VISUALIZING PARALLEL WORKSPACE ACTIVITIES

Anita Sarma and André van der Hoek:
School of Information and Computer Science
Department of Informatics
University of California, Irvine
Irvine, CA 92697-3425 USA
{asarma,andre}@ics.uci.edu

Abstract

Palantír is a configuration management workspace awareness tool that continuously informs developers of the changes that are made in parallel by other developers in other workspaces. In order to achieve its goal of reducing the number of merge conflicts when developers commit their artifacts, Palantír deliberately breaks traditional workspace isolation in order to promote better coordination of parallel activities. In this paper we examine four different visualizations that developers can use for visualizing the activities in other workspaces. We discuss their strengths and weaknesses, role within Palantír, and opportunities for future improvements.

Key Words
Configuration management, cooperative work support, software development, workspace visualization

1. Introduction

At its core, a configuration management system is geared towards helping developers coordinate their activities. Most configuration management systems do so by separating the overall development effort over multiple, isolated workspaces, such that one developer's changes do not affect those of another developer. This kind of isolation works fine when work is divided among developers in a mutually exclusive manner. In that (ideal) case, there will be no conflict when a developer places their changes back in the central repository. Unfortunately, the ideal case can normally not be achieved and consequently changes made by different developers in different workspaces regularly conflict with each other [7,8,13]. These conflicting changes lead to integration problems that often must be manually resolved [10].

Palantír is a configuration management workspace awareness tool that is based on the hypothesis that being aware of each other's workspace activities enables developers to better coordinate their parallel changes and lessen the number of conflicts that will occur. In effect, Palantír is based on the premise of early detection: rather than discovering that two changes are in conflict at the moment of committing the second change, Palantír helps developers uncover potential conflicts as they are making their changes in their respective workspaces. Palantír shares this goal with several other research [1,4,5,9,11,12], but distinguishes itself by: (a) being a generic infrastructure that can plug into any configuration management system, and (b) providing a rich set of information that can be displayed in a variety of different ways. To demonstrate the first point, Palantír has been integrated with RCS [15], CVS [2], and Subversion [16]. The second point is highlighted by the fact that Palantír's different visualizations not only inform a developer of which other developers change which other artifacts, but do so in a pair-wise fashion while presenting additional, detailed information such as a measure of change severity.

In our previous work [14], we described Palantír in terms of its goals, architecture, and implementation, and discussed our experience in integrating Palantír with two different configuration management systems. Furthermore, we showed how Palantír addresses such concerns as scalability, flexibility, and configurability. One of the observations resulting from the work, though, is that different users prefer different kinds of visualizations that present them with different kinds and amounts of information. While Palantír's architecture supports the incorporation of many different kinds of visualizations, we had only prototyped two such visualizations. In this paper, we describe our ongoing work in further refining those two existing visualizations as well as two new visualizations that we have developed in response to a few brief user interviews.

The remainder of the paper is organized as follows. In Section 2, we briefly reiterate the overall architecture and approach of Palantír. Section 3 introduces our four visualizations and discusses their strengths and weaknesses. We conclude in Section 4 with an outlook at our future work.

2. Approach

Palantír itself is not a configuration management system. Rather, it complements and does not interfere with existing configuration management systems by only focusing on collecting, distributing, organizing, and presenting relevant workspace information.

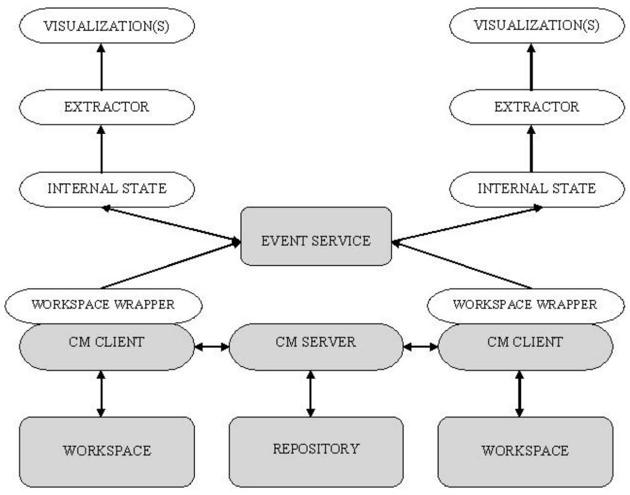


Figure 1. Palantír Architecture.

