Ligature: Combining node-and-link graph rendering with a timeline for sensemaking in software development repositories

Gina Venolia
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
gina.venolia@microsoft.com

CHI 2006—Social Visualization Workshop, April 22–27, 2006, Montreal, Canada.

Abstract
This paper describes an index over communication artifacts related to software development, and proposes a system for visualizing and browsing that combines aspects of a node-and-link graph rendering with a lifespan timeline, to support sensemaking in root-cause analysis of software development failures.

Introduction
The software development process leaves a vast trail of data behind it documenting the online activity of developers, testers, project managers, and other participants. The vast store of historical artifacts—old emails, bug reports, check-in messages, specifications, etc.—provide valuable, but mostly-untapped, resources for answering crucial questions about the project, such as, *Why was this code written this way? Are there known problems in this code? Why did the build break? Why did this critical bug reach our customers?* Investigating questions like these is sometimes called *sensemaking*, which may be defined as “the process of searching for a representation and encoding data in that representation to answer task-specific questions”

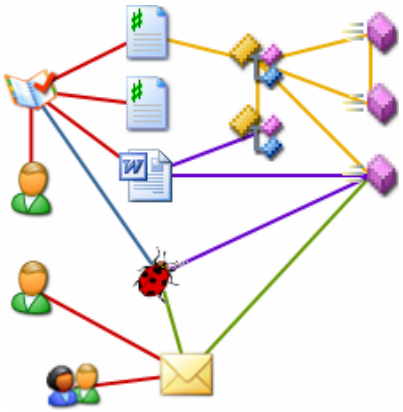

Figure 1. The Bridge index is a graph over software-related artifacts (shown here as a toy example) where the links come from the structure of the source schema (red lines), the structured documents (yellow), plain-text allusions (green), and other sources (blue and purple).

[1]. Sometimes the answers can be found in a single artifact. When this is the case a good search tool over the relevant store could help the user to find the crucial artifact. However it's common that the answer is spread across many artifacts in the data trail. When this is the case the user must explore many artifacts, understand the relationships among them, and piece together the answer from multiple bits of evidence. This suggests that a simple search tool is not enough—what is needed is a sophisticated interface for exploring a collection of related artifacts.

One task that relies heavily on this kind of in-depth exploration of software artifacts is *root-cause analysis*, or *RCA*, which is the process of finding the reasons for critical failures in the software development process. I have interviewed one of the people responsible for this activity at a large software corporation. The output of each analysis that he completes is a report documenting the chain of events that contributed to the failure and suggesting solutions to prevent such failures in the future. The framework that he uses for his sensemaking process is a word-processor document containing a chronological list of artifacts related to the failure, including text snippets from the artifacts and his own annotations interpreting them. There may be hundreds of entries in a typical investigation chronology. He discovers his source material by laboriously searching the repositories containing artifacts of potential interest, searching for keywords and phrases, peoples' names, and the identifying numbers of key bugs, knowledge-base articles, and builds. Each repository has its own unique search interface so this process is quite tedious.

(It should be mentioned here that his process is much like any other historical investigator examining electronic records, whether historian, detective, or lawyer. The general methods he uses are not unique to software or RCA, but are a part of sensemaking in general.)

The goal of the work I present here is to support RCA of failures in the software development process.

The Bridge Index

I have built a system, called the *Bridge*, which creates a single full-text search index over the bug database, the source-code control system, and archives of several key email discussion lists relating to the development of a major computer operating system [2].

