Using Visualizations to Analyze Workspace Activity and Discern Software Project Evolution

Roger M. Ripley, Anita Sarma, and André van der Hoek

Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425
{rripley,asarma,andre}@ics.uci.edu

ISR Technical Report # UCI-ISR-06-1
January 2006

Abstract. Real-time awareness of other developers' activities is gaining acceptance as a complementary strategy to traditional SCM (software configuration management) locking and branching. Thus far, this type of awareness—including our work—has focused only on *individual* developers, with information regarding individual artifacts provided in a contextualized visualization. Here, we build upon our prior work but take a broader perspective: visualization and exploration of workspace activity and evolution on a *project-wide* basis. We believe this visualization will help not only developers, who can benefit from this high level view by understanding how their work relates, but, more importantly, *managers*, who now have a comprehensive view of all project activities, allowing them to intelligently steer development and adjust task assignments. Another interesting aspect of our work is that we can visualize the evolution of workspaces—and the emergent project evolution—either live or *post mortem*: since our tool stores all the workspaces' events, we can replay, stop, rewind, and visually inspect the effort at any given point in time to find trends, problems, and other patterns of interest.

1 Introduction

Coordination of development activities is one of the core tasks of software configuration management (SCM) systems. SCM systems typically use one of two strategies to coordinate multiple developers working on a common set of artifacts: enforcing sequential development via locking or maintaining parallel branches of concurrent development for later merging. Lately, awareness is being used to complement these SCM strategies of locking and merging. The hypothesis is that, armed with awareness of parallel activities, developers can self-coordinate their actions by placing their work in the context of others' actions.

While some awareness tools (CVS watch [1], COOP/Orm [2]) provide awareness of activities taking place in the repository, others take a step further and provide workspace awareness by informing developers of ongoing parallel changes (State Treemap [3], Jazz [4], Palantír [5]). These awareness tools generally provide information regarding which artifacts are being changed by which developers in parallel, along with other metadata for the changes (e.g., when was the

artifact changed, what is the size of the change, would the change cause any artifact in the local workspace to be out-of-sync). Awareness information is either presented to the user as separate visualizations or by embedding awareness widgets in the development environment.

Thus far awareness tools (including our work) have focused only on needs of individual developers, with information regarding individual files provided in contextualized visualizations. While these tools do help developers in identifying "remote" concurrent changes to artifacts that are present in their "local" workspace, they fall short in representing the full extent of parallel activities of the entire team. None of these tools describe the effect of the changes on other artifacts in the project, the evolution of the system, or the development process within a workspace. These tools are marginally useful to a project manager looking for guidance in task realignment or even confirmation of instincts.

In this paper, we build on our previous work to provide a workspace activity viewer that allows one to visualize and explore workspace activities and their evolution on a project-wide basis. The workspace activity viewer is a three-dimensional (3D) visualization tool that builds on the underlying infrastructure of our research prototype, Palantír, and its events [6]. The 3D visualization gives an overview of all workspaces at the same time, either from the perspective of developers or artifacts, while simultaneously allowing the viewer to discern the loci of activity, by placing the most important and relevant data at the forefront of the visualization. We believe that this view will help both developers, who can benefit from the high level view and place their work in the context of others' activities, and managers, who can now get a comprehensive view of all project activities and adjust task assignments if needed. In addition to showing the current state of a project, there is a movie-like capability to replay past events, which can be used for analysis of project dynamics and other forensics. Visualizing this quantity of data may lead to scalability issues, an issue which we address integrally through the use of user-definable filters.

To test the behavior of our tool we ran it on five open source projects, of which two will be discussed later. Since we lacked real workspace activity data—CVS has no provision to capture it—we simulated actual workspace activities based on randomizations of the patterns in between the known check-outs and check-ins. While this necessarily represents a limitation, it does allow us to make interesting observations regarding the evolution of real projects in real settings. Our next work, obviously, is aimed at actually instrumenting a real project with our data collection tool, so that we will be able to visualize accurate data regarding ongoing changes in workspaces, rather than simulated data in between check-ins. We are currently in the process of doing so with a commercial repository.

The remainder of this paper is organized as follows. Section 2 provides background into the need and acceptance of awareness to complement SCM systems. Next, in Sect. 3, we reintroduce Palantír, the basis for our work. Section 4 delves into the implementation of our workspace activity viewer. In Sect. 5 we demonstrate two projects that our tool has visualized. Next, we consider related work in Sect. 6 and conclude in Sect. 7 with an outlook at future work.

2 Background

Empirical studies have shown that awareness is a critical component that must be supported by SCM systems. Even though automated merging resolves about 90 percent of conflicts, the remaining 10 percent prove difficult to resolve and lead to lower quality software or project delays [7]. One of the ways developers compensate for this problem is by bypassing the SCM system and creating informal conventions to provide a level of awareness. In a study conducted at a tool development firm, it was found that developers used the check-in/check-out information available in the SCM system to identify which other developers were working on the same artifact, which artifacts have been checked out, and which tasks a particular developer is currently involved in [8]. The developers then used this information to coordinate their activities accordingly. Using this information a developer may race to check-in their own changes before their colleague to avoid having to do any conflict resolution on their part. In another study of a software development project, it was observed that developers regularly sent out email to the entire development mailing list before checking in their changes [9]. The email informed and warned other developers of impending changes and possible conflicts.