Figure 1 presents the architecture of Palantír. The bottom six grey components represent components traditionally found in configuration management systems; they are used unchanged. The middle grey component represents Siena, the event notification service [3], which Palantír uses to broadcast and filter events pertaining to activities in other workspaces. The white components are Palantír components that incrementally implement its functionality. Arrows represent information flow. We briefly discuss the role and functionality of each of the Palantír components in the following.

- A *workspace wrapper* intercepts relevant workspace activities and emits events regarding their occurrences (e.g., artifact is placed in workspace, artifact has undergone changes, artifact is placed back in repository). Workspaces and their access mechanisms differ per configuration management system. Therefore, each wrapper is built for one specific configuration management system and translates its particular workspace conventions to standard Palantír events.
- An *internal state* component collects, preprocesses, and stores all the relevant events from both the local workspace and the remote workspaces, thereby creating an organized overview of workspace activities. In order not to maintain a view of all workspaces, the component leverages information regarding artifacts in the local workspace to only subscribe to Siena events pertaining to workspaces that also operate on those artifacts. This significantly reduces the number of events that are received, thereby reducing the eventual cognitive burden on the user.
- An *extractor* allows a developer to select a subset of events that they may want to view. Often, human knowledge can help in deciding what may and may not be relevant events (e.g., it may have already been agreed upon that developer Joe will not make any changes to certain artifacts). Palantír supports such human input by providing an extensive selection mechanism through which a user may exercise their preferences and describe in which events they are interested (e.g., based on type of events, author, time period, and/or severity).
- A *visualization* is responsible for organizing and displaying the activities as they occur in parallel in different workspaces. Only events that pass the extractor are visualized. We have developed four visualizations thus far – a ticker tape visualization, a tabular visualization, an explorer visualization, and a fully graphical visualization.

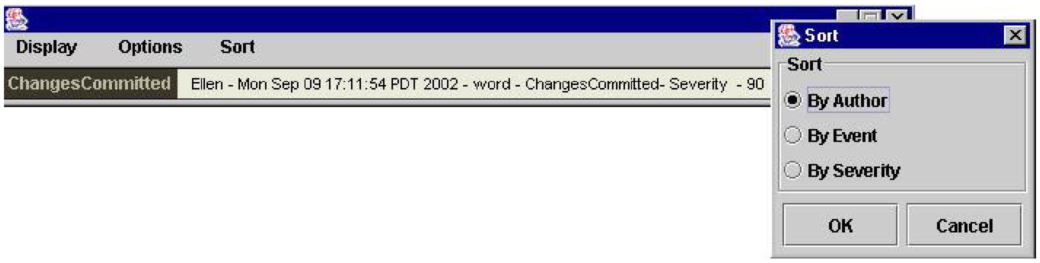
Using the above components, Palantír continuously shares information with developers regarding the activities of other developers. Rather than developers having to proactively obtain limited information directly from the repository, Palantír automatically provides them with workspace awareness in the form of an accurate, complete, and always up-to-date picture of the ongoing activities in other workspaces.

3. Visualizations

The architecture of Palantír is purposely designed to support multiple visualizations. Different users typically have different desires and ways of working, which should be accommodated by giving them the option to operate with a view that supports their working style. Thus far, we have developed four different visualizations: a ticker tape visualization, a tabular visualization, an explorer visualization, and a fully graphical visualization. We discuss each of these visualizations below.

3.1 Ticker Tape Visualization

The first visualization is a simple scrolling marquee that is similar to the one provided by Elvin [6]. Shown in Figure 2, the ticker tape scrolls one-by-one through the set of events as they occur in each of the workspaces. Events can be sorted as desired, per author, event type, or severity. Clearly, if too many events scroll by the ticker tape loses its effectiveness. Therefore, the ticker tape best serves in the role of an alert mechanism. Using the extractor component, a developer should set a relatively high threshold for the severity of the events they want to see. As a result, the scrolling marquee will generally be empty but when it is not, a developer is informed of a likely se-



ChangesCommitted | Ellen - Mon Sep 09 17:11:54 PDT 2002 - word - ChangesCommitted- Severity - 90

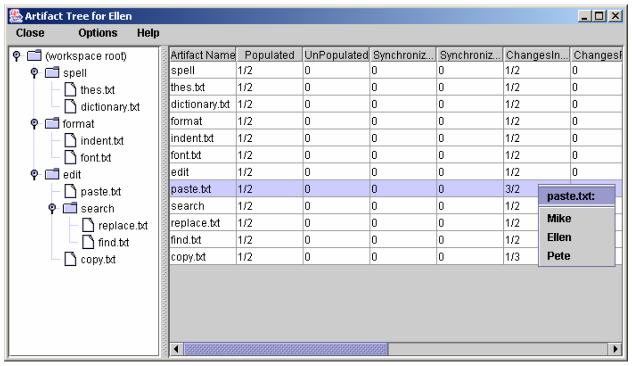
Sort: [✓ By Author] By Event By Severity

Figure 2. Ticker Tape Visualization.

rious situation that warrants further investigation. Any of the other visualizations can then be used to understand the details of that situation.

