Integrating Palantír and Eclipse: Workspace Awareness in Application Development

Roger M. Ripley, Ryan Y. Yasui, Anita Sarma, 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, ryasui, asarma, andre\}@ics.uci.edu

October 24, 2004

Work supported by a 2004 IBM Eclipse Innovation Grant.
A Typical CM Scenario

Each workspace contains local copies of artifacts stored in the repository.
Direct Conflicts

A Typical CM Scenario

Pete’s Workspace

Ellen’s Workspace

CM Repository

Conflicting changes to the *same* artifact.

Even though CM systems can detect direct conflicts, at the point a merge fails, they need to be manually resolved.
Indirect Conflicts

A Typical CM Scenario

Pete’s Workspace

Ellen’s Workspace

CM Repository

Conflicting changes to different artifacts.

Indirect conflicts are not detected by CM systems.
Problem

Current CM systems purposely isolate developers in their workspaces.

- This provides coordination support, but...
- Developers are unaware of parallel activities, leading to direct or indirect conflicts.
- Breaking this “bad” aspect of isolation could solve the problem.
Approach

Providing workspace awareness to users will enable them to detect potential conflicts earlier, as they occur, and thereby avoid them.

Palantír provides workspace awareness by building on top of existing CM facilities and concentrates on the collection, distribution, organization, and presentation of relevant workspace information.

- Independent from the underlying CM system.
- Presents only relevant information using visualizations.
  - Only includes activities relating to local workspace artifacts.
  - Can extract and present only a subset of desired events.
  - Displays the severity (magnitude) of remote changes.
The green bars indicate changes made in the local workspace. The red bars indicate changes made in remote workspaces.
Motivation

- Workspace awareness should be provided as peripheral information as developers do not want to be distracted while they work.

- The standalone Palantir implementation presents the visualizations in separate windows, requiring a definite mental context switch to take advantage of this awareness information.

Some of this burden could be relieved by placing peripheral awareness information *within* an IDE.
Decorator Visualization

Motivation

- Triangle decorators annotate resources in the *Navigator* and *Package Explorer* views.
- The size and darkness denotes the average remote severity.
- The numeric severity is shown to the right of the resource name.

By monitoring the size and color of the triangles, a developer can easily gauge concurrent activities in others’ workspaces at a glance.
Original Visualizations

- The original Palantír visualizations are activated within Eclipse.
- Developers can reference these visualizations when they need more detail than the decorator view affords.
**Workflow**

Palantír originally was based upon a very simple workflow model: check out, make changes, check in.

Eclipse supports a more complex model—the local workspace persists between sessions—which Palantír needed to be adapted to.
The server facilitates persistent client state.

The workspace wrapper bridges Eclipse and Palantír events.
Issues with Eclipse

- There is not a generic Team API (JSR-147), so our integration is specific to CVS—other CM systems would require changes.
- Integrating Swing code within SWT was not possible before Eclipse 3.0; this would require the existing visualizations to have been rewritten to appear within the IDE.
- The decorator facility only allows unformatted text to be added to the label.
  - We would have liked to add a graphical bar that would grow and shrink, or been able to change the color.
Conclusion

- Goal: integrate Palantír’s workspace awareness into an IDE to lessen the amount of mental context switching required.

- The plug-in architecture of Eclipse allowed for a mostly smooth integration, which was prototyped in approximately six months.

- Some of the issues, such as integrating Swing code within SWT, are being or have been addressed by the Eclipse project.

- The lack of a generic repository API was the greatest shortfall we encountered with Eclipse. (JSR-147 should address this.)

- Palantír has been initially deployed for feedback and testing. It is being used by three class projects at UCI and two outside groups at other educational institutions.


**Future work**

- Development and implementation of a change impact metric.
- Integration with Subclipse (Eclipse Subversion plug-in).
- Three-dimensional workspace activity monitor.
- Case studies to determine the effectiveness of Palantír.

**Download Palantír at**

http://www.ics.uci.edu/~asarma/Palantir