There are two approaches to providing awareness in SCM systems, repository-based awareness and workspace awareness. Repository-based awareness tools allow developers to register interest in and be notified when specific artifacts in the repository are changed (Coven [10], Night Watch [11], COOP/Orm [2]). While these repository-based awareness tools make developers aware of development activities upon completion, it is, as the aforementioned studies have shown, sometimes too late if a conflicting situation is discovered at this point.

Workspace awareness tools, on the other hand, provide developers with information of ongoing activities in others' workspaces in real time (Jazz [4], Palantír [5]). These tools therefore allow developers to detect conflicts earlier, as they are emerging during development. Early detection of potential conflicts provides developers with the opportunity to take proactive steps to avoid them.

While current workspace awareness tools, including our work, are geared solely toward the individual developer (showing how concurrent changes relate to artifacts in the local workspace), there are repository-based awareness tools [12–14] that track system evolution based on the lines of code that have changed over time. However, since these tools gather their information from the repository, they can only visualize changes that have already been committed, with no insight into what happened in the workspace itself in between commits. There is a need for workspace awareness tools that provide an overview of both past and ongoing changes in workspaces for better handling of project management.

3 Palantír Infrastructure

Our work in providing a 3D visualization for workspace activities builds upon our existing infrastructure for Palantír. In this section, we discuss the important parts of Palantír (on the philosophy and design of Palantír, see [5, 6]); in the next section, we discuss how we have leveraged these parts in building our solution.

Palantír is a workspace awareness tool based on the hypothesis that conflicts in parallel development can be considerably reduced, both in magnitude and number, by providing developers with an insight into ongoing project activities. Palantír does not supplant existing SCM systems, but builds on top of them: collecting, distributing, organizing, and presenting workspace information.

Palantír does not change the way users interact with the SCM system. Instead, Palantír silently intercepts the interactions of the developer with the repository (e.g., check-in, check-out, synchronize), monitors the activities of the developer in the workspace (e.g., edit, local save, local delete), and then transmits that information to other "remote" workspaces via Palantír events.

Events make Palantír independent of both the underlying SCM system and the development environment. Since different SCM systems follow different policies, events represent an artifact's state in a workspace, instead of just a captured action. Palantír is integrated with three SCM systems (RCS [15], CVS [1], Subversion [16]) and a development editor (Eclipse [17]), and can be integrated with others by changing the component responsible for capturing events. Palantír's entire set of events is described in [5].

We have implemented different visualizations to represent parallel activities. Each Palantír visualization presents information in varying degrees of detail and obtrusiveness; however, they are all geared toward the individual developer and reveal the rest of the project from the perspective of that developer. Here, we mention two of the visualizations: the Eclipse integration and the fully graphical visualization; descriptions of the other visualizations can be found in [6].

The Palantír-Eclipse integration (Fig. 1, left) "decorates" the package explorer view (a tree view of artifacts) with red triangles. These triangles serve as a quick alert, denoting the cumulative severity of changes made by other developers: the higher the severity, the larger and redder the triangle. Even though this alerts the developer that parallel activity is taking place, it does not provide a complete picture, such as where the development is taking place.

The fully graphical visualization (Fig. 1, right) complements the Eclipse integration by allowing for exploration of a hierarchical, stackable view of an artifact and its constituents. For instance, the stack for the artifact `/home/word/edit` indicates that Ellen, Pete, and Mike all have it in their workspace. Pete and Mike both are making changes to version 1.0, indicated by the question mark. Ellen, on the other hand, has checked in a new version of the artifact (indicated by the exclamation point), resulting in her workspace having version 1.1. The severity of changes is shown by the vertical blue lines. Other visualizations are similarly complementary by trading off intrusiveness, ease of use, and provided detail.

4 Workspace Activity Viewer

Thus far, Palantír has focused on individual developers, with contextualized visualizations providing awareness of activities that affect artifacts in the local workspace. We believe such self-centered views provide only a limited view of

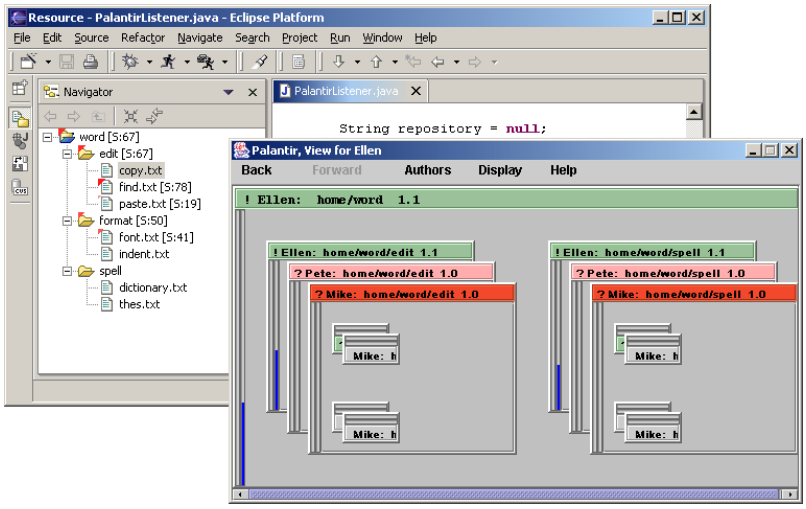


Fig. 1. Palantír visualizations: Eclipse integration and fully graphical.

activities in a project and that both developers and managers will benefit by having an overview of all concurrent workspaces and changes in each workspace.

