A Visualization for Software Project Awareness and Evolution

Roger M. Ripley, Anita Sarma and André van der Hoek
University of California, Irvine
Donald Bren School of Information and Computer Sciences
Department of Informatics
Irvine, CA 92697-3425 USA
{*rripley,asarma,andre*}*@ics.uci.edu*

Abstract

Real-time awareness of other developers' activities is a powerful tool to assist in coordination of developer activities. Thus far, this type of awareness 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 helps project managers who now have a comprehensive view of all project activities, allowing them to intelligently steer development and adjust task assignments. Developers can also benefit from this high level view by understanding how their work relates with each other and to the project as a whole. Another interesting aspect of our work is that we can visualize the evolution of workspaces—and the emergent project evolution—either live or postmortem: 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 activities in a software development project. Awareness tools, and their associated visualizations, aim to support developers by arming them with awareness of parallel activities, allowing them to self-coordinate their actions by placing their work in the context of others' actions.

Some awareness tools (CVS watch [1], COOP/Orm [2]) provide awareness of activities taking place in the repository, while 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.

Workspace awareness tools tend to focus 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.

While current workspace awareness tools are geared solely toward the individual developer, there are repository-based awareness tools. Some track system evolution based on the lines of code (or other calculated metrics) that have changed over time (Augur [6], CVSscan [7], Seesoft [8], Spectrographs [9]). Others visualize the overall structure of the system, that is, the constituent classes and interfaces, from the structural level (Ariadne [10], Evolution Matrix [11], SeeSys [12]). 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.

In this paper, we build on our previous work to provide an activity viewer that allows one to visualize and explore workspace activities and their evolution on a project-wide basis. The activity viewer is a three-dimensional (3D) visualization tool that builds on the underlying infrastructure of our research prototype, Palantír, and its events [13]. The 3D visualization gives an overview of all workspaces at the same time, from the perspective of either 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 managers, who can now get a comprehensive view of all project activities and adjust task assignments if needed, and also developers, who can benefit from the high level view and place their work in the context of others' activities. 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 one 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 between the known check-outs and check-ins. While this necessarily represents a limitation, it allows us to make interesting observations regarding the evolution of real projects in real settings. Our next work, obviously, is aimed at 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 between check-ins.

The remainder of this paper is organized as follows. Section 2 discusses the approach we took to develop the activity viewer visualization. Next, in Section 3, we discuss the infrastructure on which we have based this work. Section 4 delves into the implementation of our workspace activity viewer. In Section 5 we demonstrate two of the projects that our tool has visualized. Finally, we conclude in Section 6 with an outlook at future work.

2. Approach

Maletic, et al. [14] categorize software visualizations via five dimensions: tasks, audience, target, representation, and medium. Our goal for the activity viewer visualization was to be as malleable in each of those categories as possible, so that the same visualization can be used in a variety of situations.

The *tasks* of the visualization are three-fold. First, the visualization should serve as a passive awareness widget for a developer's or manager's desktop. Second, the visualization should allow for analysis of the present state of the system's development. Third, the visualization should allow for the history and evolution of the system to be analyzed after the fact.

The *audience* of the visualization is everyone involved with implementing a software project, from coders to managers. We feel that having the same visualization for each stratum of the organization would enable everyone involved in the project to gain a shared understanding, as well as the ability to use the visualization as a communication aid.

The *target* of the visualization is either workspace data captured by awareness tools such as Palantír or data mined from a software configuration management repository.

The *medium* of the visualization ranges from a small window in the corner of a developer's or manager's desktop, to an entire monitor, to a wall-sized display. Again, by having the same visualization usable across a range of mediums maintains continuity as interaction modes change.

The *representation* of the visualization should take advantage of basic metaphors that are easily processed by humans. This led us to choose a three-dimensional representation, since a person can look over a cityscape from a hill, and quickly pick out places of note, skyscrapers, civic center, cathedrals, even though there is an extreme amount of visual information presented to them. From that same hill, the person can use binoculars to see the faces of individuals conversing in the marketplace. Our visualization consists of stacks of cylinders: tall stacks arranged front-and-center rightly command attention, while short stacks in the distance are likely to be of little importance. Similar 3D spatial layouts have been successfully employed for other uses, such as Robertson, et al.'s Data Mountain for web browser bookmarks [15].

