International Journal of Software Engineering
and Knowledge Engineering
Vol. 25, No. 2 (2015) 253–275

DOI: 10.1142/S0218194015400094

World Scientific
www.worldscientific.com

Visualized Awareness Support for Collaborative Software Development on Mobile Devices

Meng-Yao Chen[*,‡], Cong Chen[†,§], Shu-Qing Liu[*,¶] and Kang Zhang[†,‖]

[*]*School of Computer Software, Tianjin University, 92 Weijin Road Nankai District Tianjin, 300072, China*

[†]*Department of Computer Science, The University of Texas at Dallas Richardson, TX 75080, USA*

[‡]*chenmy@tju.edu.cn*
[§]*congchen@utdallas.edu*
[¶]*liusqpsy2008@126.com*
[‖]*kzhang@utdallas.edu*

To foster innovation and competition, an increasing number of software teams are becoming distributed. Such a distribution makes continuous collaboration and continuous awareness support a necessity and also a great challenge. Traditional desktop-based approaches are insufficient for the requirements of continuous awareness. In software development practice, an awareness tool on mobile devices is also desirable for team members to obtain the awareness information continuously. This paper addresses how to effectively present collaborative development activities using aesthetic visualization on mobile screens. Our approach supports multiple views suitable for software developers as well as team leaders. A baseline usability experiment and an eye-tracking experiment have evaluated the effectiveness and usability of the visualization method.

Keywords: Awareness; visualization; mobile; collaborative development.

1. Introduction

Software development is a typical group activity. Such teamwork requires intense collaboration among team members as well as other outside stakeholders [4]. Whether a team can keep abreast of team members' activities becomes an important factor to effective team collaboration [8]. To avoid unnecessary conflicts, team members are supposed to understand other members' activities. Project managers should acquire the information about the project progress and the members' job schedules in time. Such an insight into collaborators' activities is often called *awareness*. In collocated teams, members can communicate timely. The awareness of members' activities is usually obtained through direct interactions, such as

‖Corresponding author.

monitoring each other, informal communication, and aid of others. To foster innovation and competition, an increasing number of 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 [22]. The loss of awareness not only harms team effectiveness and mutual trust [3], but also affects contributors' willingness and enthusiasm of work [13]. It is particularly important to access development information anywhere and anytime, which is called *continuous awareness* [26].

Continuous awareness inherently requires its implementation to support multiple platforms [4]. Current awareness approaches are mostly 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. Inspired by the concept of mobile CSCW (computer-supported cooperative work) [23], we call the awareness support on desktop platforms *desktop awareness*, and its support on mobile platforms *mobile awareness*. In globally distributed teams, awareness needs often change with people's role, time, and place. Users' mobility often disables the effectiveness of desktop awareness tools [1]. Mobile platforms have several advantages over desktop platforms, such as portability and flexibility. A mobile awareness tool would complement the insufficiency of continuous awareness support on desktop platforms. Although mobile awareness faces several challenges due to the limitations of memory capacity, display size, and computational power, with the advances of the mobile technology, the performance of mobile devices has improved significantly, which makes the research on mobile awareness more favorable.

Our aim is to utilize the strengths of mobile platforms for an all-around, continuous awareness support for different roles in distributed software teams.

Our original work, Team Radar Mobile [4], has explored techniques for implementing continuous awareness using consistent visualization on multiple platforms. Although enhanced with a layout algorithm that considers screen boundary and limited performance of mobile devices, its design does not utilize the small screen space optimally. In our previous paper [6], we have improved the original visualization by using an adapted treemap [16] layout and a visual design in Mondrian style [30]. The improved visualization makes full use of precious screen estate and blends in an aesthetic view to maximally engage the user. A task-based experiment has been conducted to preliminarily evaluate the effectiveness and efficiency of our design [6].

The main difference of this paper is that we have introduced an objective evaluation of the usability of the approach using an eye-tracking experiment. We used an eye-tracker to study users' eye gaze and attention when viewing the visualization on the screen. Combined with the task-based evaluation, the analysis on the eye-tracking information has enabled us to discover potential design problems of the user-interface and visualization features.

The rest of the paper is organized as the following. We first lay out the background of our work in Sec. 2, followed by an introduction of the mobile information visualization and the design principles in Secs. 3 and 4. Then the details of the enhanced Team Radar Mobile system are discussed in Sec. 5. Section 6 presents a preliminary evaluation and results of the approach. The eye-tracking experiment and results are given in Sec. 7. Section 8 discusses the limitations of current work. Finally, Section 9 concludes 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 collaborative and mobile visualization.

Awareness is now considered an important approach to many collaboration problems in software development [22]. There are a number of approaches in the community to improving awareness, such as workspace awareness (e.g. Palantir [28] and Gasper [14]) and social awareness (e.g. Ariadne [27] and [3]).

The value of mobile platforms for awareness support has already been demonstrated in the CSCW literature. The ConNexus and Awarenex projects [29] introduce design principles and experiences learned in extending awareness services from desktop to mobile platforms. Papadopoulos [23] has discussed key requirements for awareness in mobile CSCW.

Visualization is often used as a tool for presenting information and conducting analysis. Sense.us [10] is a web site supporting asynchronous collaboration across a variety of visualization types. Mahyar and Tory [21] have created a visual analytical technique, linked common work, to facilitate synchronous collaborative sensemaking.

As the computation power of mobile devices grows, research on mobile visualization has become popular in recent years [12]. Mobile Information Visualization aims to develop more efficient and satisfying user interfaces for searching and exploring large information space on small screens [2, 24, 25]. Chittaro [7] enumerate many restrictions that a researcher must consider when developing visualization applications on mobile devices, such as the limited screen size, onboard hardware, different width/height aspect ratio and performance. Chittaro also suggests 6 steps in designing mobile visualization: mapping, selection, presentation, interactivity, human factors and evaluation, which have inspired the design of our visualization.

The present work is based on our previous work on Team Radar Mobile [6, 5] and has enhanced its visualization considering the specific requirements of mobile information visualization and the collaborative information in software development. A more objective, eye-tracking experiment has further examined the usability of the approach.

3. Collaborative Information in Software Development

Awareness information in collaborative software development often includes the following elements:

- Developers who contribute to the information,
- Time when a modification occurs,
- Files with changes and their paths,
- Change mode, such as add, delete, and conflict.

As a motivating example, a software developer Bob added a new file called *main.java* at 15:20 on April 20, 2014 in the directory of *Android/test/src/*. According to the characteristics of the information, we can display the information based on (1) project structure and (2) developers' activities. This paper uses a project-based approach that visualizes developers' activities.

