Presentations by Programmers for Programmers

Li-Te Cheng
IBM Research
li-te_cheng@us.ibm.com

Michael Desmond, Margaret-Anne Storey
Dept. of Computer Science, University of Victoria
michael.desmond@gmail.com, mstorey@uvic.ca

Abstract

A common form of live technical presentation is that given by programmers for a programming audience during conferences, demonstrations, code reviews, and tutorials. Such presentations require manual switching between general presentation software and the integrated development environment (IDE), as well as reconfiguration of the IDE's UI to be readable by an audience. In this paper, we present a novel system that allows programmers to easily combine traditional slideware with seamless transitions to user-specified regions of the IDE along with special effects for live demonstration.

1. Introduction

Live technical presentations typically involve a combination of "slideware" materials shown in a traditional general presentation tool such as Microsoft Powerpoint and an interactive demonstration. Slideware materials might include images of model schematics and screenshots from the live demonstration. One important form of technical presentation that we are interested in supporting are presentations given by programmers for a programming audience. Such presentations could occur during commercial and academic programming conferences, customer demonstrations, internal code reviews, and tutorials given in classrooms. Such live presentations typically involves slideware combined with manual switching over to the integrated development environment (IDE) where the programmer can walk through programming artifacts and finally run a demo. Because the IDE is normally meant for personal use at a desk, the presenter must reconfigure the user interface of the IDE to be suitable for presentation to a live audience (e.g. adjust the fonts, remove unnecessary elements of the user interface, etc). Alternatively, the presenter can take screenshots and place edited pictures into the presentation's slides (or prepare a movie) to show only the relevant pieces of information to the audience. Thus, the presenter must spend significant time and effort to prepare for a presentation. If screenshots or a movie was used, the presentation's demonstration will lose its "liveliness".

To address this challenge, we present "Tours", a novel system specifically designed for presentations given by programmers for programmers. The system works within the Eclipse IDE (www.eclipse.org), and allows presenters to easily construct presentations that combine slideware from general presentation tools with transitions that focus on specific user-specified regions of the IDE. These regions of the IDE are "live", allowing the programmer to interact with them during the presentation as if it was a normal programming session, and yet apply special effects normally associated with general presentation tools.

In the remainder of this paper, we discuss related work that helped motivate our system's features. We then describe the system that we developed, and close with a discussion on future directions, including feedback from programmers who experienced presentations given with our system.

2. Motivation and Related Work

Our system is motivated from our previous work with TagSEA, a system that helps programmers navigate through "waypoints" – locations in source code tagged with keyword metadata [6]. One feature was "routes" - the ability to create an order set of waypoints to navigate through different parts of the code, which is similar to JTourBus [4]. Our original intention with routes was to support personal navigation, but this was not used at all by users in our qualitative study [7], although NomadPIM integrated TagSEA to provide "code tours" [2].

The JTourBus work hinted at the potential of routes for education, guiding students around source code to help familiarize themselves with programming. The Gild system also considered how to simplify the Eclipse IDE for the educator as well as the students for first year programming courses [8]. However, we wanted to consider other types of presentation situations – such as code reviews and programming conference talks. In informal discussions about these situations with professional programmers and managers in the IBM organization, some indicated a desire that presented materials be connected with the actual programming artifacts, not frozen screen captures. Also, there was concern about the amount of work required to set up for such presentations, as well as the need for ways to stay focused on relevant artifacts being shown in the cluttered IDE user interface. This suggested that simply following a sequence of relevant locations in source code was not enough for a compelling presentation.

The missing element was the "spectator experience", as described by Reeves *et al.* [5]. They highlighted the importance of considering the "performance" aspects of public-facing systems, including general presentation tools. In the case of general presentation tools like Powerpoint, they indicated that it would be desirable for the presenter to "fluidly orchestrate the show for the spectator and reduce distractions". Once the presenter switches over to the IDE, a programmer-oriented presentation can be fluid, where the presenter may navigate through programming artifacts and manipulate them. At any point, the presenter may decide to make impromptu decisions, or may be prompted by the audience (e.g. in response to questions during a code review or tutorial), to deviate slightly from plan, and maybe even write new source code on the spot. However, without careful setup, there may be many distractions along the way (e.g. changing the fonts, adjusting layouts, removing unnecessary elements).