3. 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], [13]); 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 software configuration management (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, save, local delete), and then transmits that information to other "remote" workspaces via Palantír events.

4. 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 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 limitation of current Palantír visualizations, with respect to our objectives in this paper, is that they only present the current view of project activities; they do not depict any project evolution based on the past 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 are changing concurrently, who is responsible for major code changes) to better manage their project and teams. As long check-out/check-in cycles hide potential conflicts, we believe a project management view that visualizes the history and project dynamics will help managers in making these kinds of decisions.

We have built a workspace activity viewer on top of Palantír's infrastructure 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, as shown in Figure 1. 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 Figure 1, 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 "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 Figure 1, 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 Figure 2), whereas a single large cylinder signifies a one-time large change to an artifact (center workspace in Figure 2).

The view can be configured in a number of ways to

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, or the number of external references to methods whose implementation has changed.

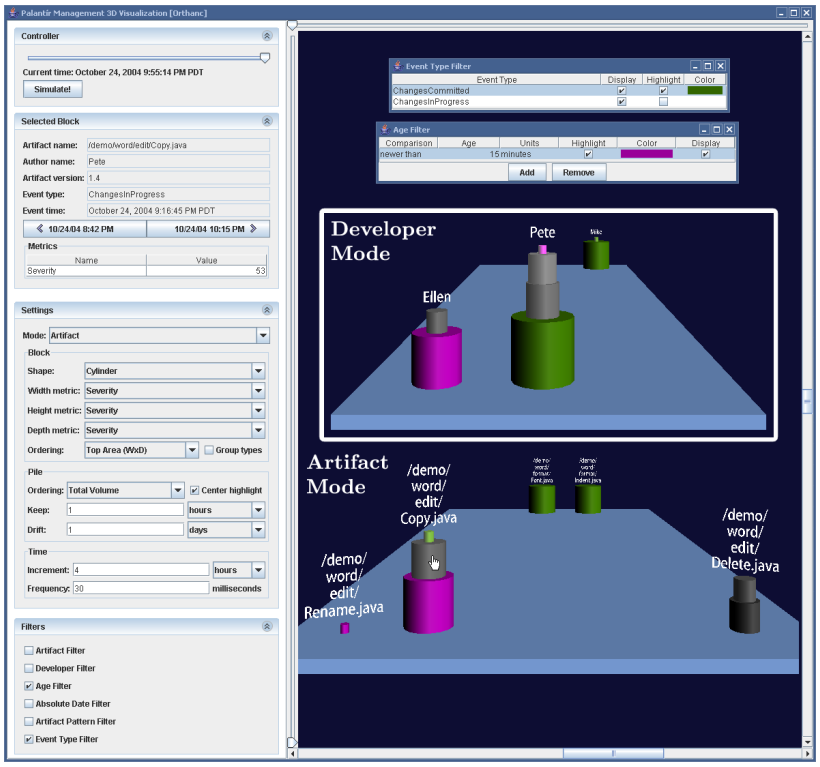


Figure 1. 3D Visualization in Artifact Mode; Developer Mode inset.

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).

4.1.1. 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 Figure 2 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 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 (Section 5).

4.1.2. Rotation. A drawback of the aforementioned view is that, as changes age, 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 Figure 3). This view represents recency of changes based on the relative position of the stacks and concurrency

Table 1. Filters.

Artifact	The artifact filter allows the user to hide or highlight particular artifacts via 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 allows the user to hide or highlight particular developers by selecting them from a list of all the 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., July 7, 2007 10:30 AM). The filter applies to individual cylinders.
Artifact Pattern	The artifact pattern filter is similar to the artifact filter, but instead of selecting from a tree view of all the artifacts, the user is able to enter regular expressions to match against artifact names. This allows the user to hide 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.
Parallelism	The parallelism filter allows the user to highlight artifact cylinders that are on the stacks of multiple developers, or hide those that are not. This filter applies to individual cylinders in developer mode and entire stacks in artifact mode.