The project-based approach mainly focuses on a continuous acquisition of the project's changes and progress. A project structure is essentially a file system, typically hierarchical by nature. The problem can therefore be formulated as hierarchical information visualization. The visualization of hierarchical information is a hot topic in the information visualization community. Hierarchical visualization can assist users in understanding the information and relationship among various information elements, and gaining an overview of the structure. Hierarchy is typically represented in a tree structure, similar to a tree expanded from its root with many branches. Two types of information can be stored in a hierarchical structure. One is the structural information, and the other is the content. Structural information refers to the relationships among the tree nodes, such as parents, children, siblings and so on. In our case, the structural information is essentially the project's file structure, and the content includes various software development activities.

Hierarchical information visualization mainly includes two methods: node-link and space-filling. The node-link method is based on the traditional tree diagram as drawn in graph theory. H-tree, Radial view and Hyperbolic-tree [19] are the representatives of the node-link method. Node-link trees explicitly present the relationships between tree nodes, but do not utilize screen space efficiently. Space-filling visualization maps hierarchical information to a rectangular area in a space-filling manner [16]. Treemap [15] is the most notable example of this method, which utilizes all of the screen space and also facilitates easy labeling.

Due to the limited screen space, the node-link tree is not suitable for mobile devices. We therefore adopt the space-filling approach to visualize the hierarchical collaborative information in a compact fashion.

4. Design Principles

Our work aims at assisting project managers and software developers to understand the project information in real time while on the move. When designing the visualization, we focus on helping users retrieve information timely and accurately.

Awareness in software development mainly concerns the changes made by developers in a project. The key to our design is the efficient use of mobile devices to obtain such information. Two primary concerns of the visualization are: (1) What types of awareness information are needed in the process of software development; (2) How to design a mobile information visualization technique for the information. As introduced in Sec. 3, the project-based approach focuses on a continuous acquisition of the project's changes and progress, while the developer-based method focuses on a continuous acquisition of developers' performance.

It is desirable for all the information to be displayed in the screen simultaneously to achieve a general awareness of the project. In our case, we could display the changes of all the classes of a project using color coding at the pixel level, i.e. a row of pixels represent a class or its method as discussed later.

Meanwhile the characteristics of mobile platforms, such as smaller screen, limited computation power and users' intermittent focus (contrasted to more continuous focus on desktop computers) [29], pose special requirements for mobile application design.

We believe that artistic design is a key factor of successful information visualization. In order to create a virtual presence environment that promotes users' cognition and interests, as well as to increase information density, Team Radar Mobile combines the information visualization with abstract art to maximally engage the user. It not only clearly shows the information, but also allows the users to enjoy it in an aesthetical way, following the principles of aesthetic computing [11].

5. Team Radar Mobile Enhancement

Figure 1 illustrates the architecture of the Team Radar system. TeamRadar Desktop monitors and captures events of interest in local workspaces, and sends them to TeamRadar Server, which broadcasts them to other registered clients. Team Radar Mobile [4] analyzes the received information, stores and presents it with three methods of visualization.

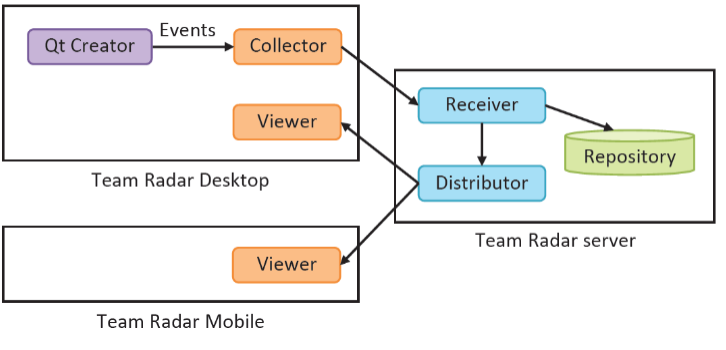


Fig. 1. The architecture and key components of the overall Team Radar system.

Table 1. Awareness information supported by Team Radar Mobile.

Information	Benefit	Benefited party
Project development progress	Monitor the progress of the project	Manager
Developers' activities	Understanding the workload of developers Assisting task arranging	Manager
Other developers' activities	Understanding the workload of colleagues Assisting information exchanging	Developer
Project Structure	Understanding the organization structure of the project	Manager & Developer
Conflicted file	Reducing merge conflict and repetitive work	Developer

People of different roles in a software team have different information needs and communication patterns. Project managers typically have wider range of information need, as they concern about project-level issues, such as progress, budget, schedule, risks, etc. 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 about technical issues, such as requirements, design decisions, code changes, bug status, etc. To meet the aforementioned challenges, Team Radar Mobile supports the following information elements, as explained in Table 1. Team Radar Mobile currently works on the Android mobile operating systems, which supports touch screen and is popular in the mobile market.

The visualization on Team Radar Mobile provides three views, presenting the project from different perspectives: Overview, Detail View and Developer View, as detailed in the remaining part of this section.

5.1. *Overview*

The overview is to give managers or developers a general coverage of the project overall activities. Figure 2 presents the visualization of the overview. This visualization imitates Piet Mondrian's style of abstract paintings, making up a white or gray background and is painted as a grid of vertical and horizontal black lines, with rectangle blocks filled with three primary colors [30].

In our visualization, each rectangle grid represents a class in the project. In Fig. 2(a), each modified method in a class is shown as a color bar inside its class rectangle. Figure 2(b) shows only the method being modified the most in a class, making the visualization closer to Mondrian's style and allowing users to click and view details of the modification. The layout is generated automatically using a square-merging algorithm (Fig. 3). The side of a grid indicates the number of the methods in the class. First, Team Radar Mobile generates equal sized squares. The number of squares is always greater than the number of the classes in the project to be visualized. Next, it randomly selects some of the squares and merges them to make the number of the grids equal to that of the classes. Every method in a class may involve several types of changes, including modify, edit, add, delete, and conflict. Each of the changes is color coded (e.g. modify-white, edit-gray, conflict-red, add-blue, delete-yellow). The grid is

