Contextualized Awareness: A Lynchpin of Software Development

Anita Sarma
University of California, Irvine
Donald Bren School of Information and Computer Sciences
Department of Informatics
Irvine, CA 92697-3440 USA
asarma@ics.uci.edu

1. Introduction

Software development is a complex socio-technical endeavor involving intricate interdependencies among artifacts and multifaceted interactions among developers. Breakdowns in coordination are often the primary cause for conflicts in software development. Problems involved in coordination have been extensively researched by both the Software Engineering and the CSCW communities and has produced two very distinct coordination methodologies: the *formal* and the *informal* approach to coordination [1].

On the one hand, collaborative Software Engineering tools to date follow the formal process-based approach and discretize time and tasks in concrete, but isolated process steps. This discretization enables these solutions to scale well and be efficient. However, this approach is fundamentally flawed in assuming that human activity can be codified and that periodic resynchronization of tasks is an easy step. On the other hand, coordination tools provided by the CSCW community reflect the exact opposite approach: rather than constraining and guiding a user in their tasks, the focus is on informally raising awareness by informing users of ongoing, parallel activities so that they can interpret this information and self-coordinate amongst themselves. While this has lead to novel tools and approaches, these tools suffer from issues of scalability and cognitive overload.

The formal and the informal approaches have their strengths and weaknesses and there is an emerging need for a new approach that combines the strengths of these approaches while overcoming their shortcomings. It has been widely noted that developers often create ad-hoc processes around the established approach to facilitate coordination in their teams. For instance, it was observed that developers in companies that followed a formal process-based approach (configuration management system) created informal processes of sending email or conversing over Instant Messaging to circumvent the system and enable fellow developers to be aware of forthcoming changes [2, 3].

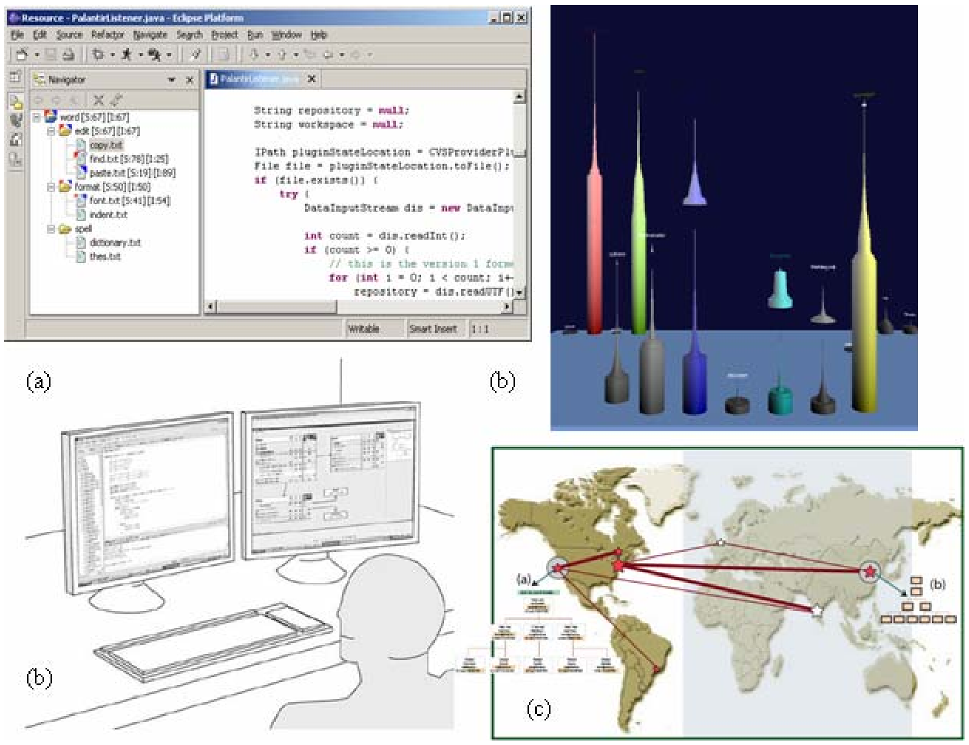
Our research proposes an integrated approach that combines formal and informal coordination to provide both the tools and the information necessary for users to self-coordinate. We call this approach the Continuous Coordination paradigm [1]. In particular, tools following this paradigm should retain the checkpoints and measures of the formal approach to coordination, while providing developers a view of each others relevant activities between the formal checkpoints. That is, following this approach software development will retain the underpinnings of the formal processes (e.g., discretized task units, private workspaces, and synchronization points), but will promote awareness of relevant ongoing activities so that developers can create a context for their work. In doing so, these tools can (a) provide developers with measures to understand the potential relationships among their individual work and the work of their colleagues and (b) ensure there is no information overload by only transmitting relevant and timely information.