3.2 Tabular Visualization

Shown in Figure 3 is the second visualization, which presents information in tabular form. In this visualization, the artifacts in the local workspace are shown on the left hand side as organized in an expandable tree. A developer can selectively open those artifacts in which they are interested and view a detailed summary of all relevant activities in all workspaces. For instance, the artifact "paste.txt" is currently present in three workspaces, and has undergone some changes in two of those workspaces. Examining a cell in the table gives detailed information. For instance, "paste.txt" is currently in the workspaces of Mike, Pete, and Ellen. Note that columns can be reordered in order for a developer to have their preferred information presented first.



Artifact Name	Populated	UnPopulated	Synchroniz...	Synchroniz...	ChangesIn...	ChangesI
spell	1/2	0	0	0	1/2	0
thes.txt	1/2	0	0	0	1/2	0
dictionary.txt	1/2	0	0	0	1/2	0
format	1/2	0	0	0	1/2	0
indent.txt	1/2	0	0	0	1/2	0
font.txt	1/2	0	0	0	1/2	0
edit	1/2	0	0	0	1/2	0
paste.txt	1/2	0	0	0	3/2	
search	1/2	0	0	0	1/2	
replace.txt	1/2	0	0	0	1/2	
find.txt	1/2	0	0	0	1/2	
copy.txt	1/2	0	0	0	1/3	

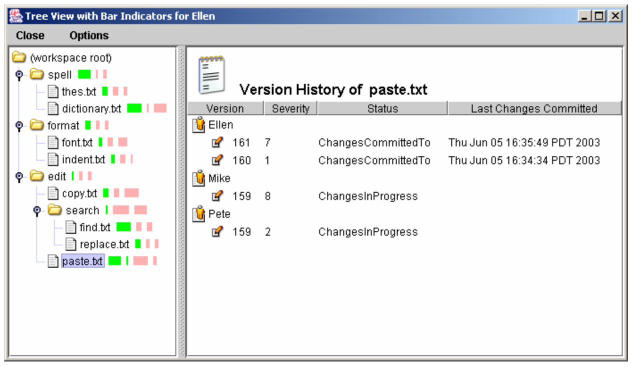
Figure 3. Tabular Visualization.

The tabular view is helpful in providing a cursory glance at the activities in other workspaces. It is more detailed than the ticker tape visualization, and focuses more on providing a cumulative view rather information regarding individual events. As in the ticker tape visualization, if a developer feels the need for further investigation of a particular troublesome situation they can revert to the fully graphical visualization.

We are currently examining several enhancements to this visualization. In particular, we would like to add highlighting to draw attention to particular conditions. For instance, based on a user-defined criterion of highlighting all artifacts with a severity greater than ninety percent, the visualization would underline each artifact that matches that condition and place a red border around its severity field.

3.3 Explorer Visualization

The third visualization is based on the tabular visualization, but instead of using numerical tallies it introduces graphical elements for highlighting the presence of potential conflicts.



Version History of paste.txt

Version	Severity	Status	Last Changes Committed
Ellen			
161	7	ChangesCommittedTo	Thu Jun 05 16:35:49 PDT 2003
160	1	ChangesCommittedTo	Thu Jun 05 16:34:34 PDT 2003
Mike			
159	8	ChangesInProgress	
Pete			
159	2	ChangesInProgress	

Figure 4. Explorer Visualization.

Shown in Figure 4, the left hand side once again is an expandable tree view. In the explorer view, however, the tree view is enhanced with vertical bars indicating the severity of ongoing and committed changes: the longer the bar, the higher the severity of the change. Although not visible in this black and white view, changes are color coded to distinguish changes in a local workspace from changes in other workspaces.

Clicking on the name of an artifact presents the history of that artifact in each of the workspaces. For instance, "paste.txt" has moved through two versions in Ellen's workspace, one in Mike's, and one in Pete's. Note that the status of each version is listed (changes are in progress or changes have been committed), along with a numerical number indicating the severity of a change (the higher the number, the larger the change). This helps a developer in gauging whether they should contact the other developer to enter a discussion in order to avoid future conflicts. Clicking on a single version of an artifact brings up detailed metadata regarding a change.

