness. The mean completion time of group 2 is 29.4% lower than that of group 1 with mean difference being 40.333. The two-tailed p-value is 0.013 and is significant at the 0.95 confidence level. Fig. 7 compares the average correctness and completion time for all tasks of two groups.

Table 3
Performance comparison.

Task	Group 1		Group 2	
	Correctness (%)	Time	Correctness (%)	Time
1	57	152	86	127
2	71	171	100	89
3	100	102	86	53
4	43	167	57	114
5	86	144	86	134
6	100	87	100	64

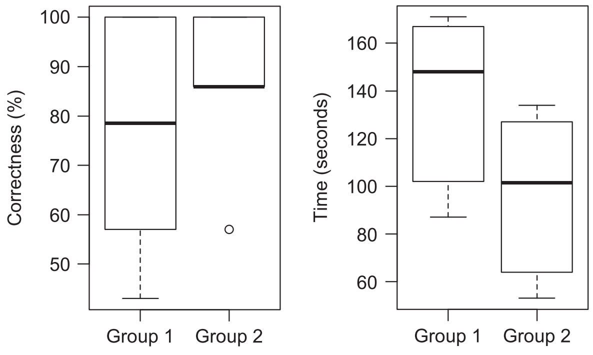


Fig. 7. Box plots for average correctness and completion time.

As expected, the visual approach outperformed the non-visual approach for those tasks that require a comprehensive understanding of the project, such as tasks 1, 2, and 4. Many subjects could get the answer right after viewing the animation. Subjects without the visual assist, however, often had to traverse the event script again and again to find out the required information.

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

For task 3, which requires a metric for developers’ activeness, the visualization did not provide information precise enough, leading to one subject in group 2 miscounted the second active developer as the most active one.

7. Discussion

7.1. *Possible improvements*

Though the visualization on Team Radar Mobile significantly improves the effectiveness and efficiency of the awareness system, the subjects were encouraged to provide suggestions for its further improvement. They have pointed out some possible improvements for the current system.

The analytical features of the current version of Team Radar are primitive. In some cases (e.g., task 4), the subjects still could not get the answer directly from the visualization. Some subjects suggested it would be simpler to analyze the information automatically and show the results 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 afterimages is limited. Although users can configure how much past information to be shown by adjusting the duration of afterimages, keeping the afterimages and light trails for too long may pollute the screen. For example, in task 5, the subjects were able to find out the most recent files Mike has edited from the light trail of Mike's icon. But the light trail only shows the recent several (20 by default) seconds’ history. The viewer still need to review a large portion of the animation to get the complete history of Mike's recent activities.

Although using the same tree presentation on all the platforms ensures visual consistency, and the layout algorithm with boundary constraint and distortion improves screen utilization, it is still advisable to develop a new visualization of tree structure dedicated for mobile devices that makes the best use of small screens. We have found node-link graph hard to layout on limited screen space. Alternative approaches under consideration include techniques similar to RELT [24] and TreeMap [46].

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

7.2. Implications

Our work has implications for both theoretical and practical work on awareness. With the proliferation of distributed software development, many new challenges regarding awareness support have emerged. In a distributed team, members' awareness need not only changes with their roles, but also changes with place and time. Current research on awareness has not emphasized the continuous changing aspect of awareness need. The concept of CA we have proposed expands the scope of awareness research regarding the need of continuous and integrated awareness support.

Traditional awareness tools only support desktop platforms, ignoring the awareness needs of mobile users and not providing enough support of ad-hoc collaboration. To meet the various awareness needs of different roles, and to create more changes for opportunistic collaboration, we have proposed to implement CA through the cooperation of multiple platforms. To our knowledge, Team Radar Mobile is the first awareness tool on mobile platforms in the software engineering community. Our work on mobile awareness also has implications for the design of awareness tools on mobile platforms. The design requirements we have identified, the solutions to the requirements, and the lessons we learned from the experiment could be generalized for the design of other similar tools. The evaluation also provides evidence for the potentials of the visual mobile approach to awareness.

8. Conclusions

This paper has presented a research on continuous awareness (CA), including its concept, design, implementation, and evaluation. Continuous coordination, a new collaboration paradigm, aims to combine the strengths of informal awareness-based approaches with those of formal process-based approaches. CA is an extension of this paradigm on awareness support, addressing the need of continuous and integrated awareness information.

We have implemented a CA system by extending the visualization on desktop platforms to mobile platforms. The visual mobile approach complements the insufficiency of desktop platforms regarding CA support. We have identified design requirements for mobile awareness visualization, and have proposed several visual metaphors to aid its implementation. The efficacy of the visual mobile approach is validated with a controlled experiment.