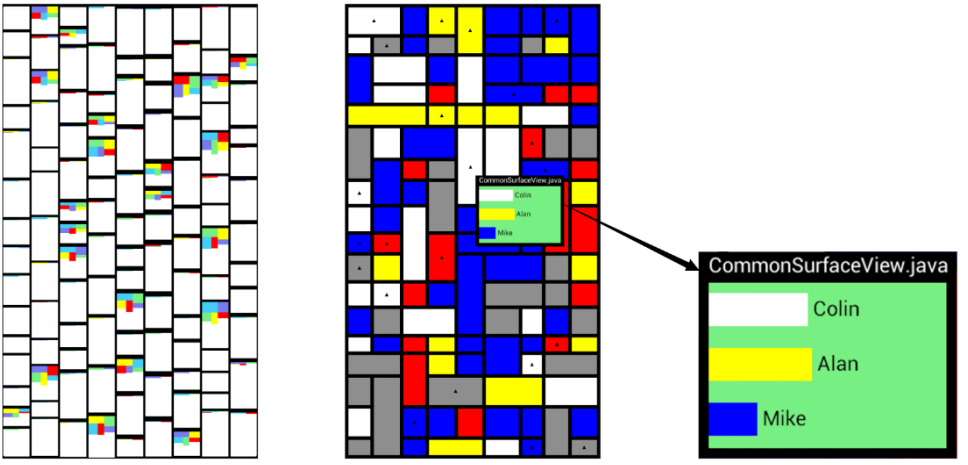


Fig. 2. Overview screens of Team Radar Mobile.

filled with one of these colors to indicate the type of change that occurs the most. For example, *Main.java* has 50 methods, of which 30 are recently added, therefore, the grid representing this class is filled with blue.

Each grid embodies the amount of all the methods in a class, rather than the methods that have changed. In order to clearly recognize the project's main classes that the developers worked on, we use a tiny black triangle inside the grid to mark the class that has a higher proportion of changed methods.

The overview also supports interactive operations. By clicking a grid, the user can obtain an information box about the corresponding class. Figures 2(b) and 2(c) show the information box in the overview and its enlarged view respectively. In the information box, the class's name is displayed on the top, and all the developers' contributions to this class are displayed using a bar chart. By a simple click, managers or developers can easily view the developers' job schedules for the class.

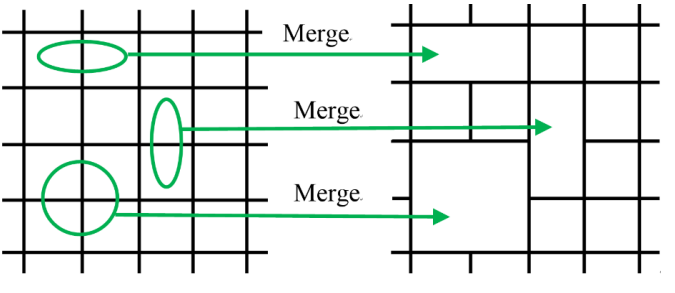


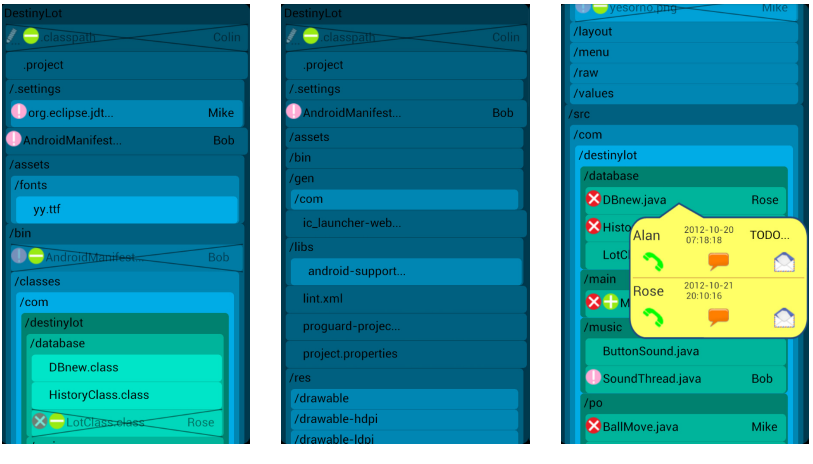
Fig. 3. The generation process of the grid mimicking Mondrian's style (from left to right).

The overview offers a general understanding of the project. By simple interactive operations, it can provide visual information on developers' contributions to a class. This visualization divides the screen according to the classes' sizes, making the best use of the screen space. Another innovation is the imitation of the Mondrian style, which creates a virtual environment that promotes the user's perception and engagement.

5.2. *Detail view*

It is critical to make full use of the screen space for displaying information on mobile devices. Considering the characteristics of the hierarchical information and the small screen, we adapt the treemap to display software collaborative information, as shown in Fig. 4(a).

The treemap visualization technique makes efficient use of available display space, mapping hierarchies onto a rectangular region in a space-filling manner [15]. The treemap approach to visualizing hierarchies enables meaningful drawing of hierarchical information on limited screen space of mobile devices. However, the original treemap cannot meet the requirements of mobile devices. For example, some leaf nodes could be too small to label. The adapted treemap in the detail view in our approach divides the screen only in the vertical direction and can be extended by swipe gestures. The collaborative information consists of the name of the developer, the name of the file, time and activity type. These types of information in the detail view are displayed by presenting the directory structure of a project in the adapted form of treemap. First, a message received from the Team Radar Sever is stored as a tree node according to the project's hierarchical structure. Non-leaf nodes represent



(a) (b) (c)

Fig. 4. Screenshots of the detail view.

directories and leaf nodes denote files. The collaborative information, such as the developer's name, type and time, is included in the file node (leaf node).

Since the screen is divided only in the vertical direction, the drawing algorithm is simplified from the original treemap algorithm. The parent node containing the directory's name is drawn first, and then its children will be drawn. If the child is a leaf node, the collaborative information will be presented in the node. The following is the pseudo code of the adapted treemap algorithm.

```
draw (TreeNode t){
  if (t is leaf)
    draw_leafNode(t);
  else
    draw_nonLeafNode(t);
  for (TreeNode n : t.children())
  {
    draw(n);
  }
}
```

The time complexity of the algorithm is $O(|V|)$, $|V|$ being the number of vertices, which satisfies the complexity constraints of mobile devices.

The change mode information is denoted by different icons, as shown in Fig. 5. Icon (a) represents a new file, and icon (b) indicates that the file has been deleted. Modified files are marked with icon (c), and the file with conflict is marked with icon (d) Icon (e) expresses that the developer is in the process of editing the file. Due to the limited screen space, too complicated icons are difficult to distinguish and thus are avoided. These icons are designed based on the principle of being explicit and easy to understand, while providing users easy access to the information. In addition, the color of each icon indicates the severity of the change. For example, the conflict icon is colored red, which indicates that the information needs more attention.

The collaborative information in a file node only contains the developer who modified the file recently. In practice, there may usually be more than one person worked on this file in a recent period. More collaborative information about a file can be accessed by tapping on the corresponding node, as shown in Fig. 4(c). The information in the pop-up window includes the developer's name, the operation time and the tags the developer marked in the file. The instant messaging feature of a

(a) (b) (c) (d) (e)

Fig. 5. The icons in the visualization denoting different change and working modes.

mobile phone is brought into full play. Users can send instant messages through the application, making communication convenient.