Another shortcoming of current Palantír visualizations is that they only present the current view of project activities; they do not depict any project evolution based on the past and present activities in workspaces. It is well known that managers generally monitor the state of the project (e.g., when is the project stagnating, which parts of the project have concurrent changes, which developers are responsible for major code changes) to better manage their project and teams. As long check-out/check-in cycles hides potential conflicts, we believe a project management view that visualizes the history and project dynamics will help managers immensely in making these decisions.

The isolation of a typical SCM workspace necessarily limits insight into the development lifecycle of artifacts within that workspace (although a significant amount of work is currently aiming to mine software repositories, this work only can observe *results* of workspace activities, not the activities themselves [18]). Even though SCM systems are abundant sources of repository-based data, they nonetheless have little knowledge of workspace data such as whether an artifact is incrementally changed or scattered, whether changes generally are made artifact by artifact or in lots of artifacts at the same time, and so forth.

We have built a workspace activity viewer for Palantír that addresses these types of issues by (1) presenting an overview of the development activities of the entire team, and (2) providing insight into the evolution of the project.

4.1 Overview of Development Activities

The workspace activity viewer is a 3D visualization that presents a snapshot of all ongoing changes taking place in a set of workspaces at a particular time,

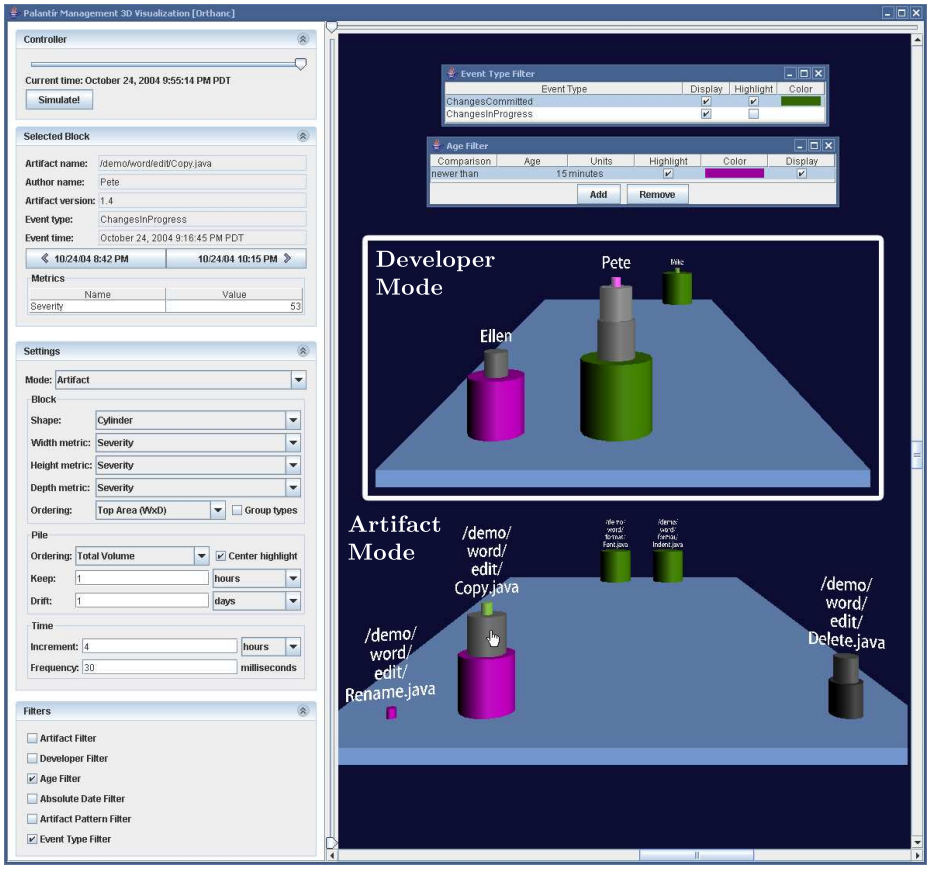


Fig. 2. 3D Visualization in Artifact Mode; Developer Mode inset.

as shown in Fig. 2. The visualization has two primary modes: *developer-centric* and *artifact-centric*, which we will discuss shortly. Common to both modes, the visualization shows only active artifacts and workspaces, thereby eliminating the clutter of inactive entities. Stacks of cylinders represent workspace activities (changes to artifacts), with each cylinder corresponding to a particular artifact in a particular workspace with the dimensions representing the size of the change (the bigger the change, the larger the cylinder).[1]

In the developer-centric view (see Fig. 2, inset), a stack of cylinders represents a developer's workspace with each cylinder representing an artifact being changed in that workspace. Workspaces with many activities correspond to tall stacks of cylinders. The stacks of cylinders with the most recent changes are placed in the front of the view and, as time elapses, stacks for workspaces with

[1] In the future, we plan to assign different metrics to each dimension (height, width, and depth), e.g., the number of interfaces that have changed.

