CodeTimeline: Storytelling with Versioning Data

Adrian Kuhn
Software Practices Lab[1]
University of British Columbia

Mirko Stocker
Institute for Software
University of Applied Sciences Rapperswil

***Abstract*—Working with a software system typically requires knowledge of the system's history, however this knowledge is often only tribal memory of the development team. In past user studies we have observed that when being presented with collaboration views and word clouds from the system's history engineers start sharing memories linked to those visualizations. In this paper we propose an approach based on a story-telling visualization, which is designed to entice engineers to share and document their tribal memory. Sticky notes can be used to share memories of a system's lifetime events, such as past design rationales but also more casual memories like pictures from after-work beer or a hackathon. We present an early-stage prototype implementation and include two design studies created using that prototype.**

***Keywords*-Software Visualization; Software Evolution; Humans and Social Aspects; Tools and Environments**

I. Introduction

Understanding and working with a software system typically requires knowledge of the system's history and understanding the rationale of past design decisions. This kind of institutional memory is typically not documented and thus often referred to as tribal knowledge of the development team. While senior team members may rely on their institutional memory to recall past decisions, junior team members cannot do so and even senior team members my not be aware of all tribal knowledge. Industry experience has shown that requiring engineers to maintain more exhaustive documentation has traditionally failed, however new approaches that draw from the experience of social-networking have shown promising documentation efforts. For example, the popular Stackoverflow website uses an incentive-driven point system to draw from the expertise of a public user base[2]. However, these approaches typically require a large and public user base, that is crowd-sourcing. In this paper we propose an approach for the documentation of closed-group systems motivated by story-telling approaches used in in computational journalism. We present the prototype of a dashboard that features a *story-telling visualization* of the system's history which is designed such as to entice team members to record and share their institutional memory.

Story-telling with data is a branch of information visualization that originates from computational journalism. Story-telling visualizations are designed to invite their reader to get engaged with the visualized data by establishing a personal connection between the reader and the presented data [1], [2]. While some story-telling visualizations are purely "read-only," such as CNN's interactive map linking casualties in Iraq to their US hometown[3] (see Figure 1), others are "read-write" and encourage their readers to collect or label data sets, such as The Guardian's crowd-sourcing of expense reports of parliament members[4]. More recently Facebook announced Timeline[5], a story-telling overhaul of the user's profile page which has been designed to charm users to share past lifetime events, such as holidays and family events.

Given our experience with the *Ownership Map* [3] and the *Sourcecloud*[6] visualizations we conjecture that story-telling visualizations has the potential to encourage software engineers to share and document their tribal knowledge. We observed during past user studies that the visualization of code ownership and word clouds from a system's history evokes strong emotional response in software engineers and that it entices them to share their anecdotal knowledge of the system's history. When presented with an ownership visualization, as shown in Figure 2, engineers typically start to immediately share memories about past commits and their rationale, often combined with personal stories about the people involved. When working with code ownership maps we started to refer to it as "programmer's family album" because developer responses were often so emotional.

In this paper we hence propose an interactive pinboard that features a *story-telling visualization* of the system's history which is designed to entice team members to record and share their institutional memory about both past and ongoing events in the system's history. The pinboard either features a color-coded ownership map or a series of word clouds. Users can add sticky notes with past design rationales as well as more casual memories such as photos from an after-work beer or a milestone party. Besides establishing a personal connection between engineers and data minded form a system's history, the other design objective of code timelines is to transition from individual narratives to a centralized repository with the team's shared narrativ.

[1]Work presented in this paper was conceived while the first author was with the SCG lab at University of Bern, Switzerland.

[2]See http://stackoverflow.com

[3]See http://www.cnn.com/SPECIALS/war.casualties/index.html

[4]See http://mps-expenses.guardian.co.uk

[5]See http://facebook.com/timeline/about

[6]See http://misto.ch/tag-cloud-visualization-for-source-code

978-1-4673-1067-3/12/$31.00 © 2012 IEEE

Figure 1. CNN's story-telling map of casualties in the Iraq war. The map links the location of war casualties to the home town of services members and thus entices readers to explore the visualized data set by browsing the casualties relates to the places where they or their friends life.

II. Approach in a Nutshell

As a proof of concept, we present CodeTimeline an early-stage prototype of our pinboard idea. The pinboard features a story-telling data visualization where software engineers can add lifetime events related to the history and evolution of a software system. The pinboard offers two different views that illustrate different dimensions of the system's history. The *collaboration view* visualizes code ownership and collaboration patterns and hence the social history of the system (see Figure 2), while the *sourcecloud flow view* features word clouds with the vocabulary that has been added and removed between releases and hence the vocabulary-related history of the system (see Figure 3).