The scalability and readability of a visualization technique is often affected by excessive information. If all the information is presented simultaneously, users can be overwhelmed or even distracted. Windows file browser inspires us and guides us in designing interactive functions. When a user taps a directory node, the detail view will expand the node and display all the nodes under the directory node or keep the nodes folded. In addition, pinching two fingers expands or folds the nodes at the same level, as shown in Fig. 4(b).

5.3. *Developer view*

The developer view allows a project manager to acquire a specific developer's development activities. This visualization presents the collaborative information organized by developers, as shown in Fig. 6. All the developers' names are displayed on the screen vertically. Detailed development activities about a developer are displayed under the name, including the files contributed by the developer, modified time, TODO tags inserted and change type. The types of information and the visual symbols in this view are similar to the overview, except that the information is classified by developers.

Basic tapping operations are permitted to hide or show the information under a specific developer, inspired by the metaphor of pushing and pulling the drawer. Hiding irrelevant developers' information cleans up the screen for the developer under inspection.

Fig. 6. A screenshot of developer view.

Fig. 7. The top level menu of the visualization.

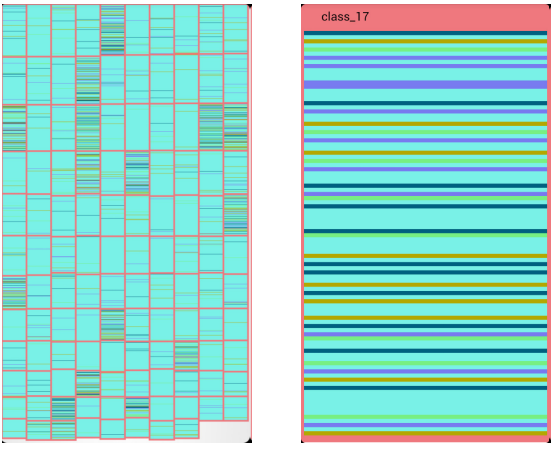
In addition, the developer view supports instant messaging. When a project manager or developer finds a problem in collaboration, he/she can inform involved developers immediately. Upon tapping the developer's name label, an instant messaging toolbox shows up, allowing the user to make a phone call, send a message or email to the developer. Managers could inspect developers' workloads or problems by using this view. Developers can also obtain other developers' activities to plan and coordinate their own activities.

Users can switch between the three views through the top level menu shown in Fig. 7.

5.4. *Support for big software data*

To maximize scalability, we have designed a pixel-based visualization view for Team Radar Mobile, such that the total number of methods in a software project is bounded by the available pixels on the mobile screen. For example, Fig. 8(a) shows a big software project that includes over 10,000 methods distributed in various classes divided by red grid lines. The methods are color-coded to reflect their activity status. Each individual class could be enlarged to occupy the entire screen for class-level view (Fig. 8(b)) by touching on the corresponding red square in Fig. 8(a). Further zooming in by clicking on a colored line in Fig. 8(b) will bring the user to the developer view discussed in Sec. 5.3. This screenshot is from our implementation on an Android phone with the screen resolution of 1280*720 pixels.

The basic idea is to generate the scalable view based on the total number of methods within the given project. The number of screen pixels representing an individual method depends on the total number of methods and classes of the project. First, a minimally required number of columns is computed based on the given screen width. A greedy algorithm is then used to divide the number of classes to fill the columns, by ordering the classes based on their sizes (i.e. number of methods). The pseudo code of the algorithm is outlined below.



(a) (b)

Fig. 8. Pixel-based view of big software data (10,000+ methods).

```
ScalableView
{
  if (count(methods) < vertical resolution)
    for (# of classes)
      drawClassBorder;
      for (# of methods)
        drawMethodBasedOnStatus;
  else
    compute # columns needed to fill screen width;
    divide classes for columns using Greedy algorithm
    for (# of columns)
      drawClassBorder;
      for (# of methods)
        drawMethodBasedOnStatus;
}
```

The above is not an optimal algorithm but fast enough to generate a satisfactory layout. It may result in a small empty space at the bottom-right cover, as shown in Fig. 8(a). Improvements to generate more optimal and aesthetic layouts will be our future work.

6. Baseline Evaluation

The effectiveness and usability of the Team Radar Mobile approach to continuous awareness are evaluated with a small group of users. We first performed a baseline evaluation, for which the test subjects were asked to finish several awareness-related

tasks and answer corresponding questions. This section reports the baseline evaluation and results, followed the report of our further findings using the eye-tracking technology in Sec. 7.

6.1. *Experiment setup*

To test the effectiveness and feasibility of Team Radar Mobile for presenting software development information, we measure user performance on a series of awareness-related tasks. The tasks are designed to cover all the features of Team Radar Mobile (see Table 1 for details). The awareness events occurred in an Android application project, and developers' names were made anonymous. In this experiment, the participating subjects reviewed these events and performed several tasks based on what they had reviewed. Table 2 lists the tasks.

The experiment project includes 437 files (including directories) and 118 change events in total. The tasks are designed based on this project's change events. These events are displayed in all the three views, which can be switched through the top-level menu. The subjects used the three visualization views synchronously to perform the tasks. They performed the experiment using the same mobile phone with a screen resolution of 1280*720 pixels in a quiet laboratory environment.

Table 2. Experimental tasks.

1. Find the change type of the file src/com/destinylot/main/MainActivity.java (added, deleted, modified, conflict, edit).
2. Find the conflicted file in the src/com/destinylot/po/directory, and who have participated in the change of this file.
3. When did Alan modify the file BallMoveNormal.java? What are the important icons?
4. Find out the most conflicted developer.
5. Find the developers who participated in this project recently (within 2 weeks).
6. Find the most conflicted file, and who have made changes to this file.
7. Assume you are developer Rose who has a conflict with another developer in the file src/com/destinylot/database/DBnew.java, find out the developer and send a message to him or her.
8. Assume you are a project manager, identify what phase the project is most likely at (UI, database, and back end).

6.2. *Process*

Eleven participants were recruited voluntarily as experimental subjects. The average age of the subjects is 24. The subjects are all graduate students from the School of Software Engineering, Tianjin University in China. They are equipped with software engineering knowledge and familiar with the collaborative software development process.

First, we gave the subjects a brief introduction to the system and then spent 10–15 minutes training them on its usage. The training was divided into two phases. In the first phase, the subjects were free to use the application to get familiar with the

interface, features, and different visualization views. In the second phase, we answered the questions the subjects encountered to ensure the experimental environment is functional and the subjects were able to complete the tasks. Any technical problems arose in the training were solved before the experiment started. Finally, we processed with the experiment and assessed the subjects' performance.

