Appeared in IEEE Software, Special Issue on Studying Professional Software Design, Vol.29, Issue 1, pp.34-38, January/February, 2012. [DOI: http://dx.doi.org/10.1109/MS.2011.125]

Toward Unweaving Streams of Thought for Reflection in Professional Software Design

Kumiyo Nakakoji(1) Yasuhiro Yamamoto(2) Nobuto Matsubara(1) Yoshinari Shirai(3)

(1) Key Technology Laboratory, Software Research Associates Inc., Japan
(2) Precision and Intelligence Laboratory, Tokyo Institute of Technology, Japan
(3) NTT Communication Science Laboratories, NTT Corporation, Japan

ABSTRACT

Software designers make design decisions covering a wide variety of aspects of the software to be designed through a nested intertwined process. Some of these dependencies among design decisions may not be obvious, though, especially for those who did not go through the design process. Extending or altering an existing design decision without fully understanding its dependencies may therefore result in a deterioration of the quality of the software design. Our approach to the challenge of avoiding such problems uses a recording of a design meeting and the Design Practice Streams (DPS) tools. DPS helps a designer to browse segments of the video data relevant to the designer's focused topic by specifying a region on the whiteboard or by choosing a few terms used in the transcript. This paper outlines the challenge, presents DPS, illustrates an experience of using DPS and applying it to the meeting record of a design study, and discusses this approach.

KEYWORDS:

software design support, design meeting record viewer, interaction design, DPS (Design Practice Streams)

Appeared in IEEE Software, Special Issue on Studying Professional Software Design, Vol.29, Issue 1, pp.34-38, January/February, 2012. [DOI: http://dx.doi.org/10.1109/MS.2011.125]

1. INTRODUCTION

For some, software is simply something executable by a computer system. For others, software is something people interact with. The design of software deals with the design of both the world of *making* (through planning and constructing the structure of the source code for the system to be designed) and that of *using* (through deliberating and composing how and via what steps a user would interact with the system to be designed as well as what user interface components).

Software designers make design decisions covering a wide variety of aspects of the software to be designed. Requirements specifications of functionality, software architectures, object models, and event diagrams serve *the world of making*. User stories and usage scenarios, interaction flows, user operation sequences, interface components layouts, and screen transitions serve *the world of using*.

By analyzing the transcript data of the three software design videos of the SPSD (Studying Professional Software Design) workshop, Baker and van der Hoek [1] have identified that each design process consists of nested and intertwined cycles, each of which represents a period of focus on a given aspect of the system being designed.

This is inevitably so because design by nature follows the process of coevolution of problem understanding and solution framing [7]. Designers need to externalize a partial decision in one aspect based on a partial understanding of other aspects [6]. Thus, design decisions made for one aspect of the system are guided, constrained, and governed by other decisions made in the context of other aspects.

Some of these dependencies among design decisions may not be obvious, especially for those who did not go through the design process. In practice, a software design project sometimes extends over a long period of time, and new members of a design team may need to work on an existing design. Extending or alternating an existing design decision without fully understanding its dependencies may result in a deterioration of the quality of the software design. Designers need to know what other decisions depend on a particular design decision, what arguments led to that particular decision, and what had and had not been considered when making that decision.

Our approach to address this information challenge is to use a recording of a design meeting. By looking at the conversations and whiteboard drawings at the time when a decision was made, a designer may understand what was being discussed when the decision was made, what had been worked on prior to making the decision, and what was worked on immediately after the decision was made.

The Design Practice Streams (DPS) tools help a designer browse the segments of a meeting video that are relevant to what the designer wants to investigate further. DPS uses temporal, spatial, and symbolic relations to retrieve segments of the video data relevant to a user's focused topic, which is expressed by specifying a region on the whiteboard or by choosing a few terms used in the transcript.