Each of these views consists of a base layer with the visualization of the data pulled from the system's source code repository, as well as a layer with user-contributed life-time events and a timeline that marks major releases. All user-provided life-time events are either linked to an object of interest on the base visualization (region or commit-bubble on the collaboration view, words on the sourcecloud flow) or to a point in time on the timeline.

Our prototype is implemented as an Eclipse plug-in and (will be) available on Github[7]. Eclipse comes with plug-ins for various revision control systems, giving us convenient access to a project's history and its branches and tags for the timeline. For the sourcecloud visualizations we use Cloudio which is part of Zest, Eclipse's visualization toolkit.

In the following we explain the construction of source-cloud flow view as they are a novel contribution of this work. For the construction of the collaboration view, please refer to our previous work on ownership maps [3].

[7]See http://github.com/misto/Sourcecloud

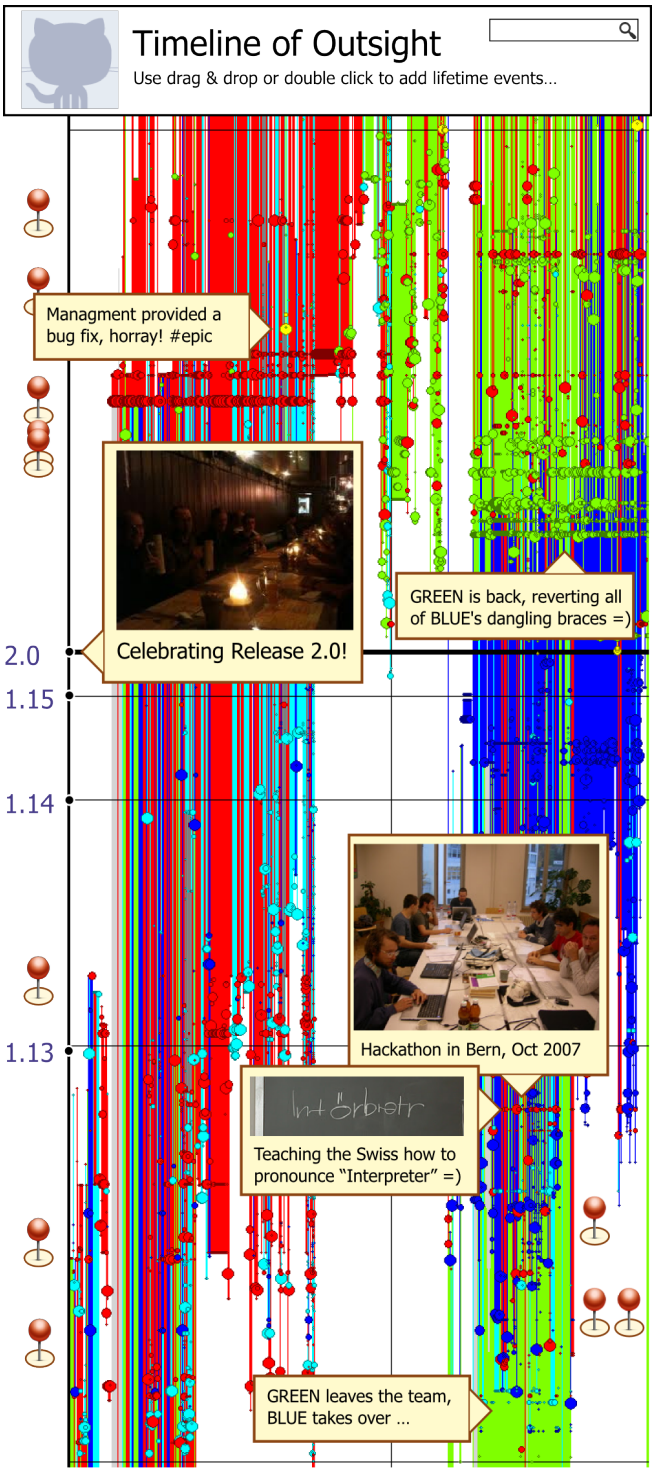

Figure 2. Timeline of Outsight, a closed-source web application. The base layer shows an onwership map, were color indicates developer and lines are the lifeline of files, and bubbles are commit to those by the the developers. The color of a file's lifeline indicates which developer owns most part of it. We manually added sticky notes, based on our memories from an informal meeting where we showed the map to the development. For privacy we replaces usernames with color names and used pictures taken from the personal photo album of the first author.

A. Software Summarization with Word Clouds

Source code contains many terms, as identifiers or in accompanying comments, from both the application domain as well as the technical solution space and can easily be mined by a tool. We extract all terms in the source code and split them by camelcase and remove a list of common stop words. We employ word cloud visualizations to summarize the extracted terms. Word clouds can visualize multiple dimensions of the extracted terms through font size and font color.