6.3. *Results*

The performances of the subjects on these tasks are measured by average correctness and completion time (in seconds) of their answers to the questions associated with the tasks. The result is shown in Table 3.

Table 3. Test result.

Task	Correctness	Time	SD of Time	Task	Correctness	Time	SD of Time
1	100%	32	7.6	5	100%	29	5.2
2	100%	48	6.7	6	82%	21	2.4
3	100%	56	9.5	7	100%	52	6.8
4	82%	182	29.4	8	55%	92	18.3

As we had expected, the visual approach generated high accuracy. The tasks (e.g. Tasks 1, 2, and 7) that can be answered using the detail view are all done correctly. All the subjects could get the answer directly from the treemap within a short period of time.

Results from the tasks that rely on the developer view (e.g. Tasks 3, 4, and 5) are very different. Task 3 has a perfect result in terms of correctness and time, but Task 4, which needs the subjects to count, has a poor result in completion time.

Task 6, which could be performed in the overview, has a very short completion time. Its correctness is quite satisfactory but not perfect. Task 8 requires a comprehensive understanding of the project. Team Radar Mobile did not produce an excellent result for this task, since the subjects did not participate in the project development.

7. Eye-Tracking Experiment

In the above baseline evaluation, the task completion rate and accuracy are used to assess effectiveness, and the operation time is used to evaluate efficiency. These evaluation parameters may not reflect the user' true behavior accurately. Eye tracking technology, however, has long been used in usability research [20]. It can reveal the locations that the user's are interested in on the screen, and also show the change of the user's attention. The eye tracking data could assist us in interpreting the above baseline evaluation results and further evaluating the effectiveness of the visualization design.

Eye-tracking is the process of tracking where a person is looking at. With the help of an eye tracker, the user' pupil is detected, and detailed tracking data on the user'

visual attention is obtained. Eye-tracking technology has been widely used in fields ranging from psychology, reading research, to neuroscience research. In usability studies and market research, eye-tracking provides information to reveal users' and consumers' experiences and behaviors. Two main benefits of eye-tracking technology are in assisting to improve user interface design and understand human behavior. With the improvement of eye-tracking technology, an increasing number of researchers begin to utilize the eye-tracking technology for mobile phone designs [9]. Few researchers, however, use the eye-tracking technology to assess visualization methods. In addition, most existing experiments on mobile phones present a simulated mobile interface on a desktop eye tracker instead of being in a true mobile environment. The results obtained this way may be unreliable. We used a Tobii X120 Eye Tracker, which is specifically designed for mobile devices, to ensure that the experiment is accurate and results reliable.

7.1. *Eye tracker and its tracking data*

The Tobii X120 Eye Tracker used in this experiment can measure how people view real-word flat surfaces or screen such as physical objects, projections and video screens. It is specifically designed to test mobile devices, providing a distraction-free test environment, in which a test subject can naturally interact with the mobile device. The eye tracker's data rate is 60 or 120 Hz and the average time to tracking recovery is 100 ms. The high accuracy and precision of the tracking technology ensures that the research results are reliable.

Our experiment uses four methods to present eye tracking data: live-observation and video, gaze plot, hot spot, and area of interest (AOI). Live-observation allows us to view the test subject's eye movement in real time and the video provides a dynamic replay. The time and sequence of eye fixations can be visualized in the gaze plot. The hot spot map integrates a large amount of eye movement data and display it on the screen. The AOI analysis allows us to define focused areas of visualization and user interface. The obtained tracking results can be cross-checked and compared with the baseline evaluation results.

7.2. *Experiment setup*

For this experiment, we recruited eleven participants as experimental subjects. They are also graduate students from the School of Software Engineering, Tianjin University in China and are familiar with the general concept and process of collaborative software development.

To support cross-checking and comparison, we choose three of the tasks used in the baseline evaluation, i.e. Tasks 2, 3 and 6, for this eye-tracking experiment, because all the three tasks involve the use of every visualization view.

Each subject starts the test with a short calibration phase to ensure that the eye tracker can accurately extract his/her eye movement data. Unfortunately, the eye

tracker does not work with all kinds of human eyes, for example, wearing eyeglasses and eyes with small pupils would reduce the tracking precision. In the calibration process, three of the subjects could not participate in the test since their pupils were not recognized by the eye tracker.

We then gave the subjects a brief introduction to the system and then spent 10–15 minutes training them on its usage as the previous experiment. To minimize errors in tracking, we also spent 5 minutes providing instructions to the subjects for the eye tracking experiment, such as maintaining a good posture, not shaking head, and not blocking the camera. The experiment set up and environment are shown in Fig. 9. The person on the left controls and records the test, and the one on the right is a test subject. To avoid any external disturbance, the experiment was conducted in a quiet office environment.

To complete the tasks, the subjects should operate on the mobile phone and obtain the information for the tasks. To solve Task 2, a subject needs to click on the menu and chose the "Detail View". In Detail view, he/she should navigate to the "*src/com/destinylot/po/*" directory by clicking and sliding on the screen and then find out the files with conflict icon and check who have participated in the change of this file. For Task 3, the subject should first enter the "Developer view" and then find Developer Alan and click to expand her modifying records. Next, he/she should navigate to the record for *BallMoveNormal.java* and check the modified time and the change icons. For Task 6, the subject should choose the "Overview" and find the biggest red grid with a tiny black triangle, which indicates the most conflicted file. Finally, the subject should click the grid to see who have made changes to this file.

Fig. 9. Eye–tracking experiment set up and environment.

After the subject completed the tasks, the eye-tracking system generated a video that records the screen and the subject's fixations. As only the subject knows whether the eye tracking data is accurate, we played the video to the subject. During the playback, the subject would point out any inaccurate records. According to the feedback, we evaluated the accuracy of the experiment and removed the data with errors.

7.3. *Results and analysis*

We use Tobii Studio, an eye tracking software tool, to analysis the experimental data. All the subjects' data for three tasks have been recorded and analyzed. Due to the limited space, we present the data from one representative subject in this paper. Figure 10 presents the visualization of the eye tracking result for this subject. The eye tracking data of each task is presented in the gaze plot and hot spot map. The AOI (area of interest) denoted by a pinkish rectangle in Figs. 10 and 11 is an area that we expect the subjects to pay attention to. First, we analyze the gaze plot and hot spot map generated from one subject's eye movement data and compare the eye tracking data with the AOI.