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 Figure 3, 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. Awareness Widget

The visualization can be run as an awareness widget in the corner of a developer's or manager's desktop—or on a second display—so that they can passively track the state of the project they are working on, and notice when major activity is occurring. As shown in Figure 4(a), the large stack in the center front of the visualization represents an artifact being worked on by multiple developers who are all making large changes to it. The small stacks that surround it, while

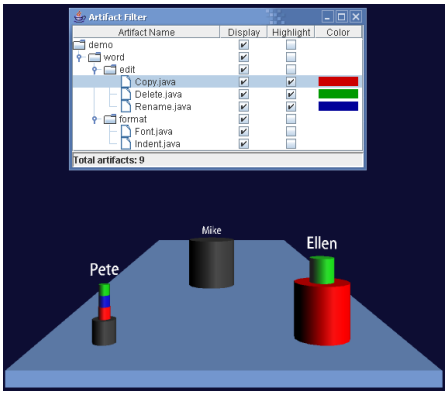


Figure 2. Development styles with artifact filter.

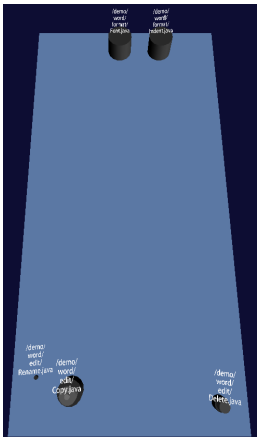


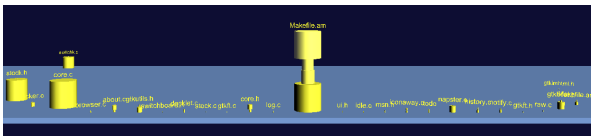
Figure 3. Overhead view.

they also represent parallel work, represent changes of much smaller magnitude. Figure 4(b) shows the same project after the changes being made to that artifact have been committed. The stack is now on the far left and is drifting toward the back of the visualization as the artifact becomes dormant.

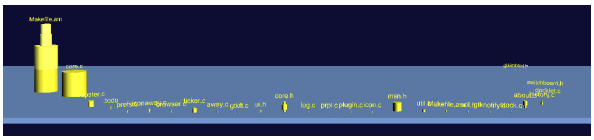
4.3. Tracing Project Evolution

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.

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



(a) Actively developed artifact with large changes is positioned in the center front.



(b) Artifact moves to side (left) and toward the rear as changes are committed and activity stops.

Figure 4. Visualization as awareness widget.

trends, problems, and other patterns of interest. This playback can either be 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.

When being used to analyze project evolution, running the visualization on multiple large high-resolution displays, or even wall-sized displays, may be necessary. As will be shown later in Figure 6, there can be many stacks being shown simultaneously, especially when using the visualization in artifact mode.

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 Sections 5.1 and 5.2, respectively. The only data 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 visualized.

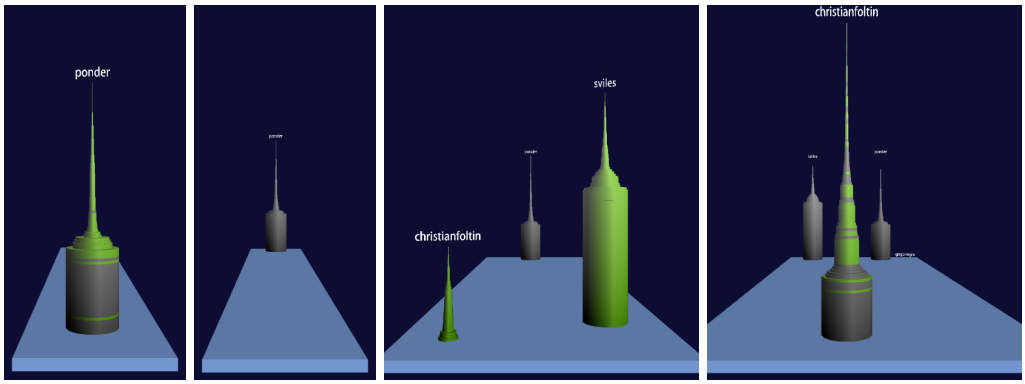
Project Name	Developers	Artifacts	Events
ArgoUML [16]	15	1635	483,026
FreeMind [17]	5	637	151,490
Gaim [18]	22	1861	1,180,192
jEdit [19]	8	1431	450,732
Scarab [20]	37	2305	884,030

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 [17] 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 previous 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 Figure 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 Figure 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 Figure 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 correlate these observations to the "authors and contributors"



(a) April 2001 (b) March 2003 (c) November 2003 (d) March 2004

Figure 5. Evolution of the FreeMind project.

section of the FreeMind website which documents who worked on the project for each version [17].

