Tesseract: Interactive Environment for Exploration of Project Relationships

Larry Maccherone, Anita Sarma, Patrick Wagstrom, and Jim Herbsleb
Institute for Software Research - Carnegie Mellon University
{LMaccherone, asarma, pwagstro, herbsleb}@cmu.edu

Abstract

Important insights can be gained by exploring the connections between project entities as recorded in the siloed databases maintained by software projects. We have developed Tesseract, a software archive browser that utilizes cross-linked displays to enable visual exploration of relationships between artifacts, developers, issues, and project communications.

1. Tesseract

Each piece of data in a software project's archive is gathered for a particular purpose which is largely satisfied by being able to recall that textual record at a later date. For instance, an issue database is maintained so that a list of outstanding issues can be displayed and developers who make progress on an issue can record those status changes.

However, more questions can be answered by exploring the connections between project entities as well as understanding the strength and relative timing of those connections [3]. For instance, it may be beneficial for a developer embarking on a new task to identify the artifacts that were modified or created when a similar feature request was accomplished. She might also want to know who has been editing these files the most in the last month or with whom those original developers communicated frequently when the original feature request was implemented [1].

Furthermore, functionalities such as cross-linking, aggregation and temporal filtering can help achieve much more nuanced understanding of a project. For example, artifacts that are frequently committed together signal dependencies among artifacts. Similarly, communication records can be used to build a social network and this can be compared to the one implied by the artifact dependency network [2]. Gaining such nuanced understanding is difficult (at best) when using the standard interfaces for these project archives.

Tesseract analyzes different project archives, such as change management systems, issue repositories, and communication records to determine the interrelationships, which are then graphically displayed via four juxtaposed panes, enabling users to easily explore the data set. Tesseract also uses the analysis technique proposed by Cataldo et al. [2] to show the user where the set of technical dependencies matches the social interactions, and where it does not. The four display panes of the Tesseract User Interface (Figure 1) are:

Figure 1. Tesseract user interface

1. **The Project Activity pane** displays the overall activities (commits on the top, communication on the bottom) in a time series display. It allows users to select a time period for their investigation, which then acts as a filter for all the other panes.
2. **The Files pane** links files that are frequently changed together, which is a surprisingly good heuristic for determining artifact interdependency [2] that works even in situations where code analysis fails (different programming languages, non-call-graph dependencies, etc.). The number of times two files are committed together is represented by the thickness of the edges in the network. A textual listing of the file names is provided to allow quick identification of specific files by name.

3. **The Developers pane** displays relationships among developers. Two developers are expected to communicate if they edit interdependent artifacts often enough (thresholded). The edges in this network are colored red when developers who are expected to have communicated failed to do so. For grey (communication without expectation) or green edges (communication coinciding with expectation), the thickness of the edges is derived from the number of times developers communicated. Similar to the file network, a textual listing of the developer names is provided.
4. **The Issues pane** displays defect or feature related information as a stacked area chart as well as in a detailed listing.

The critical advantage of Tesseract's approach is that it enables the interactive exploration of the different connections among different project entities. Users can also change the perspective of their investigation by drilling down on specific artifact(s) and developer(s). For instance, a user might drill-down to only the developers he personally knows in the 'Developers' pane to find whether any of his acquaintances have expertise which would help with his current task. Other user actions to facilitate investigations include: (1) clicking on an entity to highlight, in yellow, all related entities in the other panes, (2) hovering over a node to display additional information about the node and highlight any other nodes with an edge to the hovered over node, (3) panning zooming, and moving individual nodes and (4) searching to quickly find an entity when they have some partial information.

2. Usage scenario

Tesseract can be used in two completely different modes: (1) to gain "retrospective**"** insight into development patterns, or (2) to gain "prospective" insight for current work and planning. Here, we will discuss one "retrospective" insight usage. Additional "retrospective" and "prospective" scenarios are scripted for live demonstration.

Usage scenario – Patterns (retrospective):
Figure 2 provides two snapshots of project history, where there were high bursts of activity. We can make the following observations from Figure 2(a): (1) Stephen Walther is the primary contributor having changed literally every file; (2) Stephen is central and in contact with most other developers (green lines between Stephen and other developers), but very few developers are communicating among themselves (red lines); (3) the file network is densely connected indicating a high degree of coupling; and (4) this time period shows a continuously increasing list of open issues.

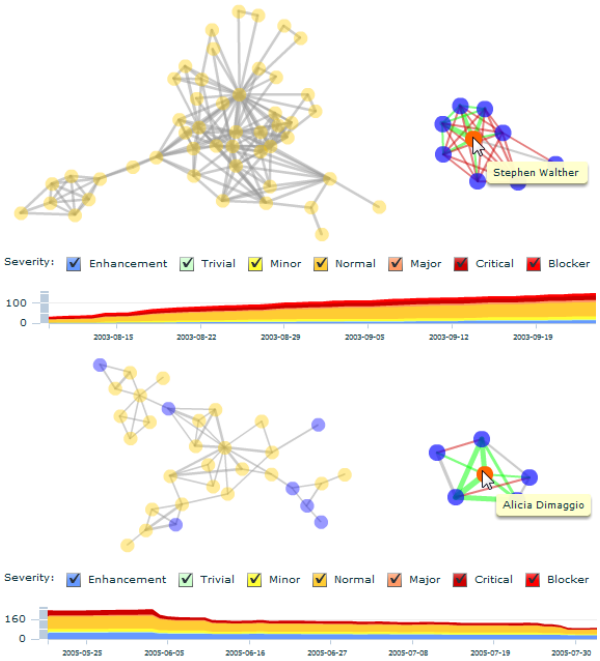

Figure 2. Comparing two time periods

When we investigate the later time period, Figure 2(b), we see very different patterns in communication, artifact dependency, and the trend for open issues under the leadership of Alicia. These contrasting patterns do not necessarily imply any causal relationships, but certainly provide interesting insights into the project that merits further investigation. Readers can try the demo available at http://crc.maccherone.com/tesseract

3. Acknowledgements

This effort is partially funded by the NSF grant number IIS-0414698, IIS 0534656, and the Software Industry Center and its sponsors, particularly the Alfred P. Sloan Foundation. Effort also supported by a 2007 Jazz Faculty Grant.

4. References

[1] Bird, C., et al. 2008. *Chapels in the Bazaar? Latent Social Structure in OSS.* FSE. p. 24-35.

[2] Cataldo, M. and J. Herbsleb. 2008. *Communication Networks in Geographically Distributed Software Development.* CSCW. p. 579-588.

[3] de Souza, C.R.B. and D. Redmiles. 2008. *An Empirical Study of Software Developers' Management of Dependencies and Changes.*ICSE. p. 241-250.