The Bridge goes beyond a simple search index in two ways. First, it records directed links between related pairs of artifacts. Link analysis is a crucial part of a modern full-text search system [3]. For World Wide Web search, it's easy to detect the links within HTML documents. For software artifacts it's not so simple—my system uses several complementary means to detect the relationships. Some relationships are explicit in the source schema, e.g. a source-code control system explicitly represents each check-in, the person who performed it, and the files affected; these can be represented in the index as links. Other links come from the contents of structured files, e.g. a source code file might contain a class which derives from another class and contains methods which invoke other methods. Interpersonal communication—bug messages, emails, check-in messages, etc.—often contain allusions to artifacts, e.g. "bug 123456" or "the Account::Add method". which can be detected and

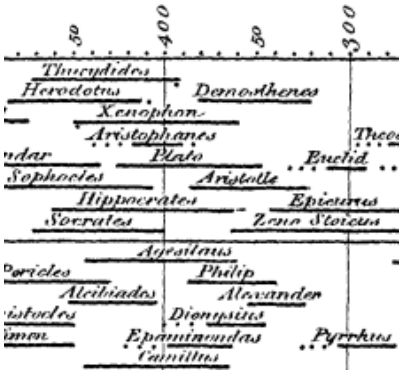

Figure 2. Detail from Priestly's *A Specimen of a Chart of Biography* (1765) showing the lifespans of notable people in classical antiquity [4].

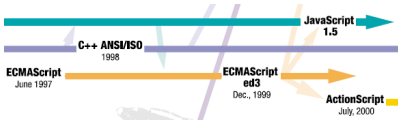

Figure 3. Detail from Lévénez' *History of Programming Languages* (2004) showing lifespans of and influences between programming languages [6].

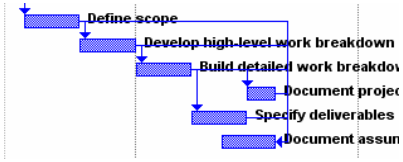

Figure 4. Detail of a view in Microsoft Project showing a Gantt chart with inter-task dependencies.

represented as links. These are the three primary sources of links in the current implementation of the Bridge but others are possible. The artifacts and links together form a graph (see Figure 1). The graph can become huge: the Bridge index representing the bug, check-in, and some email activity related to the development of the Microsoft Windows operating system over July 2005 through January 2006 contains about five million artifacts and ten million links.

The second way the Bridge goes beyond a typical search index is that each artifact and link records a range of dates indicating its lifespan. Virtually all of the source data that contributes to the Bridge index is associated with discrete dates. Each email, check-in, and bug edit is dated, and so the lifespans of artifacts and links created to represent them encompass those dates.

Links and lifespans can help to improve basic search functionality when utilized as terms in the search scoring function. They can also provide rich data for browsing over artifacts and visualizing the relationships among them.

Timelines, Lifespans, and Links

Visualizations of timelines comparing multiple lifespans have a long history (see Figure 2), both as one-off charts and in interactive systems [5]. Timelines allow visual comparison of the lifespans of the items represented. They allow the viewer to eliminate some hypothesized relationships among the items, e.g. Socrates couldn't have met Aristotle because their lifespans didn't overlap, and ActionScript couldn't have inspired the initial development of ECMAScript because ECMAScript came first. Organization by time provides a natural framework for the sensemaking process, and allows narrative to be used as a sensemaking tool.

It's not unusual for lifespan timelines to be augmented with links showing relationships among the items, such as seminal influences among world religions, world languages, art movements, or computer languages (see Figure 3). Gantt charts used in project management are sometimes augmented with links depicting dependencies between tasks (see Figure 4). UML sequence diagrams show communication between computer processes over time. The addition of links further informs the viewer about hypothesized relationships among the items.

Proposed Browser

Inspired by these techniques, I'm proposing a browser on the Bridge index, which I'm calling *Ligature*, which uses a timeline visualization of the lifespans of artifacts and shows links among the artifacts (see Figure 5).

The timeline of artifact lifespans is straightforward, echoing Figure 2. Given the size of the Bridge index only a tiny portion can be shown at once. This suggests that a browser is needed, i.e. an interactive system that allows the user to manipulate a subset of artifacts which will be shown.

The links are somewhat more complicated. Each link has a lifespan so links too should be rendered as a range of dates. This is unduly complicated, so a simpler visual representation is desirable. Intuition suggests that a link's birth is much more salient than its death, so we can render a link as a moment in time, i.e. as a vertical line connecting two artifacts.