Future work includes designing better graph visualization for small screens, implementing more analytical features, and a comprehensive evaluation of the entire Team Radar system.

References

[1] D.J. Herbsleb, D. Moitra, Global software development, Software IEEE 18 (2001) 16–20.

[2] I. Steinmacher, A.P. Chaves, M.A. Gerosa, Awareness support in global software development: a systematic review based on the 3C collaboration model, in: Proceedings of the 16th International Conference on Collaboration and Technology, Springer-Verlag, 2010, pp. 185–201.

[3] P. Dourish, V. Bellotti, Awareness and coordination in shared workspaces, in: Proceedings of the 1992 ACM Conference on Computer-supported Coop. Work, ACM, 1992, pp. 107–114.

[4] C. Gutwin, S. Greenberg, Workspace awareness for groupware, in: Proceedings of the Conference on Companion on Human Factors in Computing Systems: Common Ground, ACM, 1996, pp. 208–209.

[5] M. Lanza, L. Hattori, A. Guzzi, Supporting collaboration awareness with real-time visualization of development activity, in: Proceedings of the 14th IEEE European Conference on Software Maintenance and ReEng., IEEE Computer Society, 2010, pp. 207–216.

[6] I. Omoronyia, J. Ferguson, M. Roper, M. Wood, a review of awareness in distributed collaborative software engineering, Software: Practice and Experience (2010) 1107–1133.

[7] D. Redmiles, A. van der Hoek, B. Al-Ani, T. Hildebrand, S. Quirk, A. Sarma, R. Silva Filho, C. de Souza, E. Trainer, Continuous coordination: a new paradigm to support globally distributed software development projects, Wirtschaftsinformatik 49 (2007) S28–S38.

[8] R. Sindhgatta, B. Sengupta, S. Datta, Coping with distance: an empirical study of communication on the jazz platform, in: Proceeding of the 2011 ACM International Conference on Companion on Object Oriented Programming Systems Languages and Applications Companion, ACM, 2011, pp. 155–162.

[9] V. Bellotti, S. Bly, Walking away from the desktop computer: distributed collaboration and mobility in a product design team, in: Proceedings of the 1996 ACM Conference on Computer Supported Coop. Work, ACM, 1996, pp. 209–218.

[10] L.E. Holmquist, J. Falk, J. Wigström, Supporting group collaboration with inter-personal awareness devices, J. Personal Technologies 3 (1999) 13–21.

[11] C. Chen, K. Zhang, Team radar: visualizing team memories, in: Proceedings of the 6th International Conference on Evaluation of Novel Approaches to Software Engineering, 2011, pp. 114–120.

[12] A. Sarma, D. Redmiles, A. van der Hoek, Palantír: early detection of development conflicts arising from parallel code changes, IEEE Trans. Software Eng. 5 (2011) 1.

[13] R. Hegde, P. Dewan, Connecting programming environments to support ad-hoc collaboration, in: Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering, IEEE Computer Society, 2008, pp. 178–187.

[14] C. de Souza, S. Quirk, E. Trainer, D. F. Redmiles, Supporting collaborative software development through the visualization of socio-technical dependencies, in: Proceedings of the 2007 International ACM Conference on Supporting Group Work, ACM, 2007, pp. 147–156.

[15] F. Calefato, F. Lanubile, N. Sanitate, G. Santoro, Augmenting Social Awareness in a collaborative development environment, in: Proceedings of the 4th International Workshop on Social Software Engineering, ACM, 2011, pp. 39–42.

[16] A. Burak, T. Sharon, Analyzing usage of location based services, in: Proceedings of the CHI '03 Extended Abstracts on Human Factors in Computing Systems, 2003, pp. 970–971.

[17] A. Oulasvirta, M. Raento, S. Tiitta, ContextContacts: Re-designing smartphone's contact book to support mobile awareness and collaboration, in: Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices and Services, 2005, pp. 167–174.

[18] P. Ljungstrand, Context awareness and mobile phones, Personal Ubiquitous Comput. (2001) 58–61.

[19] J.C. Tang, N. Yankelovich, J. Begole, M. Van Kleek, F. Li, J. Bhalodia, ConNexus to awarenex: extending awareness to mobile users, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2001, pp. 221–228.