3.4 Fully Graphical Visualization

The last visualization is a fully graphical visualization that presents a developer with a hierarchical view of an artifact and its constituents (in this case version 115 of the folder "home/word"). Each constituent artifact may itself contain other artifacts and each artifact in the view may exhibit multiple versions (as indicated by stacks of artifacts). The visualization acts like a web browser and lets a user zoom in or zoom out of the hierarchy by double clicking artifacts and pressing a back button, respectively. A user, thus, can monitor the state of other workspaces at a high level and zoom in to explore potential problems that may be present.

Color-coding separates different workspaces. For instance, the stack for the artifact "/home/edit/copy.txt" indicates that Ellen, Pete, and Mike each have a version of the artifact in their workspace. Pete and Mike each have version 115 in their workspace, and their changes are still in progress as indicated by the question mark. Ellen, on the other hand, already has checked in a new version of the artifact (as indicated by the exclamation mark), resulting in her having version 117 in her workspace.

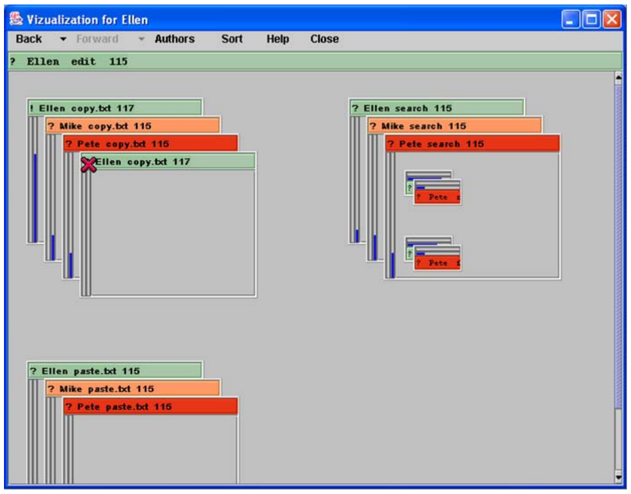


Figure 5. Fully Graphical Visualization.

Artifacts are sorted per their severity, making it easy to quickly spot the largest potential conflicts. Severity is visually indicated by a progress bar: the fuller the bar, the higher the severity. Graphical icons (not shown in the figure) help in highlighting several other types of workspace changes besides a user modifying an artifact. In particular, the icons identify new artifacts, artifacts that have been deleted, and artifacts that have moved from one location in a workspace to another. These kinds of changes usually have quite a bit of impact on the overall project, hence our special treatment.

An important aspect of our visualization is that it shows pair-wise conflicts. Normally, the visualization shows all potential conflicts. By clicking on an artifact being modified in another workspace, however, the view changes to one in which only conflicts between the local workspace and the selected remote workspace are shown. This makes it much easier to locate and understand the impact of potential conflicts.

Figure 6 shows how pair-wise comparisons work from the perspective of Ellen. Normally, the visualization presents Ellen the view shown in the left pane. All potential conflicts are shown among all different workspaces. To know the conflicts with just Pete, Ellen simply clicks on Pete's workspace and is now presented with the middle pane in which only conflicts between Ellen and Pete are shown. Analogously, when Ellen wants to know the conflicts between her and Mike, she simply clicks on Mike's workspace; she is then presented with the pane on the right hand side of Figure 6. Clearly, this mechanism helps in effectively dealing with a large set of workspaces, and can easily highlight the exact nature of a conflict.

4. Conclusions

One of the main functionalities of a configuration management system is to shield developers from the effects of other developers' changes. This, unfortunately, limits the insight that a developer has into the overall ongoing state of a project. To overcome this situation, Palantír continuously shares information regarding the activities taking place in local and remote workspaces. By letting developers choose from a variety of visualizations that inform them of who is modifying which artifacts in parallel, Palantír complements current configuration management system functionality with support for the human identification and intervention of potential conflicts.

Our next steps are to empirically validate Palantír and its visualizations. In particular, we would like to determine the tradeoffs among the various visualizations and understand where they may be improved and what additional information may need to be present for them to be truly effective. One particular dimension we are currently exploring is to add a measure of change impact (determining the impact of a change on a workspace) to our present measure of severity (which determines the size of a change only).