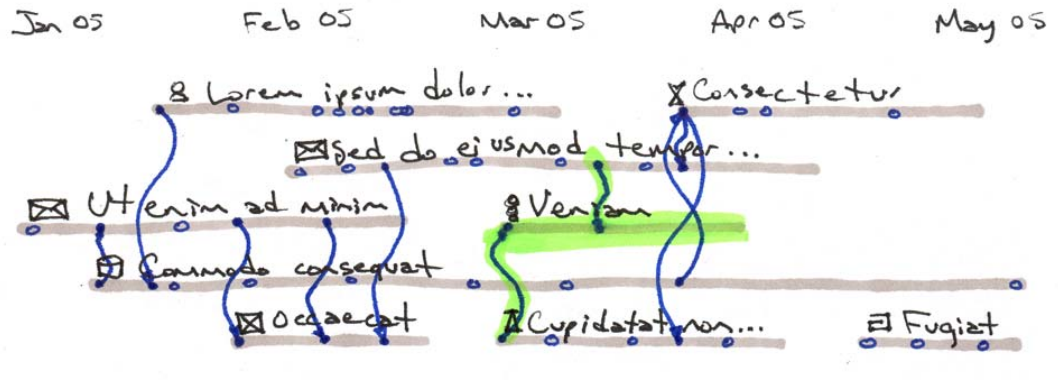

Figure 5. A sketch of the proposed browser. Time flows from left to right as in traditional timelines. Each artifact is shown as a labeled, horizontal line whose extent indicates its lifespan. Each link whose endpoints are both visible is shown as a curved line whose endpoints align with the start of the link's lifespan and whose shape indicates the link's direction. A link with only one visible endpoint is shown as an open circle.

A link is shown as a line only if both artifacts it relates are visible. If only one is visible then a mark may be placed on it to indicate the link. Taken together the marks indicate a pattern of activity for the artifact.

As mentioned before, links are directed. They could be rendered as straight lines with arrows. The resulting overlap, and the necessity for reading the arrowheads, suggests that an alternative representation of link direction may be desirable. By curving the links into an S or reverse S shape, the direction and length of the link becomes apparent—even if only a bit of it is visible. Also the overlapping of links is reduced.

Browsing is the act of adding to or removing from the set of selected artifacts. One way to do that is by following links. It may be reasonable to "expand" an artifact by showing all the artifacts linked to it.

The set of selected artifacts is not merely a transitory thing. The hundreds of items in the RCA investigation chronology are essentially equivalent to the selected set, so it's crucial that the set be persistent and that the user can maintain multiple, independent sets.

To support the complete requirements of software development RCA the browser must include the ability to annotate artifacts and be integrated with a search UI.

Conclusion

This is the barest sketch of a browser to support sensemaking in root-cause analysis for software development failures. Much remains to be done to build and evaluate the system. If it evolves into a useful tool for that task then it may be appropriate to apply it to other sensemaking domains.

The Ligature browser, which combines node-and-link graph rendering with lifespan timelines, may be appropriate for other datasets that associate a date with each link.

Bibliography

[1] Daniel Russell, Mark Stefik, Peter Pirolli, and Stuart Card, "The Cost Structure of Sensemaking" in *Proc. INTERCHI '93*. ACM Press (1993), pp. 269-276.

[2] Gina Venolia, "Bridges between Silos: A Microsoft Research Project" (2005). http://research.microsoft.com/~ginav/bridges-between-silos.doc

[3] Soumen Chakrabarti, *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kauffman (2003), pp. 203-254.

[4] Joseph Priestley, *A Chart of Biography*. London (1765).

[5] Catherine Plaisant, Brett Milash, Anne Rose, Seth Widoff, and Ben Schneiderman, "LifeLines: Visualizing Personal Histories" in *Proc. CHI '96*. ACM Press (1996), pp. 221-227.

[6] Éric Lévénez, *History of Programming Languages*. O'Reilly (2004).