2. DPS: DESIGN PRACTICE STREAMS

DPS uses a simple time-stamp mechanism to relate the video data of a design meeting with the digital whiteboard drawing data (i.e., a set of time-stamped strokes drawn during the design meeting) and the textual transcribed data (i.e., a set of time-stamped utterances). DPS does not require any

manual tagging, annotations, or semantic interpretations; thus, it has no discernible impact on the flow of the meeting..

DPS identifies a set of strokes that are included within the region specified by a user on the whiteboard. Each stroke of the whiteboard drawing data is time-stamped, and the time stamps are used to identify one or more segments of the video data. Similarly, DPS identifies a set of utterances that include the few terms selected by a user from the transcript data. Each utterance is time-stamped, and then can be used to identify one or more segments of the video data.

DPS consists of three components: *MovieViewer*, *StrokeViewer*, and *TranscriptViewer* (Figure 1).

With *MovieViewer*, a user can play, pause, and fast-forward a meeting video, as well as change the playing speed, by using the control bar located at the bottom (see Figure1 (c), (f)). DPS may identify one or more segments of the movie based on the user's selection of a region in StrokeViewer or a few terms in TranscriptViewer. The identified segments, indicated with gray areas on the control bar, can then be continuously played.

StrokeViewer, which is based on the time-based sketchbook interface ART019 [8], consists of a canvas (representing a whiteboard) in the middle, a list of snapshots of the canvas on the left, and a stroke timeline pane on the right. The bottom table lists the information for each of the time-stamped strokes stored in the stroke database.

The timeline pane lists all the strokes ever written on the whiteboard according to their time stamps from the top (old) to the bottom (new). The horizontal position and the span of each stroke in the canvas are reflected in the relative position and length of the corresponding short line in the timeline pane. The strokes currently visible in the canvas are represented in black and those currently not visible in the canvas are represented in white in the timeline pane. If a user erases (i.e., deactivates) a stroke in the canvas, the corresponding short line in the timeline pane changes from black to white.

Snapshots consist of a set of visible (i.e., activated) strokes on the canvas (reproducing the whiteboard strokes). A snapshot is created for every set of consecutive erasing actions. Clicking on a snapshot activates the associated strokes and displays them on the canvas, allowing one to go back to one of the intermediate states of the whiteboard. The currently displayed strokes on the canvas can be replayed in the order of their time stamps in the "Review" mode.

A user may select strokes by specifying a rectangular region or drawing a closed figure on the canvas (Figure 1(a); Figure 2). All of the strokes, each of which partially belongs to the specified region, are then identified on the canvas. If the user specifies a rectangular shape while holding a shift key, StrokeViewer identifies not only the currently visible strokes but also the currently invisible ones belonging to the region (i.e., those that had been erased or not yet drawn at the time when the displayed snapshot was taken). In either case, the identified strokes change color to gray, and the corresponding short lines in the timeline pane are also visually emphasized with a gray horizontal background (Figure 1(b)).

DPS then selects segments of the movie with the corresponding time stamps in MovieViewer (Figure 1(c)), with which the user can browse all the movie segments in which the two designers were drawing the identified strokes.

TranscriptViewer comprises two vertical timeline panes on the left of the screen, the transcript pane in the middle, and the search pane on the right (Figure 1(d). The transcript data comprise a plain text file, in which each utterance starts with a time stamp (e.g., "00:11:16") and a speaker name ("M2").

Appeared in IEEE Software, Special Issue on Studying Professional Software Design, Vol.29, Issue 1, pp.34-38, January/February, 2012. [DOI: http://dx.doi.org/10.1109/MS.2011.125]

The leftmost timeline shows the timeline of the entire recorded session, starting from the top and ending at the bottom. The timeline bar next to it shows the zoomed-in time period, the corresponding position of which is displayed with a gray background area in the leftmost timeline. A user can *zoom in* or *zoom out* over the timeline by pressing, respectively, the "+" or "-" buttons located at the bottom. The transcript pane displays text transcripts within the zoomed-in time period. Each sentence is displayed in one line (with no wrapping) at the position of the corresponding time stamp in the zoomed-in timeline.