"older" activity slowly start moving to the back, representing dormancy. Thereby a user is able to quickly discern the loci of activities from the height of the stacks and recency of these activities from their position. The artifact-centric view (see Fig. 2, bottom) behaves similarly to the developer view, but instead each stack of cylinders represents a particular artifact and each cylinder in the stack represents changes to that artifact made by a particular workspace.

For both views, a cylinder captures the state of an artifact in a workspace at a particular moment in time: it can represent either changes that have already been committed to the repository or ongoing changes in the workspace. Clicking on a cylinder displays meta-information about the change (e.g., whether the change is already committed or is in progress, the time of the change, which workspace made the change, the name of the artifact, and all the metrics generated as a result of the change) in the left panel of the visualization.

To ensure that users notice changes in the view, the cylinders slowly expand or contract to represent increasing or decreasing changes to artifacts. The shade of individual cylinders changes from dark to light as their distance from the base increases. Differently shaped stacks of cylinders represent different development practices. For example, a workspace with a large stack of small cylinders represents numerous incremental changes to several artifacts simultaneously (left workspace in Fig. 3), whereas a single large cylinder signifies a one-time large change to an artifact (center workspace in Fig. 3).

The view can be configured in a number of ways to represent different facets of project activities. By default stacks of cylinders that have a high occurrence of activities (number of cylinders in a stack) are placed in the center of the visualization to enable quick detection of the center of activity in the team space, but the sort conditions can be changed to show stacks of cylinders with the largest cumulative changes in the center (maximum volume), or cylinders with the single largest change in the center (the largest cylinder in the stack).

Filters. A number of filters can be applied to the view. Each filter has two options that can be applied to a cylinder or a stack of cylinders: *display* and *highlight*. Display determines if the entity will be shown to or hidden from the user. Highlight, on the other hand, visually accents the entity with a user chosen color and optionally, if the entity is a stack, centers it in the visualization. Table 1 gives a full overview of all possible filters; here we discuss one representative example.

The *artifact filter* presents a tree view of the artifacts in the project with the option to view, hide, or color each artifact individually. The artifacts `Copy.java`, `Delete.java` and `Rename.java` in Fig. 3 are highlighted in different colors: red, green and blue, respectively. The top cylinders on both Ellen's (right) and Pete's (left) stacks are green, signifying that both developers are working on `Delete.java`. Similarly, both Ellen's bottom cylinder and Pete's third cylinder from the top are shaded red: both are working on `Copy.java`. Pete's second cylinder from the top is shaded blue as he is the only developer working on `Rename.java`. Pete's bottom cylinder and Mike's only cylinder (center) are both shaded gray, indicating that they were not matched by the filter. Even so, a user

Table 1. Filters.

Artifact	The artifact filter presents the user with a tree view of the artifacts in the project. In artifact mode, where each stack represents an artifact, the filter affects an entire stack; otherwise, it affects only individual cylinders.
Developer	The developer filter presents the user with a list of developers in the project. In developer mode, where each stack represents a developer, the filter affects an entire stack; otherwise, it affects only individual cylinders.
Age	The age filter allows the user to set a matching criteria based on if an artifact is older than or newer than a certain relative time period to current (e.g., two weeks ago). The filter only applies to individual cylinders.
Absolute Date	The absolute date filter is like the age filter, but instead of the user choosing a time relative to current, the user specifies a specific point in time (e.g., May 12, 2002 9:00 AM). The filter applies to individual cylinders.
Artifact Pattern	The artifact pattern filter is similar to the artifact filter, but instead of presenting a tree view of artifacts, the user is able to enter regular expressions to match against artifact names. This allows the user to eliminate or highlight certain classes of artifacts (e.g., all graphics). In artifact mode, the filter applies to entire stacks; otherwise, it affects individual cylinders.
Event Type	The event type filter allows the user to match cylinders that were generated because of certain event types (e.g., setting commits and changes in progress to different colors). The filter only applies to individual cylinders.

could click on the cylinders to identify those artifacts. Color coding artifacts allows users to quickly trace which workspaces are changing specific artifacts. Other filters are discussed in the context of the case studies (Sect. 5).