5.2. Gaim

Gaim [18] is a multi-protocol instant messaging client. Figure 6 shows the Gaim project in May 2005. An artifact pattern filter is highlighting C header files (`.h`) in green, C source files (`.c`) in red, and graphics (primarily pixmaps: `.xpm`) in yellow. Figure 6(a) has no additional filters, and is fundamentally unusable due to the vast amount of artifact piles being shown—the visualization even exceeds the bounds of the display. In Figure 6(b), we add an age filter to remove everything that has been stagnant for two months. This starts to become useful, but noticing parallel work is still difficult. Therefore, in Figure 6(c), we add the parallelism filter, only showing artifacts that have been worked on by more than one developer during that time period. There are 34 artifacts involved in parallel work. Even though it is difficult to see in Figure 6(c), each of the artifact stacks consist of at least two developer segments, as in the zoomed insets. If you look closely you can even see a recurrent pattern of a small change accompanying a larger one. These artifacts would be a starting point for a manager looking for duplicated work, or for testers looking for errors manifesting from indirect conflicts.

6. Conclusions and Future Work

In this paper, we presented a prototype of a 3D visualization for workspace activity. Our work contributes a visualization that, unlike previous work focused on visualizing the state of repositories, focuses exclusively on visualizing the state of workspaces. 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.

Our future work will address three concerns: scalability, utility, and actual use. With respect to scalability, we will migrate the activity viewer to a small tiled display so we are able to visualize large projects in their entirety. For utility, we will update the filters to match sequences of events that occur over a period of time, instead of only single events and immediate state. With respect to actual use, we are in the process of hooking the activity viewer into the workspace events of a real distributed development environment.

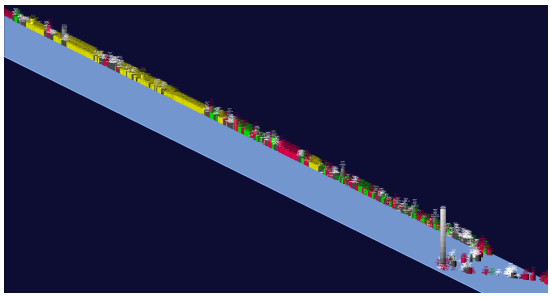
Acknowledgments

Effort partially funded by the National Science Foundation under grant numbers CCR-0093489, IIS-0205724, and IIS-0534775. Effort also supported by an IBM Eclipse Innovation grant and an IBM Technology Fellowship.

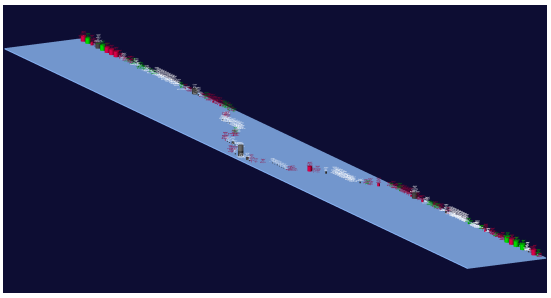
References

[1] B. Berliner, "CVS II: Parallelizing software development," in *USENIX Winter 1990 Technical Conference*, 1990, pp. 341–352.