The user may type a phrase as a query in the window at the top of the search pane (Figure 1 (d)). The system then incrementally identifies matching terms to the queried phrase extracted from the entire transcript, and lists them below the query window. When choosing one or more matching terms (with AND or OR conjunctions), all the utterances that have the selected matching terms are displayed in bold font in the transcript pane.

In the same manner as with StrokeViewer, DPS then selects a few segments of the movie with the corresponding time stamps in MovieViewer (Figure 1(f)), and the user can browse all the movie segments in which the two designers mentioned the specified phrase.

DPS can incorporate any data stream as long as it has a mechanism to segment the data stream, each segment has a time stamp, and there is a way for a user to select and highlight one or more segments. For instance, a note viewer for personal time-stamped notes taken during the design meeting may be added to DPS to be associated with the other data streams.

Appeared in IEEE Software, Special Issue on Studying Professional Software Design, Vol.29, Issue 1, pp.34-38, January/February, 2012. [DOI: http://dx.doi.org/10.1109/MS.2011.125]

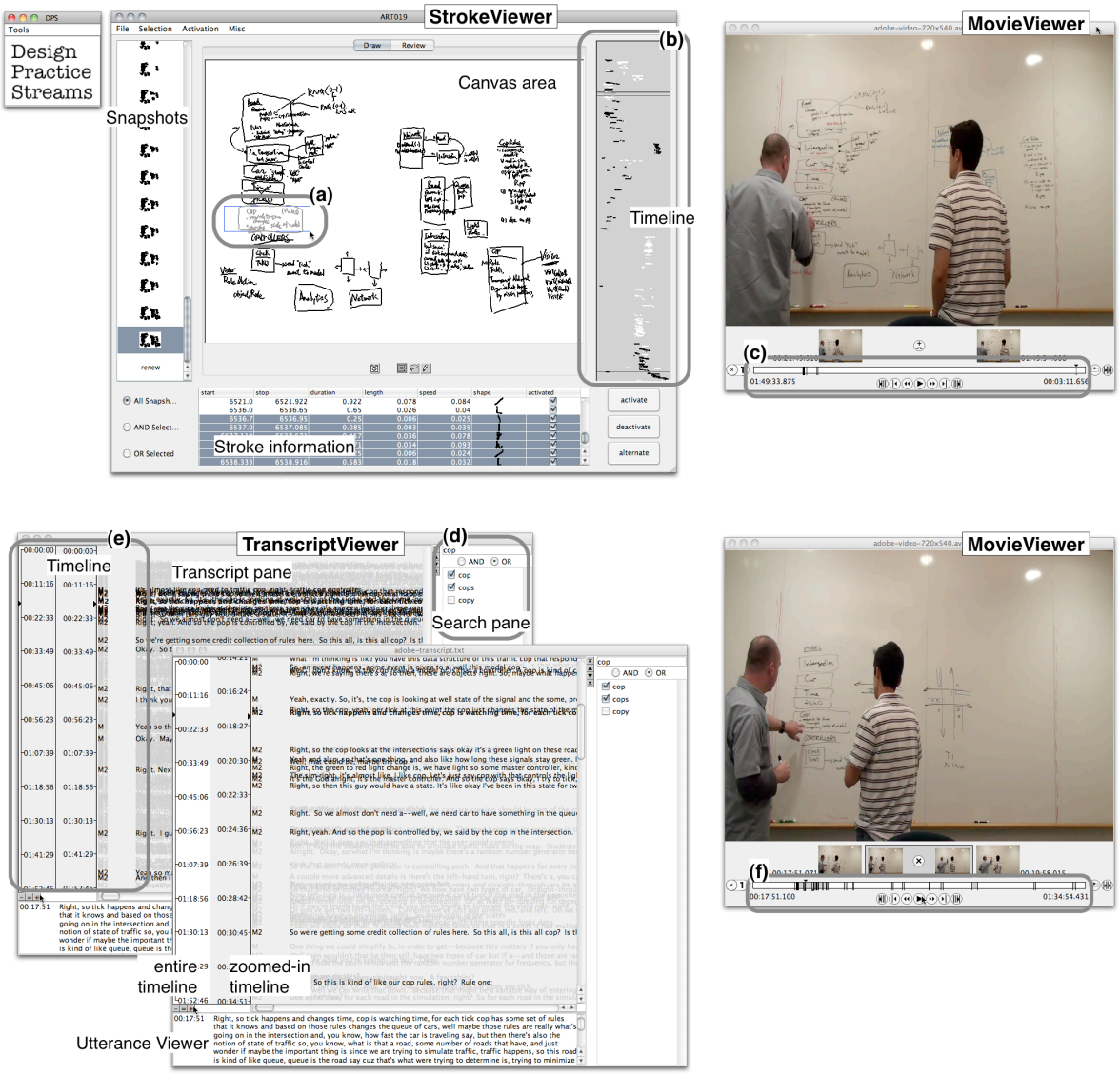


Figure 1: The Design Practice Streams (DPS) Tools

3. USING DPS

This section demonstrates our experience in using DPS to browse software design meeting records. We use the Adobe design team data from the SPSD workshop to illustrate our experience. Note that the original Adobe data did not have the whiteboard drawing data, so we have manually generated a set of stroke data for the videotaped whiteboard in sync with the video data.