The subject's scan path and fixation for Task 2 are shown in Fig. 10(a). The points mark the fixations in a numeric order and the lines are the saccadic sequences. To find the conflicted file in the "src/com/destinylot/po/" directory, the subject first scaned the screen to find out the root directory, then looked for the subdirectory until finding the target file. Apparently, many of the fixations are in the AOI. However there are not many eye fixations on the icons. We assume that the eye tracker for mobile devices is not very accurate or the icon is so recognizable that the subject gained the icon information with a quick glance. Given the accuracy of the eye tracker, the second hypothesis is more reasonable and demonstrates that our mobile visualization is indeed effective in rapid information recognition. For Task 3, the subject could first navigate to developer "Alan" easily and then found out the target record (see Fig. 10(c)). From the fixation and saccades data for Task 6 (see Fig. 10(e)) we can easily observe that the subject scanned the screen and then focused on the "most conflicted file" with ease. Figures 10(b), 10(d) and 10(f) show the corresponding hot spot maps that highlight the focused areas in red, yellow and green, with red indicating highest fixations and green the lightest. The red areas in the hot map are consistent with the AOI we have identified beforehand. The gaze plot and the hot spot map can indicate the availability of the elements in the visualization. They have indeed validated our visualization to be useful and effective in collaborative software development.

As each subject interacted with the mobile phone differently, we cannot superimpose the eye tracking data of all the subjects on the output visualization. We therefore analyzed the eye tracking data of the subjects separately. Next we will introduce the eye tracking result for Task 6 in detail.

Figure 11 shows eight subjects' scan paths when performing Task 6. As the detailed view is generated with certain randomness, it is different for each subject.

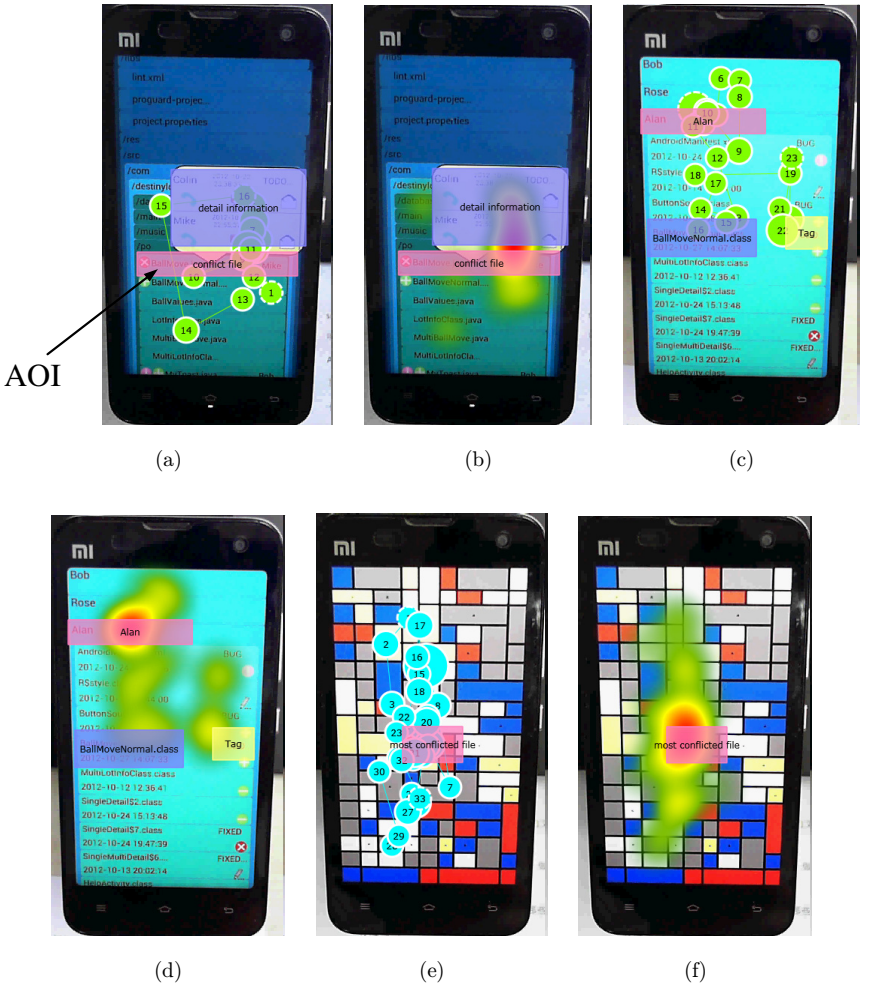


Fig. 10. The visualization of one subject's eye tracking data. (a), (c) and (e) correspond to Tasks 2, 3 and 6 with scan path and fixation visualizations. (b), (d) and (f) correspond to hot spot visualizations for the same tasks. Pink rectangles represent AOIs.

Although their scan paths are totally different, they all met the target position efficiently. When there were two files with similar information, the subject would hesitate between the two files (see Fig. 11(b)). According to video playback and subjects' feedback, we learn that most of the eye tracking data is accurate, and we compare the fixations with AOI.

There are many eye tracking metrics commonly reported in usability studies, e.g. gaze rate, number of fixations, gaze duration and so on [17]. We choose two metrics, the time to first fixation and the number of fixations on each AOI, which are most useful for our evaluation tasks. Figure 12 presents the statistical result of the two metrics for task 6.