We create a separate word cloud for each major release (or other user-definable milestones) of the project, and use color to indicate removed terms in red and emerging terms in blue. The word clouds are arranged along a timeline and hence illustrate how concepts were introduced and removed in past revisions. To detect removed and emerging terms, we use the log-likelihood ratio of the word frequencies [4]. Log-likelihood ratio detects statistically significant trends even for rare terms and hence provides better results than simply counting and comparing the occurrences of terms. A log-likelihood ratio close to zero indicates that the frequency of a term has not significantly changed between two releases, we exclude those terms from the cloud. A high log-likelihood ratio indicates a strong trend in frequency and we add these terms to the cloud, either in red for terms whose frequency decreased or in blue for terms whose frequency increased. We use color saturation to illustrate the strength of detected trends, while font size corresponds to the absolute frequency of the term.

B. Interaction with Code Timeline

Based on the memories invoked by the sourceclouds and the collaboration view, project members can then share and document the project's life-time events from their memory. Users can add sticky notes with their memories about the system's history. Email threads and chat conversations can be dragged-and-dropped onto the pinboard linked to milestones, regions on the ownership map or words in a word cloud. As a future extension, automatically generated lifetime events could be provided by analysis tools, pointing to spikes in code churn or team mutations.

The pinboard is also designed to act as an entry point for code navigation tasks. When a user control-clicks on a word in a sourcecloud, a view shows the differences in the source code that contributed to the selected trending term. On the ownership map of the collaboration view, hovering over a commit-bubble shows author and commit message; control-click takes the user the source file associated with the commit-bubble.

III. Examples of Design Studies

In this section we provide examples of how user-generated timelines might look like. Please bare in mind that we had not yet a chance to run our prototype through a user study

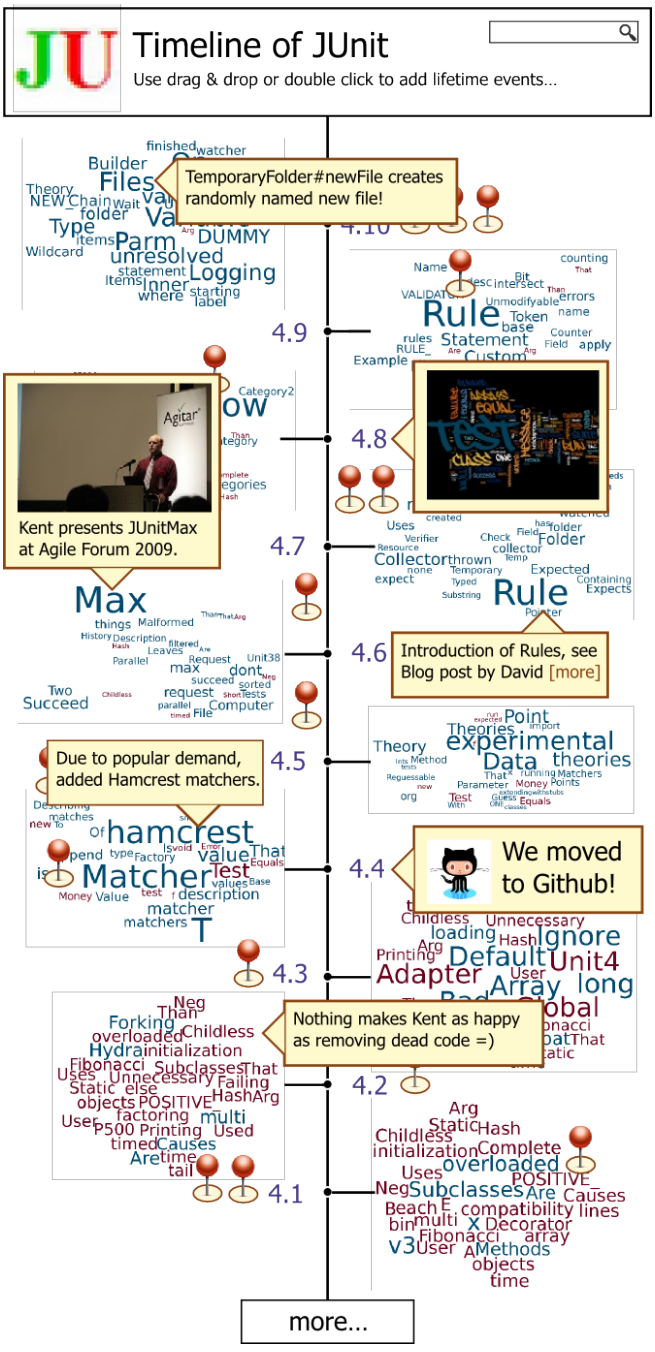

Figure 3. Code timeline of JUnit. Wordclouds are generated by our prototype and show removed and trending terms for each release, most recent release is on top. We added lifetime events that had been manually pulled from the internet to narrate JUnit's evolution. Sticky notes show lifetime events, such as talks, pictures found on the developer's blog and major changes like the move from Sourceforge to Github in version 4.4. Unexpanded lifetime events are shown as a push pin icon. Team members can add more stick notes using drag and drop from the file system or a web browser, or by double clicking a word in a word cloud, or by double clicking anywhere on the timeline.

and hence all examples have been annotated by the authors rather than actual users.