- By browsing the recorded meeting in DPS, we observe that the designers of the Adobe team used the whiteboard as a medium to record their emerging design decisions. They segmented

the whiteboard roughly into several regions, and kept each region for externalizing design decisions regarding a particular aspect of the system to be designed. They used the left-hand side of the whiteboard to list object names, the rightmost vertical area to list rules and constraints, and the central area to write user scenarios. This assures us that what has been drawn as well as the region where the drawing was made would be a meaningful way to express the designer's focus.

- Circling the boxed "cop" object in StrokeViewer (Figure 1(a)) identifies the video segments in which the two designers wrote down the object name and its properties on the whiteboard. A few discrete segments along the timeline toward the beginning of the meeting are identified (Figure 1(b),(c)), which shows that the designers spent some time adding a few properties to the "cop" object.

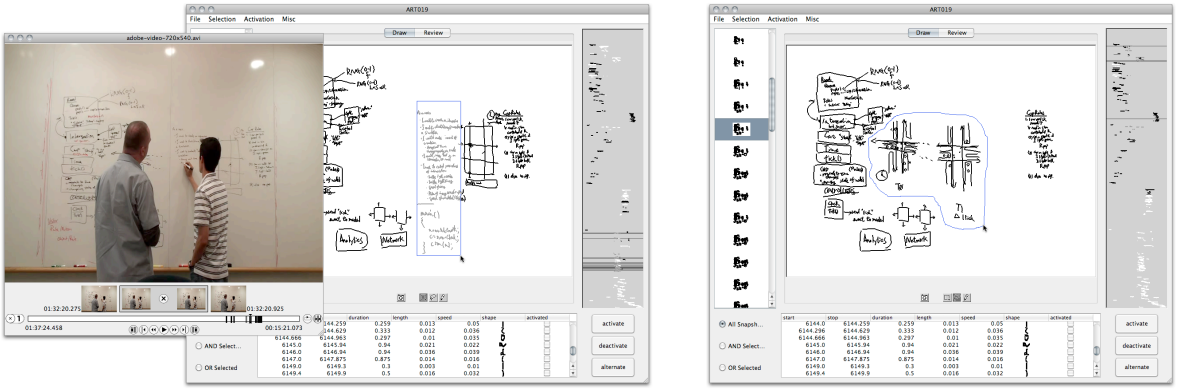
- Searching for "cop" or "cops" in TranscriptViewer identifies several segments in the video in which the two designers mentioned "cop(s)" (Figure 1(e),(f)). The timeline shows that most of these discussions were made toward the beginning of the design meeting, although a few remarks were also made toward the end of the meeting. Note that no changes were made to the "cop" drawing during the latter discussions (see above), which implies that the definition of the "cop" was untouched.