Rotation. A drawback of the aforementioned view is that as changes become old the cylinders move to the back. In situations with a large number of recent changes, changes that have occurred in the past are hard to discern behind cylinders for recent changes. To alleviate this situation, the visualization can be rotated about both the X and Y axes to better portray evolution with respect to changes over time. The visualization can be rotated about the X-axis to present an overhead view of the activities in a set of workspaces (see Fig. 4). This view represents recency of changes based on the relative position of the stacks and concurrency based on the color gradient of the cylinders. Similar to a shaded relief map, cylinders near the base are dark and lighten as the stacks grow. In Fig. 4, the three artifacts on the bottom have recent changes (with the furthest left being slightly less recent), while the two on top have been changed in the past.

4.2 Tracing Project Evolution

Project history information is critical for managers to understand the dynamics of the project, the contribution and responsibilities of individual developers, and the health of the project. Typically managers track project dynamics using tools that are not geared toward this use: bug trackers, email archives, as well as personal experience and memory.

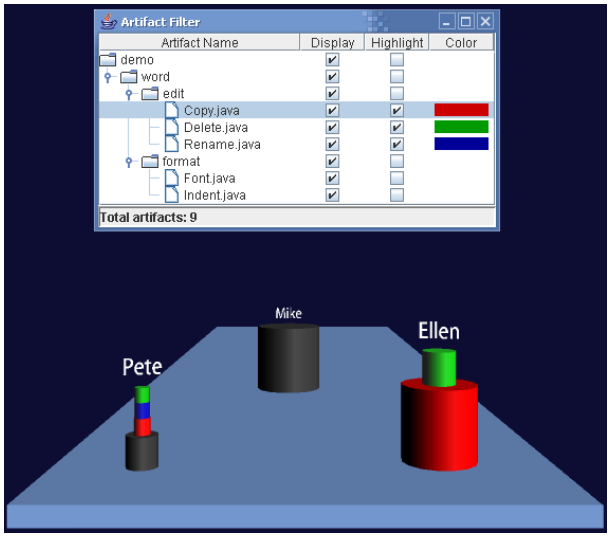


Fig. 3. Development styles with artifact filter.

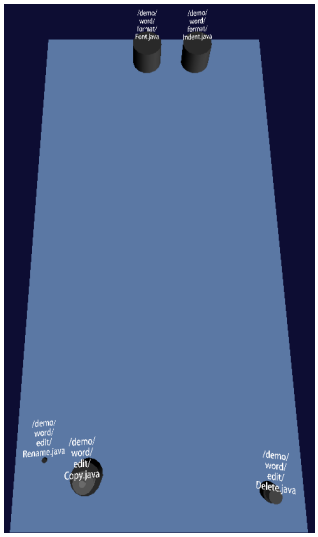


Fig. 4. Overhead view.

Palantír stores the history of all the workspace activities in a separate database. The workspace activity viewer takes advantage of this to give the user a movie-like ability to replay, pause, rewind, and visually inspect the project at any given point in time to find trends, problems, and other patterns of interest. This playback can either be as a continuous animation or in discrete steps. The history can be played backward or forward in time, and the period of time that needs to be visualized can be limited by either using the age or absolute date filters.

Certain obvious uses of being able to see the project evolution are to gain understanding regarding when the project was stagnating, when there were flurries of activity (can signify potential conflict zones), and which artifacts have been changed multiple times or by multiple developers (can signify unstable artifacts). This information can in turn allow managers to understand which artifacts are the cornerstones in the project and are used by most developers, whether task assignments were clear or should be revised, which developers contribute more, and so forth. Effects of external factors (e.g., hiring, termination, team refactoring and new management) on the project can also be observed by correlating the activities in the project space with the specific events.

5 Case Studies

To test the behavior of our tool we ran it on five open source projects, listed in Table 2. We discuss two of those projects, FreeMind and Gaim, in Sects. 5.1 and 5.2 respectively. The only data we had available was what was contained in the CVS repository. Since we lacked workspace activity data, we simulated workspace activities between check-outs and a check-ins.

Table 2. Summary of projects simulated and visualized.

Project Name	Initial Commit	Developers	Artifacts	Simulated Events
ArgoUML [19]	1998/01/26	15	1635	483,026
FreeMind [20]	2000/08/01	5	637	151,490
Gaim [21]	2000/03/22	22	1861	1,180,192
jEdit [22]	2001/09/01	8	1431	450,732
Scarab [23]	2000/12/17	37	2305	884,030

To generate the simulation data, we used the CVS metadata for each commit (who, when, and how much changed) to establish points of known state for each artifact. Before each of these check-ins, we generate events indicating that the artifact is being changed by the developer who checked it in, with the frequency and severity (magnitude of change) increasing as the commit time approaches. As a result, our simulation is accurate at commits, and plausible in between.