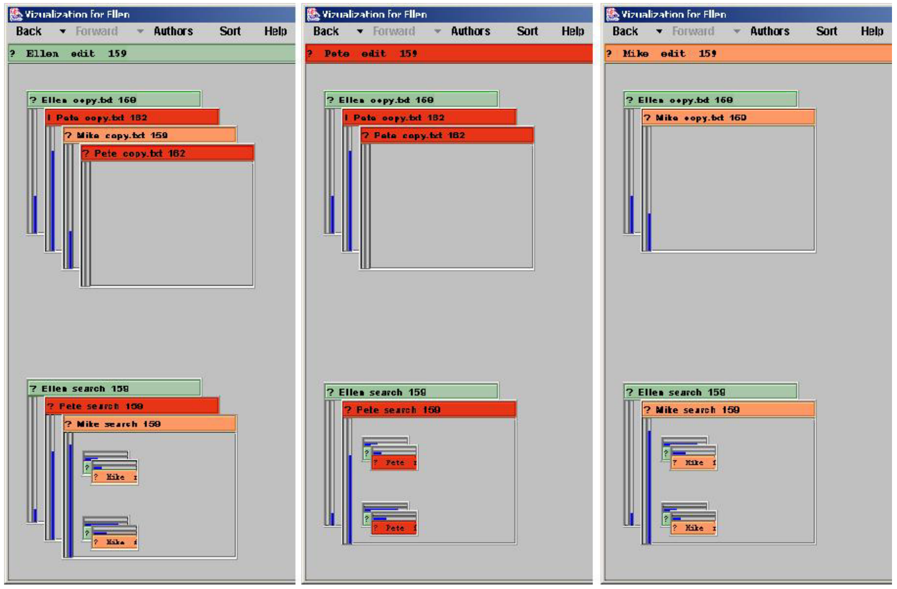


Figure 6. Pair-wise Comparison.

5. Acknowledgement

Effort funded by the National Science Foundation under grant numbers CCR-0093489 and IIS-0205724. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the National Science Foundation.

References

[1] W. Appelt. *WWW Based Collaboration with the BSCW System*. Proceedings of the Conference on Current Trends in Theory and Informatics, 1999: p. 66-78.

[2] B. Berliner. *CVS II: Parallelizing Software Development*. Proceedings of the USENIX Winter 1990 Technical Conference, 1990: p. 341-352.

[3] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, *Design and Evaluation of a Wide-Area Event Notification Service.* ACM Transactions on Computer Systems, 2001.

[4] M.C. Chu-Carroll and S. Sprenkle. *Coven: Brewing Better Collaboration through Software Configuration Management*. Proceedings of the Eighth International Symposium on Foundations of Software Engineering, 2000: p. 88-97.

[5] C.R.B. De Souza, S.D. Basaveswara, and D.F. Redmiles. *Supporting Global Software Development with Event Notification Servers*. Proceedings of the ICSE 2002 International Workshop on Global Software Development, 2002.

[6] G. Fitzpatrick, et al. *Augmenting the Workaday World with Elvin*. Proceedings of the Sixth European Conference on Computer Supported Cooperative Work, 1999: p. 431-451.

[7] R.E. Grinter. *Using a Configuration Management Tool to Coordinate Software Development*. Proceedings of the Conference on Organizational Computing Systems, 1995: p. 168-177.

[8] R.E. Grinter, *Supporting Articulation Work Using Software Configuration Management Systems.* Computer Supported Cooperative Work, 1996. 5(4): p. 447-465.

[9] C. Gutwin and S. Greenberg. *Workspace Awareness for Groupware*. Proceedings of the CHI'96 Conference Companion on Human Factors in Computing Systems, 1996: p. 208-209.

[10] M. Lanza. *The Evolution Matrix: Recovering Software Evolution using Software Visualization Techniques*. Proceedings of the 2001 International Workshop on the Principles of Software Evolution, 2001: p. 28-33.

[11] P. Molli, H. Skaf-Molli, and C. Bouthier. *State Treemap: an Awareness Widget for Multi-Synchronous Groupware*. Proceedings of the Seventh International Workshop on Groupware, 2001.

[12] P. Molli, H. Skaf-Molli, and G. Oster. *Divergence Awareness for Virtual Team through the Web*. Proceedings of the Integrated Design and Process Technology, 2002.

[13] D.E. Perry, H.P. Siy, and L.G. Votta, *Parallel Changes in Large-Scale Software Development: An Observational Case Study.* ACM Transactions on Software Engineering and Methodology, 2001. 10(3): p. 308-337.

[14] A. Sarma, Z. Noroozi, and A. van der Hoek. *Palantír: Raising Awareness among Configura-*

tion Management Workspaces. Proceedings of the Twentyfifth International Conference on Software Engineering, 2003.

[15] W.F. Tichy, *RCS, A System for Version Control.* Software - Practice and Experience, 1985. 15(7): p. 637-654.

[16] Tigris.org, *Subversion*, http://subversion.tigris.org/, 2002.