- By circling each of the boxed objects on the left side of the canvas of StrokeViewer and examining the corresponding video segments, we find that the "Road" object box has the most widely distributed segments along the timeline. This means that the "Road" object was most frequently revised and modified during the design meeting, and implies that the "Road" object may have had significant dependencies with other design decisions.

- Specifying a rectangular region around the entire user scenario sentences in StrokeViewer (Figure 2(a)) identifies a few video segments in which the two designers were writing down the entire user scenarios. This allows us to concentrate on the video segments in which the two designers talked only about the user scenario. We can also tell by looking at the timeline that the scenarios were composed after most of the boxed class objects were created.

- When we select the region where the two intersections were drawn at the center of the whiteboard in StrokeViewer, DPS identifies three primary sets of timeline segments (Figure 2(b)). Browsing the first two video segments in MovieViewer, we see that the two intersections were initially drawn as the two states of a single crossing road. One of the designers first drew the intersection on the left side and labeled it with "T0," and then drew the intersection on the right side and labeled it with "T1" to see "how pictures will change from one moment of time to another" (an excerpt from the transcript). The third video segment, which is about thirty minutes apart from the second video segment, reveals that the two designers started pointing to the two intersections and used them as if they were adjacent to each other. They drew dotted lines to connect the two intersections. Thus, "T0" and "T1" do not make any sense when dotted lines connect the two intersections, but they remained on the whiteboard.

Appeared in IEEE Software, Special Issue on Studying Professional Software Design, Vol.29, Issue 1, pp.34-38, January/February, 2012. [DOI: http://dx.doi.org/10.1109/MS.2011.125]



(a) (b)

Figure 2: Using DPS

4. DISCUSSIONS

We think that DPS would help software design practitioners in the following situations if the design meeting is recorded by videotape and a digital whiteboard:

- when a new member of a design team learns how the design has been evolved;
- when a member of a long-term design project (spanning several weeks, months, or years) looks back and validates the current state of the design between two design meetings;
- when a member of a maintenance project recollects the rationale of the design produced some time ago;
- when a member of a design project needs to explain the current state of the design to someone outside the design team, such as a project manager; or
- when a project manager checks a potentially problematic aspect of the current design to see how that part of the design had been deliberated.

Recorded meetings have long been studied, but “relatively little work has focused on the interfaces and interactions for reviewing recorded meeting content” [4]. Not many tools integrate video data with whiteboard drawing data.

Design Amanuensis [3] is one of the few efforts that integrate audio data with stroke data together with a keyword search over the audio data. DPS is an extension of Design Amanuensis, integrating video data with a region of the whiteboard. We think this extension is significant based on the observation that software designers often use a particular region of the whiteboard for a particular purpose. One may want to focus on not only strokes but also the region where the strokes are drawn.

Despite the fact that increasingly more recording devices have become available in practical settings, we think that the power of design meeting records continues to be underutilized. We have developed DPS to demonstrate that the simple time stamp-based mechanism, together with the visual interaction design, make the recorded meeting data handily available for a designer’s perusal.

Some researchers have started to explore meeting and whiteboard records specifically in the context of software development, such as studying how software developers use diagrams [2], and proposing a whiteboarding tool in software design [5].

In the same way that software developers have taken advantage of a wide variety of archival work within the working context (e.g., version control systems, communication archives, and issue-tracking systems), it is time now for us to pay more attention to archiving design meetings to help software designers gain an understanding of otherwise implicit dependencies and constraints among existing design decisions.