[20] C. Papadopoulos, Improving awareness in mobile CSCW, IEEE Trans. Mobile Computing 5 (10) (2006) 1331–1346.
[21] L. Chittaro, Visualizing information on mobile devices, Computer (2006) 40–45.
[22] S. Burigat, L. Chittaro, On the effectiveness of overview+detail visualization on mobile devices, Personal and Ubiquitous Computing (2011) 1–15.
[23] I. Herman, I. Herman, G. Melancon, M.S. Marshall, G. Melancon, S.M. Marshall, Graph visualization and navigation in information visualization: a survey, IEEE Trans. Visualization and Computer Graphics 6 (2000) 24–43.
[24] J. Hao, K. Zhang, M.L. Huang, RELT: Visualizing trees on mobile devices, in: Proceedings of the 9th International Conference on Advances in Visual Information Systems, Springer-Verlag, 2007, pp. 344–357.
[25] A. Chhetri, K. Zhang, Modified RELT for display and navigation of large hierarchy on handheld touch-screen devices, in: Proceedings of the 11th International Conference on Computer and Information Science, IEEE Computer Society, 2012, pp. 147–152.
[26] H. Shin, G. Park, J. Han, Tablorer - An interactive tree visualization system for tablet PCs, Computer Graphics Forum 30 (2011) 1131–1140.
[27] P.A. Eades, A Heuristic for Graph Drawing 42 (1984) 149–160.
[28] J. Sillito, G. Murphy, K. De Volder, Asking and answering questions during a programming change task, IEEE Trans. Software Engineering 34 (2008) 434–451.
[29] R. Holmes, R.J. Walker, Promoting Developer-specific Awareness, in: Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering, ACM, 2008, pp. 61–64.
[30] R. Holmes, J.R. Walker, Customized awareness: recommending relevant external change events, in: 2010 ACM/IEEE 32nd International Conference on Software Engineering, ACM, 2010, pp. 465–474.
[31] F. Calefato, D. Gendarmi, F. Lanubile, Embedding Social Networking Information into Jazz to Foster Group Awareness within Distributed Teams, in: Proceedings of the 2nd International Workshop on Social Software Engineering and Applications, ACM, 2009, pp. 23–28.
[32] J.D. Herbsleb, A. Mockus, T.A. Finholt, R.E. Grinter, Distance, Dependencies, and Delay in a Global Collaboration, in: Proceedings of the 2000 ACM Conference on Computer Supported Coop. Work, ACM, 2000, pp. 319–328.
[33] S. Teasley, L. Covi, S.M. Krishnan, S.J. Olson, How does radical collocation help a team succeed?, in: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, ACM, 2000, pp. 339–346.
[34] J.A. Ko, R. DeLine, G. Venolia, Information needs in collocated software development teams, in: Proceedings of the 29th International Conference Software Engineering, IEEE Computer Society, 2007, pp. 344–353.
[35] T. Fritz, G.C. Murphy, Using information fragments to answer the questions developers ask, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering—Vol. 1, ACM, 2010, pp. 175–184.
[36] A.H. Caudwell, Gource: visualizing software version control history, in: Proceedings of the ACM Internationl Conference on Companion Object Oriented Programming Systems Languages and Applications Companion, 2010, pp. 73–74.
[37] J.M. Carroll, J.C. Thomas, Metaphor and the cognitive representation of computing systems, IEEE Trans. Systems, Man and Cybernetics (1982) 107–116.
[38] P. Thompson, D. Burr, Visual aftereffects, Current Biology 5 (2009) 11–14.
[39] S. Heywood, J. Churcher, Eye movements and the afterimage–I. Tracking the afterimage, Vision Research (1971) 1163–1168.
[40] M.T. Fennell, R.P. Wishner, Battlefield awareness via synergistic SAR and MTI Exploitation, IEEE Aerospace and Electronic Systems Magazine (1998) 39–43.
[41] R. Atkinson, R. Shiffrin, K.W. Spence, J.T. Spence, Human memory: a proposed system and its control processes, in: Psychology of Learning and Motivation, Academic Press, 1968, pp. 89–195.
[42] R. Hadany, D. Harel, A multi-scale algorithm for drawing graphs nicely, in: Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science, 1999, pp. 262–277.
[43] M. Kersten, G.C. Murphy, Using Task Context to Improve Programmer Productivity, in: Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, 2006, pp. 1–11.
[44] T.M.J. Fruchterman, E.M. Reingold, Graph drawing by force-directed placement, Software: Practice and Experience (1991) 1129–1164.
[45] M. Fowler, K. Beck, J. Brant, W. Opdyke, D. Roberts, Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional, 1999.
[46] B. Johnson, B. Shneiderman, Tree-Maps: A Space-filling Approach to the Visualization of Hierarchical Information Structures, in: Proceedings of the 2nd Conference on Visualization, IEEE Computer Society Press, 1991, pp. 284–291.