	Synchronous	**Asynchronous**
Live Content	JTourBus [4] Livecoding (e.g. [9]) General screen-sharing (e.g. VNC) Gild [8]	Code review tools (e.g. [10]) Help Systems (e.g. Eclipse IDE Cheat Sheets) Scripts/Macro recordings (e.g.Netbeans IDE macro recorder) JTourBus [4] NomadPIM [2] TagSEA Routes [6] Gild [8] Guides [1]
Frozen Content	General presentation tools (e.g. Powerpoint)	General presentation tools (e.g. Powerpoint) Screencam movies (e.g. Camtasia)

Table 1: Summary of Related Work

While we are interested in supporting technical presentations for programmers, Reeves *et al.* consider a broader range of domains, notably the arts. There is also an artistic analogue for presentations for programmers – "livecoding" (e.g. [9]). Livecoding requires significant skill, which is part of its appeal: to create expressive performances from presenters who are programming on a constantly running application. We are focused on a presentation tool accessible to most programmers that may not require a constantly running application.

We summarize related work in Table 1, which is classified across two dimensions: synchronous/asynchronous versus live content/frozen content. A synchronous system is mainly intended to be given during a presentation with the audience co-located at the same time, whereas an asynchronous system is intended to be reviewed offline. A presentation with live content remains consistent regardless of any change of its demonstrated programming artifacts, and the artifacts can be altered during the presentation, whereas a presentation with frozen content does not (e.g. a screenshot of the source code). We are targeting our work towards a synchronous / live content experience, with some asynchronous support.

3. Tours: Enhanced Routes for TagSEA

Tours is an Eclipse IDE plugin that allows programmers to create and present technical talks about their work to a programming audience. Creating and presenting with the Tours system is similar to using a general presentation tool like Microsoft Powerpoint, including the notion of adding special effects. But in addition to slides, a tour (a presentation created by Tours) can include pieces of source code, which follows from the Routes concept and similar notions described in the previous section, as well as user interface elements in the IDE that can be manipulated by the presenter during the talk. In this section, we describe the Tours interface.

3.1. Designing a Tour

A tour is created like any other file in the Eclipse IDE – via the "File/New" menu. A ".tour" file is generated, and an editor is presented to the user. Figure 1 shows an example tour file called "action.tour". The editor consists of several collapsible sections: "Metadata Information", "Contents", "Tour Palette", and

"Notes". Any changes made to the tour file can be saved by the "File/Save" menu.

The "Metadata Information" section contains general information about the tour – the name, description, and author – which is presented as a form. By clicking on the triangle beside "Metadata Information", the form can be collapsed.

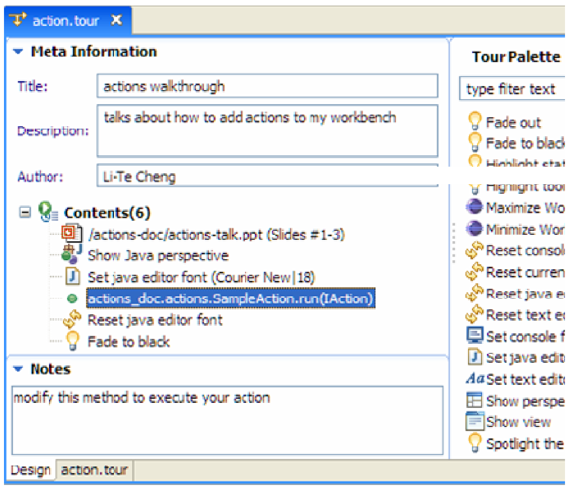
The "Contents" section contains a sequential outline of what will occur during the presentation, which is represented as a single level tree. Right-clicking on "Contents" will allow the user to configure the overall presentation setup, such as set up a time limit for the talk.