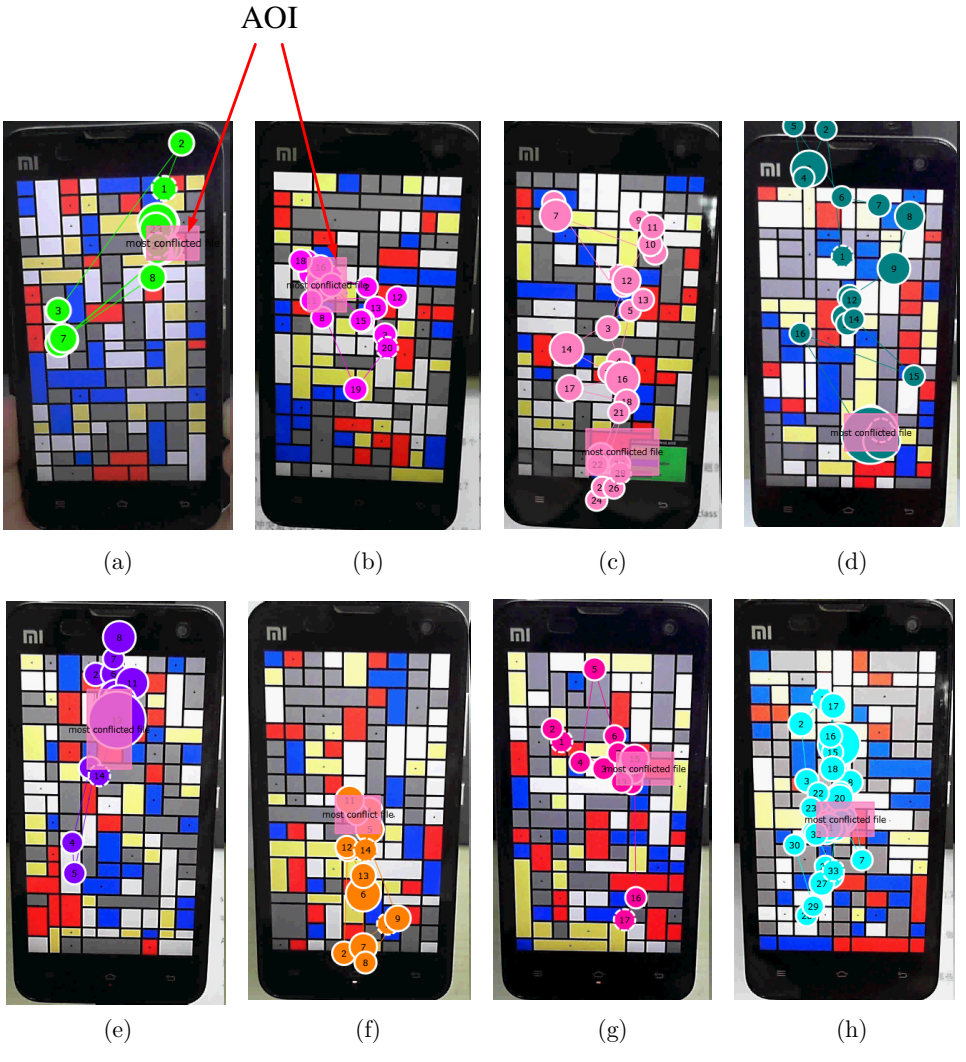


Fig. 11. Eight subjects' scan path and fixations in preforming task 6. Pink rectangles highlight AOIs.

The elapsed time to first fixation, which measures the time for the first fixation within an AOI, can be used to evaluate the efficiency of the visualization. Three subjects could fixate on the AOI in 1 second, but one subject spent 6.3 seconds. Although the time is uneven, all the subjects could locate the AOI in a trivial amount of time. The average time to first fixation is 1.92 seconds with a standard deviation of 2.13. The time to first fixation demonstrates that our visualization is efficient.

Fixation count is the number of times the subject fixates on an AOI and could imply the importance of that AOI [17]. In our task, the subjects should fixate on "the most conflicted file" (AOI) more frequently than other areas of the screen. Our

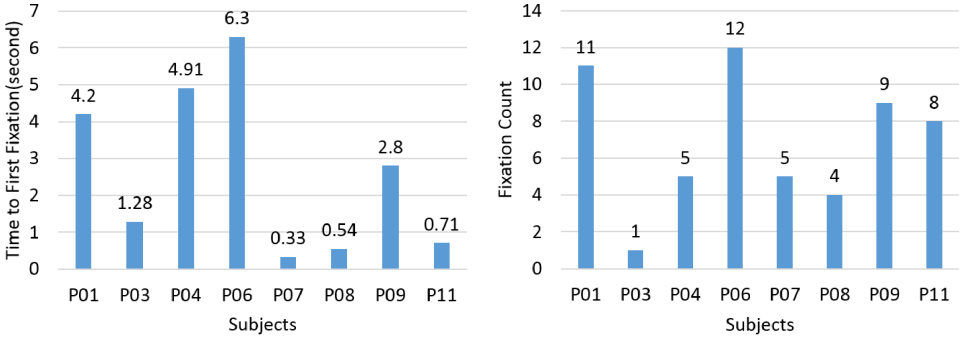


Fig. 12. (a) The time from the subjects begin the task to first fixated on the AOI. (b) The number of times the subjects fixated on the AOI.

experimental results indicate that, except subject P03 who had one fixation, all other subjects paid more attention to the AOI. The average fixation count is 3.13 with a standard deviation of 3.52. This result shows that our visualization has successfully attracted users' attention to the most important information. Thus, we consider our visualization to be highly effective and usable.

8. Discussions

The aforementioned evaluation results have shown that the enhanced Team Radar Mobile performed well in visualizing most of the collaborative development information, but the subjects pointed out various possible improvements to the current system.

The subjects could not switch well in the three views of the current implementation. They did not know which view to use to complete the tasks. For example, most subjects spent half of their time in finding an appropriate view to perform Task 2. Once the subjects understood the visualization well, they could finish the tasks immediately.

In the developer view, the layout of the grid for each individual developer is displayed randomly, which makes the tasks (e.g. Task 4) difficult to perform. The subjects could not compare the developers' information well and one of them suggested an improvement for our future work: the information for a particular developer should be organized by the change modes made by the developer and ordered by modification time.

There are two possible improvements for the overview. First, expressing the change modes using colors has caused difficulty for the subjects. They could not remember the color codes used to encode the information, except that red represents something serious. Second, the rectangles in a grid are laid out vertically and horizontally, while located randomly in the space. Some of the subjects felt difficult to

locate corresponding grids for some "most" questions (e.g. finding the most conflicted file). Most of them, though, said that the overview is helpful for understanding the general status of the project.

9. Conclusion

This paper has presented a visualization technique for presenting collaborative information in software development on mobile platforms. We have improved the continuous awareness system by enhancing the mobile platform application. The limitations in the mobile information visualization are considered in our visualization. The features of software development information have been discussed and displayed in three views. Due to the hierarchical nature of the collaborative information in software development, this paper presents a hierarchical visualization method, i.e. an adapted treemap. We also combine the visualization with art, which provides an aesthetic interface in the visualization and would potentially engage the user. The approach can assist project managers and developers, who may not otherwise have access to a desktop computer to maintain the awareness information and to improve the efficiency of software development. A task-based experiment and an eye-tracking experiment have confirmed that our visualization is effective in presenting the most important awareness information on small screens.

Our immediate future work includes the evaluation of the approach for its usability and effectiveness in monitoring real world collaborative software development. More innovative software visualization [32] techniques will be incorporated into our approach.

Acknowledgments

We would like to thank the students in the School of Computer Software at Tianjin University for their participation in the usability study. We are also grateful to the anonymous reviewers for their constructive comments that have helped us to improve the final presentation.