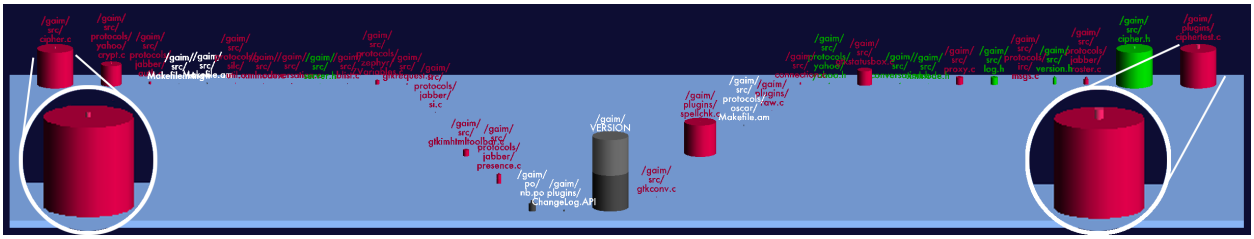
[2] B. Magnusson and U. Asklund, "Fine grained version control of configurations in COOP/Orm," in *Sixth International Workshop on Software Configuration Management*, vol. 1167, 1996, pp. 31–48.



(a) Artifact pattern filter.



(b) Artifact pattern filter & age filter.



(c) Artifact pattern filter, age filter & parallelism filter.

Figure 6. Parallel work in the Gaim project.

[3] P. Molli, H. Skaf-Molli, and C. Bouthier, "State treemap: an awareness widget for multi-synchronous groupware," in *International Workshop on Groupware*, 2001.

[4] L.-T. Cheng, S. Hupfer, S. Ross, and J. Patterson, "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, pp. 102–103.

[5] A. Sarma, Z. Noroozi, and A. van der Hoek, "Palantír: Raising awareness among configuration management workspaces," in *25th International Conference on Software Engineering*, Portland, Oregon, 2003, pp. 444–454.

[6] J. Froehlich and P. Dourish, "Unifying artifacts and activities in a visual tool for distributed software development teams," in *26th International Conference on Software Engineering*, Edinburgh, United Kingdom, 2004, pp. 387–396.

[7] L. Voinea, A. Telea, and J. J. van Wijk, "CVSscan: visualization of code evolution," in *SoftVis '05: Proceedings of the 2005 ACM symposium on Software visualization*, 2005, pp. 47–56.

[8] S. G. Eick, J. L. Steffen, and E. E. Summer, Jr., "Seesoft—a tool for visualizing line oriented software statistics," *IEEE TSE, Special issue on software measurement principles, techniques, and environments*, vol. 18, no. 11, pp. 957–958, 1992.

[9] J. Wu, R. C. Holt, and A. E. Hassan, "Exploring software evolution using spectrographs," in *WCRE '04: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04)*, Washington, DC, 2004, pp. 80–89.

[10] E. Trainer, S. Quirk, C. de Souza, and D. Redmiles, "Bridging the gap between technical and social dependencies with Ariadne," in *eclipse '05: Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*, 2005, pp. 26–30.

[11] M. Lanza, "The evolution matrix: Recovering software evolution using software visualization techniques," in *2001 International Workshop on the Principles of Software Evolution*, 2001, pp. 28–33.

[12] M. J. Baker and S. G. Eick, "Space-filling software visualization," *Journal of Visual Languages and Computing*, vol. 6, no. 2, pp. 119–133, 1995.

[13] A. Sarma and A. van der Hoek, "Visualizing parallel workspace activities," in *IASTED International Conference on Software Engineering and Applications (SEA)*, Marina Del Rey, California, 2003, pp. 435–440.

[14] J. I. Maletic, A. Marcus, and M. L. Collard, "A task oriented view of software visualization," in *VISSOFT '02: Proceedings of the 1st International Workshop on Visualizing Software for Understanding and Analysis*, 2002, pp. 32–40.

[15] G. Robertson, M. Czerwinski, K. Larson, D. C. Robbins, D. Thiel, and M. van Dantzich, "Data mountain: using spatial memory for document management," in *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, 1998, pp. 153–162.

[16] Tigris.org, "ArgoUML: Project home," http://argouml.tigris.org/, 2005.

[17] FreeMind Project, "FreeMind: free mind mapping software," http://freemind.sourceforge.net/, 2005.

[18] Gaim Project, "Gaim: A multi-protocol instant messaging (IM) client," http://gaim.sourceforge.net/, 2005.

[19] jEdit Project, "jEdit: Programmer's text editor," http://www.jedit.org/, 2005.

[20] Tigris.org, "Scarab: Project home," http://scarab.tigris.org/, 2005.