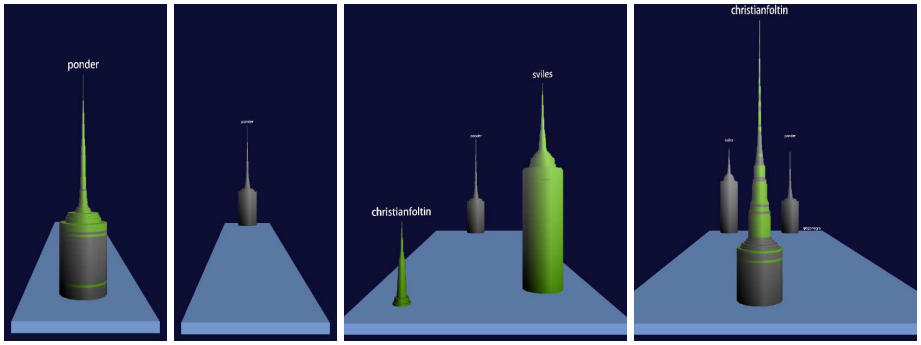
5.1 FreeMind

FreeMind [20] is mind-mapping software which, in essence, allows one to organize knowledge and thoughts. With the FreeMind project we were able to witness the evolution of the project from creation by one developer, to stagnation, to revitalization as new developers took the lead.

FreeMind's first commit took place in August 2000. Through September 2001, the creator of FreeMind, *ponder* (Jörg Müller), worked alone on the project. Figure 5(a) shows an example of that pattern from April 2001. There is only one stack, *ponder*'s. There is an active age filter highlighting artifacts in green that have been modified within the past month, and the stack is about two-thirds green. From September 2001 until October 2003, there was no activity on the project, as show in Fig. 5(b); the stack is at the rear of the field, and all the artifacts have reverted to gray. At the end of October 2003 two new developers start contributing to the project, as shown in Fig. 5(c). *sviles* (Stephen Viles) contributes primarily graphics—this is his only major contribution to the project. *christianfoltin* (Christian Foltin), who will turn out to be a major force in the development of FreeMind, is just starting to contribute. Both their stacks are solid green and at the front of the field. By March 2004, *christianfoltin* has indeed become the dominant force in the project, confirmed in Fig. 5(d). His stack is the only stack at the front of the field, is the tallest stack by far, and is over half green—indicating active work on many artifacts. We were also able to correspond these observations to the "authors and contributors" section of the FreeMind website which documents who worked on the project for each version [20].

5.2 Gaim

Gaim [21] is a multi-protocol instant messaging client. Figure 6 shows the Gaim project in March 2002, when a majority of the development effort was focused



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

Fig. 5. Evolution of the FreeMind project.

on graphics. An artifact pattern filter is highlighting C header files (`.h`) in red, C source files (`.c`) in blue, and graphics (primarily pixmaps: `.xpm`) in yellow. The option to separate changes in progress and changes committed is on, so the cylinders on top are current editing and those on bottom have been checked into the repository. Most of the activity is from *robflynn* (Rob Flynn) (stack on front left, almost solid yellow), but *seanegan* (Sean Egan) is also involved (short stack on front center, mostly yellow), and, to a lesser extent, *warmenhoven* (Eric Warmenhoven) (tall stack on back center, approximately 25% yellow). Just prior to this screenshot, all the graphics on *robflynn*'s stack were being edited (cylinders on top), then checked in (jumped to the bottom). In this screenshot, some of the graphics are being edited again (jumped back on top), such as making some pixmaps transparent and resizing others.

5.3 Discussion

Using our visualization, it has been easy to identify the core developers and to watch project evolution (periods of high activity and of stagnation, core developers "overtaken" by others). We have also been able to notice interesting situations, such as graphics being the artifacts changed by the most developers. Despite having looked at five different projects, it has been surprisingly problematic to find many instances of truly parallel work on the same artifacts and definitely not the quantity we would have expected. We believe this may be due to our choice to visualize open source projects, and have surmised some possible explanations for this phenomenon:

1. We have noticed during cursory looks at the CVS commit messages for the projects we visualized that commits are often made on behalf of other developers, therefore any conflicting parallel work is reconciled by these "gatekeepers" to the repository, and parallel work is "flattened" by the commits appearing to come from one person.

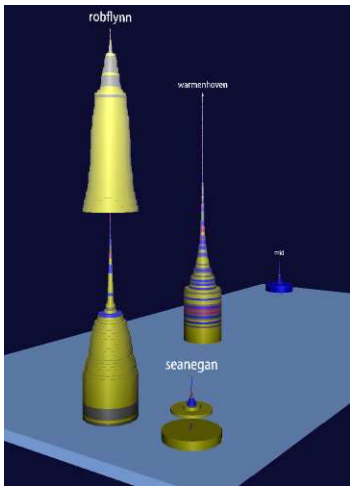


Fig. 6. Gaim graphics activity.

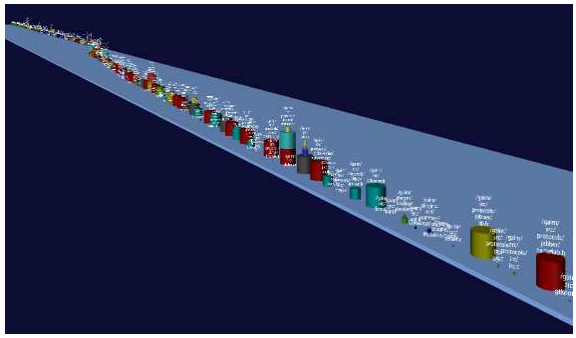


Fig. 7. Gaim project artifact view.