Creating such tools, however, requires a thorough understanding of the kinds of information that is required to assist a developer in the specific task at hand. To accomplish this, we need to carefully explore the following three aspects of information: (1) the kinds of information that is to be shared, (2) the granularity of the information that is to be provided, and (3) the presentation of information that is appropriate to a given situation or member of the team.

In Section 2 of this paper, I describe my experiences in researching different collaboration tools. In Section 3, I discuss the insights that I have gained and the open research questions that the community faces. Section 4 explains my goals for attending the workshop followed by a short bio in Section 5.

2. Interest & Experiences

My research interests lies in the intersection of Software Engineering and CSCW. Thus far, I have researched novel conceptual approaches to coordination and designed, as well as implemented, tools that draw on these approaches to improve coordination in distributed software development.



```
String repository = null;
String workspace = null;

IPath pluginStateLocation = CVSProviderPl
File file = pluginStateLocation.toFile();
if (file.exists()) {
    try {
        DataInputStream dis = new DataInp

        int count = dis.readInt();
        if (count >= 0) {
            // this is the version 1 form
            for (int i = 0; i < count; i+
                repository = dis.readUTF()
```

Figure 1. Clockwise from top: (a) Palantír, (b) Workspace Activity Viewer, (c) World View, and (d) Lighthouse.

During the course of my work, I have been involved in the implementation of four research prototypes, namely, Palantír [4], the Workspace Activity Viewer [5], Lighthouse [6], and the World View [7]. Each of these tools follows the Continuous Coordination paradigm, but provides a unique flavor of coordination support.

Palantír is a workspace awareness tool that is geared towards the individual developer to provide contextualized visualizations that present information of ongoing parallel activities that may affect artifacts in the local workspace [4]. Specifically, Palantír presents information of which artifact is being changed by which other developer, calculates a measure of the size of the change based on the relative lines of code changed, and computes the impact of remote changes on artifacts in the local workspace. Palantír currently provides a set of four visualizations, each varying in the degree of detail and obtrusiveness so that developers can choose the visualization that best suits their preferences. Figure 1(a) shows the Palantír Eclipse integration.

The Workspace Activity Viewer is a three-dimensional visualization that builds on the Palantír infrastructure to provide an overview of activities of the entire project [5]. The Workspace Activity Viewer can be used in two distinct ways (see Figure 1(b)). First, managers can obtain a high-level, real-time view of the state of the entire project, so that they can quickly gain an insight of project level concerns. For example, if artifacts are frequently being modified in multiple workspaces, this may be a sign of overlapping task responsibilities. Second, the Workspace Activity Viewer provides a movie-like capability to provide insight into the evolution of a project by replaying past events. This enables retrospective analyses to understand when a project was stagnating, this may point to the project falling behind schedule, or undergoing a frenzy of activities.

Lighthouse is a coordination tool that tracks the emerging design – a real-time representation of the design as it is being implemented through code [6]. Lighthouse overlays the "emerging design" with the original design such that any deviation from the original can be easily identified. Additionally, Lighthouse provides information of which developer is responsible for the emerging design, the status, and the latency of changes to facilitate coordination among team members. Lighthouse, therefore, provides an integrated visualization that interlaces configuration management, awareness, and design in a unified approach.

The World View application provides a comprehensive view of the team dynamics of a project, regarding the geographical location of teams, the time zones of their operations, and the interdependencies among teams [7]. This view is intended to help developers involved in global software identify global and local team members, interactions between sub-groups, and other vital information like how and when to contact global member (see Figure 1(c)).