The elements under "Contents" can be references to Powerpoint slides (e.g. the first element in Figure 1 is a reference to the first 3 slides of a Powerpoint presentation), references to programming artifacts (e.g. the fourth element in Figure 1 is a reference to the "run" method of the Java class, SampleAction), references to web pages, and references to IDE user interface elements (e.g. "Show Java Perspective"), and special effects (e.g. "Set Java Editor Font…", "Fade to Black", etc). The user can add any of these elements and rearrange them by drag and drop. The various references (slides, programming artifacts, web pages) are dragged from other parts of the Eclipse IDE user interface (e.g. the various file browser-like interfaces such as "Package Explorer" and "Resource Navigator"; programming-specific interfaces such as the "Java Outline View", "Class Hierarchy View", etc). We also support dragging in "waypoints" from the TagSEA tool, which is similar to dragging in programming artifacts. Special effects are dragged from the "Tour Palette" on the right. Double-clicking on elements brings up a dialog to configure how they will be presented (e.g. selecting specific slides, selecting fonts, etc).

Specific blocks of code can also be added to the Contents of a tour. The user can select a block of code in any editor window and select a right-click menu option. This works for any text-based editor supported in Eclipse.

The "Tour Palette" contains the set of special effects that can be applied on the entire desktop (e.g. "Fade to Black") and the Eclipse IDE (e.g. "Spotlight the Workbench"). A textbox on the top allows the user to type the first few characters of any special effect name to filter down the long list of special effects. The palette can be collapsed like a sliding drawer by clicking on the dividing vertical sash.

Each element in the Content section can have presentation notes associated with it. Typing in the text box in the "Notes" will associate what is typed with the currently selected element.


action.tour

Meta Information

Title: actions walkthrough

Description: talks about how to add actions to my workbench

Author: Li-Te Cheng

Contents(6)
- /actions-doc/actions-talk.ppt (Slides #1-3)
- Show Java perspective
- Set java editor font (Courier New|18)
- actions_doc.actions.SampleAction.run(IAction)
- Reset java editor font
- Fade to black

Notes

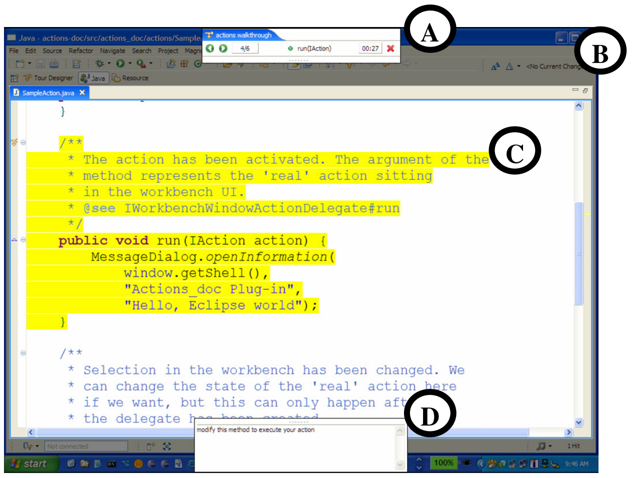
modify this method to execute your action

Figure 1: The Tour Editor

3.2. Presenting a Tour

Right-clicking on the .tour file in the Eclipse file browser and selecting "Run tour", or clicking on the "Run Tour" button in the main Eclipse toolbar will begin the presentation of the tour. An example of a running Tour is shown in Figure 2.

On the very top of the desktop (Figure 2A) is the "Runner" widget that remains on top of the desktop regardless of whatever is being currently shown in the tour. This widget has controls to advance to the next element or reverse to the previous element in the presentation, as well as indicators showing what element is being shown (e.g. name of the Powerpoint file), how far the presenter is in the presentation (how many elements have been shown versus total number of ele-



```
    }

    /**
     * The action has been activated. The argument of the
     * method represents the 'real' action sitting
     * in the workbench UI.
     * @see IWorkbenchWindowActionDelegate#run
     */
    public void run(IAction action) {
        MessageDialog.openInformation(
            window.getShell(),
            "Actions_doc Plug-in",
            "Hello, Eclipse world");
    }

    /**
     * Selection in the workbench has been changed. We
     * can change the state of the 'real' action here
     * if we want, but this can only happen aft
     * the delegate h
```

modify this method to execute your action

Figure 2: Running a Tour inside the IDE

ments, as well as a stopwatch indicator), and button to abort the presentation. The widget uses a tabbed interface, because Tours supports running multiple presentations simultaneously (each running tour will create a new tab). The user can collapse this widget by clicking on the "drawer" sash on the bottom.