2. We do not believe that most open source projects suffer from the same urgency of deadlines as commercial software does. Therefore, there would not be a rush of commits as various milestones approach. Developers also have the time to coordinate via email or instant message and wait for a reply.
3. Since open source developers are mainly volunteers, time wasted on parallel work equates to time lost. If a commercial developer is salaried, time wasted on parallel work does *not* equate to money lost (for the developer), however; the incentive to maximize self-productivity is not equivalent.

At first glance it may appear that a further issue is the scalability of the visualization when there are many stacks (such as the artifact view in a large project), as in Fig. 7. In reality, however, the value of the visualization does not diminish because we are looking for *large-scale patterns* across the entire project space. Individual cylinders can be selected for detail regarding the artifact and state, or the visualization can be zoomed in to focus on a particular area. We have been able to visualize projects with many hundreds of artifacts, and as multi-monitor setups become more commonplace this will only increase.

We are even able to use this small screenshot of the large visualization to make some interesting conclusions. Despite being a significantly large project with a significant number of developers, we note that the amount of parallel work is relatively little. We also note that two developers are clearly the lead developers, but that a significant number of other developers have contributed to the project. Most of these contributions are older, though. We also can see that typically changes in the workspace are relatively large, and take some time to develop. A manager could use this view to note that the two stacks near the center have several cylinders, indicating multiple developers, alerting the potential of overlapping task assignments—or perhaps a design flaw. Additionally, as we

already noted, there are two primary developers. This could signal unbalanced task assignments that should be adjusted.

6 Related Work

In addition to the SCM awareness tools we have discussed in Sect. 2, our work clearly relates to tools for workspace based awareness and system evolution.

Awareness tools (GroupSketch [24], MMM [25], Quilt [26]) in the realm of Computer Supported Cooperative Work (CSCW) have long provided workspace awareness, but these tools are primarily geared toward collaborative document authoring. Collaborative authoring typically requires tightly coupled awareness information (e.g., mouse clicks, keyboard events, multiple tele-pointers). The large amount of awareness information that these editors process makes them unsuitable for large teams. Moreover, such tightly coupled modes of collaboration are not required for software development. Researchers have recognized the need for workspace awareness in a software development setting and are currently investigating prototypes that address this need. Most of the research prototypes generally provide information of concurrent activities on the same artifact (e.g., State Treemap [3], Group Desk System [27], Jazz [4]) but do not provide any information on the size of the change or its effect on other artifacts. While this information is useful in understanding parallel activity, it can be greatly enhanced by providing more information on the nature of the change. Some early research prototypes (Night Watch [11], Palantír [5]) are taking the first steps in providing more information about these changes. However, most of these tools present awareness from the perspective of a single user. None of these tools track all changes in all workspaces. As far as we know O'Reilly's work [28] is the only previous attempt to visualize activities taking place in all workspaces, however, this visualization does not record a history of the activities.

There are a number of information visualization tools that represent software systems with respect to how they have evolved over time. These tools either represent the system based on the lines of code in each file [13, 12] or visualize the overall structure of the system (i.e., the constituent classes and interfaces) [29, 30] from the structural level. These tools visualize the changes based on the information available in the repository and therefore suffer the same drawback as awareness tools that depend on repositories: ignorance of workspace activity.

7 Conclusions and Future Work

In this paper, we presented a prototype of a 3D visualization for workspace activity. This visualization shows only active workspaces and artifacts and depicts activities as changes to artifacts. Each change to an artifact is denoted as a cylinder and stacks of such cylinders represent activities in a particular workspace (developer-centric) or activities carried on in different workspaces to a specific artifact (artifact-centric). We applied the visualization to several open-source projects and demonstrated project evolution and other interesting situations.

To truly examine the visualization and how it handles very large systems, we plan on running it on the HIPerWall, a 55 tile (11×5 tiles), 200 mega-pixel (28,160×8,000 pixels) massively tiled display system. This will give us an indication of even the largest projects in their entirety, which should yield some interesting results in terms of our understanding.

We are planning a case study at a private company where we will analyze their SCM repository to determine what different and interesting situations we may find. This will also serve to verify the observations regarding the nature of open-source projects made in Sect. 5.3. Additionally, deploying the workspace activity viewer to the management of a software project will be invaluable in determining the actual use and usefulness of the visualization.

Filters are useful for highlighting or hiding entities based on simple criteria. Oftentimes, what is most interesting, however, is sequences of events that occur over a period of time. We intend to provide a method for expressing complex patterns of that style and then matching on them. We will also provide a library of both filters and patterns for common uses.

8 Acknowledgments

Effort partially funded by the National Science Foundation under grant numbers CCR-0093489 and IIS-0205724, and an Eclipse Innovation Grant, 2004.