References

1. V. Bellotti and S. Bly, Walking away from the desktop computer: Distributed collaboration and mobility in a product design team, in *Proc. 1996 ACM Conf. Computer Supported Coop. Work*, 1996, pp. 209–218.
2. T. Buering, J. Gerken and H. Reiterer, User interaction with scatterplots on small screens — a comparative evaluation of geometric-semantic zoom and fisheye distortion, *IEEE Trans. Visualization and Computer Graphics*, 2006.
3. F. Calefato, F. Lanubile, N. Sanitate and G. Santoro, Augmenting social awareness in a collaborative development environment, in *Proc. 4th Int'l Workshop on Social Software Eng.* 2011, pp. 39–42.

4. C. Chen and K. Zhang, Continuous awareness: A visual mobile approach, in *Proc. 5th International Symposium on Visual Information Communication and Interaction*, 2012, pp. 69–76.
5. C. Chen and K. Zhang, Team radar: Visualizing team memories, in *Proc. 6th Int'l Conf. Evaluation of Novel Approaches to Software Eng.*, 2011, pp. 114–120.
6. M. Y. Chen, C. Chen, S. Q. Liu and K. Zhang, Mobile visualization supporting awareness in collaborative software development, in *Proc. 7th Symp. on Visual Information Communication and Interaction*, 2014, pp. 113–120.
7. L. Chittaro, Visualizing information on mobile devices, *IEEE Computer* **39**(3) (2006) 40–45.
8. P. Dourish and V. Bellotti, Awareness and coordination in shared workspaces, in *Proc. ACM Conf. Computer-Supported Coop. Work*, 1992, pp. 107–114.
9. H. Drewes, A. De Luca and A. Schmidt, Eye-gaze interaction for mobile phones, in *Proc. 4th Int. Conf. Mobile Technology, Applications, and Systems and 1st International Symposium on Computer Human Interaction in Mobile Technology*, 2007, pp. 364–371.
10. J. Heer, F. B. Viégas and M. Wattenberg, Voyagers and voyeurs: Supporting asynchronous collaborative information visualization, in *Proc. SIGCHI Conference on Human Factors in Computing Systems*, 2007, pp. 1029–1038.
11. P. Fishwick (Ed.), *Aesthetic Computing* (MIT Press, 2006).
12. J. Hao and K. Zhang, A mobile interface for hierarchical information visualization and navigation, in *Proc. IEEE Int'l Symp. Consumer Electronics*, 2007, 1–7.
13. J. D. Herbsleb, A. Mockus, T. A. Finholt and R. E. Grinter, Distance, dependencies, and delay in a global collaboration, in *Proc. ACM Conf. Computer Supported Coop. Work*, 2000, pp. 319–328.
14. C. L. Ignat, Annotation of concurrent changes in collaborative software development, in *Proc. 4th Int'l Conf. Intelligent Computer Communication*, 2008, pp. 137–144.
15. B. Johnson, TreeViz: Treemap visualization of hierarchically structured information, in *Proc. SIGCHI Conference on Human Factors in Computing Systems*, 1992, pp. 369–370.
16. B. Johnson and B. Shneiderman, Tree-maps: A space-filling approach to the visualization of hierarchical information structures, in *Proc. IEEE Conference on Visualization*, 1991, pp. 284–291.
17. R. J. Jacob and K. S. Karn, Eye tracking in human-computer interaction and usability research: Ready to deliver the promises, *Mind* **2**(3) (2003) 4.
18. K. Luyten and K. Coninx, An XML-based runtime user interface description language for mobile computing devices, in *Interactive Systems: Design, Specification, and Verification*, 2001, pp. 1–15.
19. J. Lamping and R. Rao, Laying out and visualizing large trees using a hyperbolic space, in *Proc. 7th Annual ACM Symposium on User Interface Software and Technology*, 1994, pp. 13–14.
20. M. Manhartsberger and N. Zellhofer, Eye tracking in usability research: What users really see, in *Usability Symposium*, 2005, pp. 141–152.
21. N. Mahyar and M. Tory, Supporting communication and coordination in collaborative sensemaking, *IEEE Trans. Visualization and Computer Graphics* **20**(12) (2014) 1633–1642.
22. I. Omoronyia, J. Ferguson, M. Roper and M. Wood, A review of awareness in distributed collaborative software engineering, *Software: Practice and Experience* **40** (2010) 1107–1133.
23. C. Papadopoulos, Improving awareness in mobile CSCW, *IEEE Trans. Mobile Computing* **5** (2006) 133–1346.

24. M. K. Qiu, K. Zhang and M. L. Huang, An empirical study of web interface design on small display devices, in *Proc. IEEE/WIC/ACM International Conference on Web Intelligence*, 2004, pp. 29–35
25. M. K. Qiu, K. Zhang and M. L. Huang, Usability in mobile interface browsing, *Web Intelligence and Agent Systems* **4**(1) (2006) 43–59.
26. D. Redmiles, A. van der Hoek, B. Al-Ani, T. Hildebrand, S. Quirk, A. Sarma, R. Silva Filho, C. de Souza and E. Trainer, Continuous coordination — A new paradigm to support globally distributed software development projects, *Wirtschafts Informatik* **49** (2007).
27. C. de Souza, S. Quirk, E. Trainer and D. F. Redmiles, Supporting collaborative software development through the visualization of socio-technical dependencies, in *Proc. Int'l ACM Conf. Supporting Group Work*, 2007, pp. 147–156.
28. A. Sarma, D. Redmiles and A. van der Hoek, Palantir: Early detection of development conflicts arising from parallel code changes, *IEEE Trans. Software Engineering* (2011) 1–1.
29. J. C. Tang, N. Yankelovich, J. Begole, M. Van Kleek, F. Li and J. Bhalodia, ConNexus to awarenex: Extending awareness to mobile users, in *Proc. SIGCHI Conf. Human Factors in Computing Systems*, 2001, pp. 221–228.
30. R. Alley (Ed.), Catalogue of Tate Gallery's Collection of Modern Art Other than Works by British Artists (Tate Gallery and Sotheby Parke-Bernet, London, 1981). Retrieved 18 Dec. 2007, pp. 532–533.
31. H. Y. Yoo and S. H. Cheon, Visualization by information type on mobile device, in *Proceedings of Asia-Pacific Symposium on Information Visualisation*, 2006, pp. 143–146.
32. K. Zhang (ed.), *Software Visualization — From Theory to Practice* (Kluwer Academic Publishers, 2003).
33. K. Zhang, From abstract painting to information visualization, *IEEE Computer Graphics and Applications*, 2007, pp. 12–16.

Visualized Awareness Support for Collaborative Software Development on Mobile Devices

Citation:

Chen, Meng-Yao, Cong Chen, Shu-Qing Liu, and Kang Zhang. 2015. "Visualized awareness support for collaborative software development on mobile devices." International Journal of Software Engineering and Knowledge Engineering 25(2), doi:10.1142/S0218194015400094