3. Open Research Issues

Constructing tools that follow the Continuous Coordination paradigm raises several questions regarding, among others, which information should be shared, how to avoid overloading developers with information, scalability, and general effectiveness of the approach in helping developers coordinate their tasks. The tools that I have researched thus far serve as an initial investigation into the feasibility of providing awareness to facilitate coordination in distributed software development. Each of these tools provides information in different formats and for different stakeholders. For example, while Palantír and Lighthouse provide fine-grained details necessary for the day-to-day activities of individual developers, Workspace Activity Viewer and World View provide high level information more suited for project management purposes.

The following are key considerations for improving workspace awareness tools.

- *Shared information*: identifying the "correct" information that is required or helpful to a user involved in a particular task. The aforementioned prototypes build on configuration management systems and leverage its infrastructure to draw information of activities and "push" this information to all workspaces. Particularly, information of workspace operations are collated and transmitted across workspaces to create awareness of activities. However, this information alone could be insufficient and other sources of information (e.g., chats messages, issue trackers, meeting minutes) might prove to be useful.
- *Granularity of information*: the "right" level of abstraction and analysis that is required to suit the needs of different stakeholders. Merely transmitting raw information about activities of others will simply lead to information overload. In order to identify the kinds of analysis to be done or the abstraction required, tool builders must first identify the users of the tools, the kinds of tasks they perform, the coordination problems that they face, and the manner in which they currently gain relevant information for their tasks.
- *Information presentation:* the manner in which information is presented is a critical deciding factor in a collaboration tool's adoption. We have taken different approaches through our prototypes. For instance, Palantír presents information in an unobtrusive contextualized manner by embedding awareness information in the development environment. Lighthouse is best used as a peripheral awareness mechanism with a dual monitor system setup (see Figure 1(d)). Whereas, the World View and the Activity Viewer are designed to be centralized large screen displays acting as a focal point for meetings and project discussions. There exist tradeoffs in the amount of information that can be presented and the obtrusiveness of the display. Finding the right balance is critical for a collaborative tool, but is often difficult to achieve.

4. Workshop Goals

My primary goal is to develop a deeper understanding of the human aspects inherent in the coordination activities related to supporting the software development lifecycle. This understanding coupled with my background in software engineering will provide me with the tools to develop a more nuanced approach to investigating and developing mechanisms to address the key considerations discussed in the previous section. Furthermore, I look forward to interacting with fellow researchers to seek better ways of collecting, analyzing, and presenting awareness generating information, such that it is easy-to-use, timely, and beneficial to users.

5. Bio

Anita Sarma is a doctoral candidate in the Department of Informatics of the Donald Bren School of Information and Computer Sciences at the University of California, Irvine. Her research interests include configuration management systems and their use for coordination and collaboration. She is the primary developer of Palantír and World View.

6. Acknowledgements

Effort partially funded by the National Science Foundation under grant numbers CCR-0093489, DUE- 0536203, and IIS-0205724. Effort also supported by an IBM Eclipse Innovation grant and an IBM Technology Fellowship.

7. References

[1] A. van der Hoek, et al. *Continuous Coordination: A New Paradigm for Collaborative Software Engineering Tools. Workshop on Directions in Software Engineering Environments*. 2004. p. 29-36.

[2] R.E. Grinter. *Using a Configuration Management Tool to Coordinate Software Development. Conference on Organizational Computing Systems*. 1995. p. 168-177.

[3] C.R.B. de Souza, D. Redmiles, and P. Dourish. *"Breaking the Code", Moving between Private and Public Work in Collaborative Software Development. International Conference on Supporting Group Work (Group 2003)*. 2003. p. 105 -114.

[4] A. Sarma, Z. Noroozi, and A. van der Hoek. *Palantír: Raising Awareness among Configuration Management Workspaces. Twentyfifth International Conference on Software Engineering*. 2003. p. 444-454.

[5] R. Ripley, A. Sarma, and A. Van der Hoek, *Using Visualizations to Analyze Workspace Activity and Discern Software Project Evolution*. 2006, University of California, TR: UCI-ISR-06-1.

[6] C. van der Westhuizen, P. Chen, and A. van der Hoek. *Emerging Design: New Roles and Uses for Abstraction. Workshop on the Role of Abstraction in Software Engineering: Organizational, Managerial and Cognitive Perspectives*. 2006. Electronic proceedings, 6 pages.

[7] A. Sarma and A. Van der Hoek. *Towards Awareness in the Large. First International Conference on Global Software Engineering (to appear)*. 2006.