References

1. Berliner, B.: CVS II: Parallelizing software development. In: USENIX Winter 1990 Technical Conference. (1990) 341–352
2. Magnusson, B., Asklund, U.: Fine grained version control of configurations in COOP/Orm. In: Sixth International Workshop on Software Configuration Management. Volume 1167. (1996) 31–48
3. Molli, P., Skaf-Molli, H., Bouthier, C.: State treemap: an awareness widget for multi-synchronous groupware. In: International Workshop on Groupware. (2001)
4. Cheng, L.T., Hupfer, S., Ross, S., Patterson, J.: Jazzing up Eclipse with collaborative tools. In: 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications / Eclipse Technology Exchange Workshop, Anaheim, California (2003) 102–103
5. Sarma, A., Noroozi, Z., van der Hoek, A.: Palantír: Raising awareness among configuration management workspaces. In: Twenty-fifth International Conference on Software Engineering, Portland, Oregon (2003) 444–454
6. Sarma, A., van der Hoek, A.: Visualizing parallel workspace activities. In: IASTED International Conference on Software Engineering and Applications (SEA), Marina Del Rey, California (2003) 435–440
7. Perry, D.E., Siy, H.P., Votta, L.G.: Parallel changes in large-scale software development: An observational case study. ACM Transactions on Software Engineering and Methodology **10** (2001) 308–337
8. Grinter, R.E.: Supporting articulation work using software configuration management systems. Computer Supported Cooperative Work **5** (1996) 447–465

9. de Souza, C.R.B., Redmiles, D., Dourish, P.: Analyzing transitions between private and public work in collaborative software development. In: International Conference on Supporting Group Work (Group 2003), Sanibel Island, Florida (2003)
10. Chu-Carroll, M.C., Sprenkle, S.: Coven: Brewing better collaboration through software configuration management. In: Eighth International Symposium on Foundations of Software Engineering. (2000) 88–97
11. O'Reilly, C., Morrow, P., Bustard, D.: Improving conflict detection in optimistic concurrency control models. In: Proceedings of the Eleventh International Workshop on Software Configuration Management, Portland, Oregon (2003) 191–205
12. Froehlich, J., Dourish, P.: Unifying artifacts and activities in a visual tool for distributed software development teams. In: 26th International Conference on Software Engineering, Edinburgh, United Kingdom (2004) 387–396
13. Eick, S.G., Steffen, J.L., Summer, Jr., E.E.: Seesoft—a tool for visualizing line oriented software statistics. IEEE TSE, Special issue on software measurement principles, techniques, and environments **18** (1992) 957–958
14. Eick, S.G., Graves, T.L., Karr, A.F., Marron, J.S., Mockus, A.: Does code decay? assessing the evidence from change management data. IEEE Transactions on Software Engineering **27** (2001) 1–12
15. Tichy, W.F.: RCS, a system for version control. Software—Practice and Experience **15** (1985) 637–654
16. Tigris.org: Subversion: Project home. `http://subversion.tigris.org/` (2005)
17. Ripley, R.M., Yasui, R.Y., Sarma, A., van der Hoek, A.: Workspace awareness in application development. In: 2004 Eclipse Technology Exchange Workshop (collocated with OOPSLA 2004), Vancouver, Canada (2004)
18. Hassan, A.E., Holt, R.C., Diehl, S., eds.: Proceedings of the 2nd International Workshop on Mining Software Repositories (MSR 2005), St. Louis, Missouri (2005)
19. Tigris.org: ArgoUML: Project home. `http://argouml.tigris.org/` (2005)
20. FreeMind Project: FreeMind: free mind mapping software. `http://freemind.sourceforge.net/` (2005)
21. Gaim Project: Gaim: A multi-protocol instant messaging (IM) client. `http://gaim.sourceforge.net/` (2005)
22. jEdit Project: jEdit: Programmer's text editor. `http://www.jedit.org/` (2005)
23. Tigris.org: Scarab: Project home. `http://scarab.tigris.org/` (2005)
24. Greenberg, S., Bohnet, R.: GroupSketch: A multi-user sketchpad for geographically-distributed small groups. In: Proceedings of Graphics Interface. (1991)
25. Bier, E.A., Freeman, S.: MMM: A user interface architecture for shared editors on a single screen. In: 4th annual ACM Symposium on User Interface Software and Technology. (1991) 79–86
26. Fish, R.S., Kraut, R.E., Leland, M.D.P.: Quilt: a collaborative tool for cooperative writing. ACM SIGOIS Bulletin **9** (1988) 30–37
27. Fuchs, L., Pankoke-Babatz, U., Prinz, W.: Supporting cooperative awareness with local event mechanism: The group desk system. In: European Computer Supported Cooperative Work (ECSCW'95), Kluwer, Dordrecht (1995) 247–262
28. O'Reilly, C., Bustard, D., Morrow, P.: The war room command console (shared visualization for inclusive team coordination). In: SoftVis '05: 2005 ACM symposium on Software visualization, St. Louis, Missouri (2005) 57–65
29. Lanza, M.: The evolution matrix: Recovering software evolution using software visualization techniques. In: 2001 International Workshop on the Principles of Software Evolution. (2001) 28–33
30. Baker, M.J., Eick, S.G.: Space-filling software visualization. Journal of Visual Languages and Computing **6** (1995) 119–133