On the very bottom of the desktop (Figure 2D) is the Notes widget that shows the presentation notes associated with the currently shown element. The user can type in this widget and change the notes during the presentation, and these changes are saved with the tour file. Right clicking in this widget brings up options to increase or decrease the font of the presentation notes. Like the Runner widget, the user can collapse this widget by clicking on a "drawer" sash (on the top of the widget).

As the user steps through the tour, slides, files, and effects will be shown. Figure 2 shows a block of code in a Java program with a "letterbox" animation effect (darkening regions of the desktop with an alpha blending as seen in Figure 2B). The tour was set up to maximize the source code editor, increase the font size of the editor and highlight the desired block of code in yellow (Figure 2C). As a result, the tour provides a clear view of what the presenter wants to show to the audience, without the extraneous bits of the rest of the Eclipse IDE. Other special effects are available for more "focused" views of the programming environment, such as "focus", which blacks out everything except the a predefined targeted section of the IDE, and "spotlight", which lets the user click, during the live presentation, on arbitrary windows and toolbars in the Eclipse user interface and automatically highlight them and fade everything else out on the screen.

4. Future Directions

The Tours system offers an opportunity for programmers to give live technical presentations that weave together static slides with dynamic content. The user can interact directly with software artifacts, as well as fade out the uninteresting pieces.

We plan to deploy the tool for user evaluation and refine the design. But already we have created and shown a few tours to colleagues, and our informal feedback was quite positive. This included participants from our qualitative study of TagSEA, who were interested in using the Tours system to help in code reviews, technical talks, as well as write documentation. One participant remarked that Tours was like Routes, but with a tour guide equipped with a flashlight and a dimmer switch.

We are also planning to explore how Tours can operate in remote presentations. One idea is to also support audience participation – viewers can annotate the Notes widget, as well as bookmark interesting pieces shown during the presentation. We are also exploring integration with Dogear, a social bookmarking system [3].

5. References

[1] Dagenais, B. and Ossher, H. Guidance through Active Concerns. In *Proc. Eclipse Technology Exchange Workshop, OOPSLA 06*, ACM Press (2006).

[2] Grammel, L. NomadPIM. http://nomadpim.sourceforge.net

[3] Millen, D.R., Feinberg, J., and Kerr, B. Social bookmarking in the enterprise. In *Proc. CHI 06,* ACM Press (2006), 111-120.

[4] Oezbek, C. and Prechelt, L. JTourBus - Simplifying Program Understanding by Documentation that Provides Tours Through the Source Code, Intitut fuer Informatik, Freie Universitaet Berlin, http://projects.mi.fu-berlin.de/w/bin/view/SE/JTourBus

[5] Reeves, S., Benford, S., O'Malley, C., and Fraser, M. Designing the spectator experience. In *Proc. CHI 05*, ACM Press (2005), 741-750.

[6] Storey, M.A., Cheng, L., Rigby, P., and Bull, I. Shared Waypoints and Social Tagging to Support Collaboration in Softare Development. In *Proc. CSCW 06*, ACM Press (2006).

[7] Storey, M-A., et. al. Turning Tags into Waypoints for Code Navigation. Submitted to *CHI 07.*

[8] Storey, M-A., et. al. Improving the Usability of Eclipse for Novice Programmers. In *Proc. 2003 OOPSLA Workshop on Eclipse Technology Exchange Program,* ACM Press (2003).

[9] Wang, G. and Cook, P. On-the-fly Programming: Using Code as an Expressive Musical Instrument. In *Proc. New Interfaces for Musical Expression 04,* Hamamatsu, Japan, 2004.

[10] Yamashita, T. and Kou, H. Jupiter. University of Hawaii, http://csdl.ics.hawaii.edu/Tools/Jupiter/