Figure 2 shows a timeline of Outsight, a closed-source web application. The base layer shows an onwership map, were color indicates authors and lines are the lifeline of files, and bubbles are commit to those by the the authors. We manually added sticky notes, based on our memories from when we showed the map to the development in an informal meeting.

Figure 3 shows a timeline of the JUnit project. JUnit is a testing framework for Java. The figure shows the timeline in the sourcecloud flow view. Sourceclouds are generated by our prototype implementations and lifetime events had been manually pulled from the internet to narrate JUnit's evolution. We cross-checked the trend analysis of the sourceflow algorithm with the release notes and found that they correlate very well with the changes in each release.

IV. Related Work

Research on software visualization and mining software repositories has come up with many ways of visualizing and exposing the history of software systems, however to our best knowledge none of those approaches included an interactive or even story-telling element in order to entice engineers to share and document their tribal knowledge. Below we outline work most closest to our proposed approach.

Ogawa presented *Software Evolution Storylines* and *Code Swarm*, two approaches for visualizing the interactions between developers in software project evolution [5]. Storylines draws inspiration from XKCD's "Movie Narrative Charts" and the aesthetic design of metro maps. Both approaches include a strong story-telling element by engaging readers through their personal connection with visualized project and have gained considerable attention from the developer community on Slashdot, Reddit and even Youtube. It is last but not least the popularity of these visualizations that guided the design of our interactive pinboard.

Begel presented Codebook, a social networking website that includes both people and technical artifacts as social actors [6]. Codebook is designed such as to entice developers to start interacting with technical artifacts the same way as they are familiar to interact with their friends on social networking websites such as Facebook, and has hence also fits into the idea of story-telling for user engagement.

V. Concluding Remarks and Outlook

We proposed an approach to entice developers to share and document their tribal knowledge in a central pinbaord. The approach is based on an interactive pinboard using a visualization of their system's history that features strong story-telling elements. The pinboard visualizes code ownership, collaboration and changes to the source vocabulary in order to personally engage software engineers and to entice them to share their memories and narratives about the system's history. We designed the pinboard such that engineers may also share more casual lifetime events of the project, such as pictures from after-work beer or milestone parties, in order to increase user engagement. The design of the pinboard draws from our experience with previous work, where we observed that team members start sharing memories when being presented a visualization of collaboration or word clouds from a system's history, as well as from the popularity of Ogawa's story-telling visualizations among the public developer community. As a proof of concept, we presented CodeTimeline a early-stage prototype implementation of this pinboard, and showed example mock-ups of how user generated timelines might look like. We expect the design of the prototype to adapt considerably based on feedback from first user studies and are looking forward to evaluating our approach by observing teams using our story-telling pinboard in a longitudinal, i.e. long-term study.

Acknowledgment

The authors would like to thank Erwann Wernli for his review and helpful comments on a draft of this paper.

References

[1] E. Segel and J. Heer, "Narrative Visualization: Telling Stories with Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1139–1148, 2010. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2010.179

[2] *Storytelling with Data Workshop*, co-located with VISWEEK 2010, 2010. [Online]. Available: http://thevcl.com/storytelling

[3] T. Gîrba, A. Kuhn, M. Seeberger, and S. Ducasse, "How developers drive software evolution," in *Proceedings of International Workshop on Principles of Software Evolution (IWPSE 2005)*. IEEE Computer Society Press, 2005, pp. 113–122. [Online]. Available: http://scg.unibe.ch/archive/papers/Girb05cOwnershipMap.pdf

[4] A. Kuhn, "Automatic labeling of software components and their evolution using log-likelihood ratio of word frequencies in source code," in *MSR '09: Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories*. IEEE, 2009, pp. 175–178. [Online]. Available: http://scg.unibe.ch/archive/papers/Kuhn09aLogLikelihoodRatio.pdf

[5] M. Ogawa and K. L. Ma, "Software evolution storylines," in *Proceedings of the 5th international symposium on Software visualization*, ser. SOFTVIS 2010. New York, NY, USA: ACM, 2010, pp. 35–42. [Online]. Available: http://dx.doi.org/10.1145/1879211.1879219

[6] A. Begel, Y. P. Khoo, and T. Zimmermann, "Codebook: discovering and exploiting relationships in software repositories," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ser. ICSE '10. New York, NY, USA: ACM, 2010, pp. 125–134. [Online]. Available: http://dx.doi.org/10.1145/1806799.1806821