REFERENCES

1. A. Baker, A. van der Hoek, Ideas, Subjects, and Cycles as Lenses for Understanding the Software Design Process, Design Studies, Vol. 31, Issue 6, pp. 590-613, November 2010.

2. M. Cherubini, G. Venolia, R. DeLine, A.J. Ko, Let's Go to the Whiteboard: How and Why Software Developers Use Drawings, Proceedings of CHI 2007, ACM, New York, pp. 557-566, 2007.

3. M.D. Gross, B.R. Johnson, E.Y-L. Do, The Design Amanuensis: An Instrument for Multimodal Design Capture and Playback, Proceedings of CAAD Futures 2001, pp. 1-13, 2001.

4. H.R. Lipford, G.D. Abowd, Reviewing Meetings in TeamSpace, Human-Computer Interaction, Vol. 23, Issue 4, pp. 406-432, 2008.

5. N. Mangano, A. Baker, M. Dempsey, E. Navarro, A. van der Hoek, Software Design Sketching with Calico, Proceedings of ASE 2010, ACM, New York, pp. 23-32, 2010.

6. D.A. Schoen, The Reflective Practitioner: How Professionals Think in Action, Basic Books, New York, 1983.

7. Y. Yamamoto, K. Nakakoji, Interaction Design of Tools for Fostering Creativity in the Early Stages of Information Design, International Journal of Human-Computer Studies, Vol. 63, No. 4-5, pp. 513-535, October 2005.

8. Y. Yamamoto, K. Nakakoji, Y. Nishinaka, M. Asada, ART019: A Time-Based Sketchbook Interface, Technical Report, KID Laboratory, RCAST, University of Tokyo, March 2006.

Appeared in IEEE Software, Special Issue on Studying Professional Software Design, Vol.29, Issue 1, pp.34-38, January/February, 2012. [DOI: http://dx.doi.org/10.1109/MS.2011.125]

AUTHOR BIOGRAPHIES

Kumiyo Nakakoji is the director at Key Technology Laboratory, Software Research Associates, Inc., Japan. Her research interest is knowledge-interaction design for nurturing creative knowledge work, such as scholarly work or software development. Nakakoji has a Ph.D. in computer science from the University of Colorado at Boulder. Formerly, she served as a full professor at the University of Tokyo's Research Center for Advanced Science and Technology. She is a member of the ACM, IEEE Computer Society, Japanese Society for Artificial Intelligence, Japanese Cognitive Science Society, and Japanese Society for the Science of Design. Contact her at kumiyo@sra.co.jp.

Yasuhiro Yamamoto is an associate professor at Precision and Intelligence Laboratory, Tokyo Institute of Technology. His research interests include human-computer interaction, interaction design, and design process direction. Yamamoto received a Ph.D. in information science from Nara Institute of Science and Technology (NAIST), Japan. He worked as an associate professor at RCAST, University of Tokyo, where he co-directed the Knowledge Interaction Design laboratory. He is a member of the ACM and IEEE Computer Society. Contact him at yxy@acm.org.

Nobuto Matsubara is a researcher at Key Technology Laboratory, Software Research Associates, Inc., Japan. His research interests include learner-centered multimedia annotations, learning-support tools, and frameworks for embodied interaction. Matsubara has an M.S. in computer science from Osaka Electro-Communication University. He is a member of the Information Processing Society of Japan. Contact him at matubara@sra.co.jp.

Yoshinari Shirai currently works at Corporate Marketing Headquarters, NTT West Corporation, Japan, as an engineer. His research interests include history-enriched environments, ubiquitous computing, and interaction design. Shirai received a Ph.D. from the Graduate School of Engineering at the University of Tokyo. He is a member of the Information Processing Society of Japan, Virtual Reality Society of Japan, and Human Interface Society. Contact him at way@fw.ipsj.or.jp.