MCT: A Tool for Commenting Programs by Multimedia Comments

Yiyang Hao*†, Ge Li#*†, Lili Mou*†, Lu Zhang*†, and Zhi Jin*†‡
*Software Institute, School of Electronic Engineering and Computer Science, Peking University
†Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education
Beijing 100871, P. R. China
‡Academy of Mathematics and System Science, China Academy of Science, Beijing 100190, P. R. China
felixhao@pku.edu.cn, {lige, moull12, zhanglu, zhijin}@sei.pku.edu.cn

***Abstract*—Program comments have always been the key to understanding code. However, typical text comments can easily become verbose or evasive. Thus sometimes code reviewers find an audio or video code narration quite helpful. In this paper, we present our tool, called MCT (Multimedia Commenting Tool), which is an integrated development environment-based tool that enables programmers to easily explain their code by voice, video and mouse movement in the form of comments. With this tool, programmers can replay the audio or video when they feel like.**

A demonstration video can be accessed at:
http://www.youtube.com/watch?v=tHEHqZme4VE

I. Introduction

During the development of a software system, leaving messages to explain the code is as important as coding itself. Messages play a vital role in software reuse and maintenance. It is very common that some engineers have to maintain code or projects written by other people [1]. Mostly, this situation is caused by developers' departure or their job has been changed.

The most used way to leave messages is writing comments and documents. Typical comments are based on texts. They are often inserted before class declaration and method declaration, and sometimes at the end of a line of code [2]. Comments contain much information about the functionality of a certain code block, and the programmer's ideas implied in the program. However, writing good comments is by no means easy. Inexperienced programmers can easily write a verbose comment by talking about everything [3][4] or an evasive comment by providing insufficient or irrelevant information. Sometimes, some programmers just feel lazy and leave blanks. Numerous guidelines have been proposed to help programmers write good comments [5][6].

As for documents, such as Javadoc[1], they provide sound and enriched information about software projects. For example, Javadoc supports simple formats and embedded URL resources. However, a disadvantage of documents is that they are too formal and often take much time to construct. On the other hand, when reading the documents, people are sometimes overwhelmed by the size, so they still do not have a brief but clear idea of the whole project. More importantly, targeted readers of documents are the users of this project, not programmers [7].

Different from traditional text-based comments and documents, an audio or video clip will help a lot in code understanding. It is natural that people learn faster with multimedia information [8]. In addition, written language differs from oral language in many ways so that the audience of an audio or a video clip can perceive much more information beyond words [9]. For example, the tone with which the speaker says may indicate the really important code piece from unimportant ones. The speaker can even use gestures or teaching aid facilities during the talk. Some people are visual thinkers, and they prefer looking at a workflow, a database diagram or a UML graph. So it is more convenient if the original author has commented his/her code in the form of an audio or video clip in addition to text comments.

To help programmers who want audio or video comments in their source code, we created Multimedia Commenting Tool (MCT), which is a plug-in on integrated development environment (IDE). This is a new tool designed for software developers to record or replay multimedia comments. MCT supports the following two forms of multimedia comments:

- **Multimedia code narration.** Using MCT, programmers can record a multimedia narration, where they explain their code by audio or video. Programmers can use gestures or equipment, such as a small blackboard to draw a flow chart, to help them during the narration. Therefore, code narration is a quick way for programmers to understand the code.
- **Embedded multimedia resources.** MCT also allows to embed existing multimedia resources (e.g., images, audio clips and video clips) into source code. Voice can also be translated into text by MCT. Code reviewers can watch or listen to embedded multimedia resources inside the IDE.

As far as we are aware, MCT is the first tool that introduces the concept of multimedia comments and helps manipulate them.

II. Overview

MCT is an IDE plug-in to create, modify and play multimedia comments. The main interface of MCT is shown in Fig. 1. Multimedia comments can be created in two forms:

#Corresponding author.

[1]http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html

978-1-4673-3076-3/13 © 2013 IEEE

by recording a code narration or by embedding existing multimedia resources.

To comment by multimedia code narration, programmers can provide either an audio or a video explanation to their code. MCT can record videos from cameras and audio from microphones. At the same time, MCT also keeps track of mouse movements so as to retain correspondence between code and multimedia contents. Multimedia code narrations can be played sequentially, or can be triggered by the related code lines or variables.

To comment code by embedded multimedia resources, programmers can insert multimedia files, i.e., images, audio clips and video clips. A descriptor, indicating an embedded multimedia resource, is then inserted to the corresponding line in the source file. Clicking the descriptor, code reviewers can listen to the audio, or watch images and videos in a pop-up window.

Both forms of multimedia comments can be created during the development and updated when code changes. MCT will automatically keep consistency between multimedia contents and lines of the code.

By using MCT, code reviewers may easily understand the whole source file and any detailed code snippets.

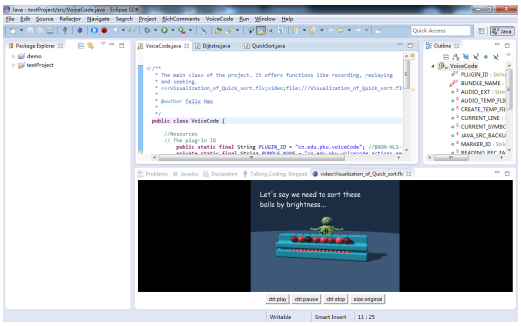

Fig. 1. Overview of Multimedia Commenting Tool (MCT).

III. Demo Walk-Through

In this section we introduce Jane, a fictitious character, who is a software developer using MCT. She is moving to a new software project due to the departure of a previous engineer. Following Jane, we can see how MCT helps programmers.

A. Creating Multimedia Comments

Before the project rotation, Jane has to leave some messages to explain her work so that other developers can easily understand her code and continue the project. She is creating multimedia comments with MCT.

1) Commenting with Multimedia Code Narrations: Jane wants to give a video explanation to her code. She clicks on the "start recording" button and starts narrating while she is coding. When finished, she clicks on the "start recording" button again. The video as well as her mouse movement is saved in a resource folder alongside the source file (Fig. 2).

After development, Jane feels the need to comment a detailed algorithm implemented in her code. She clicks on

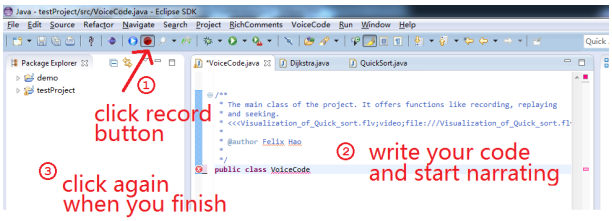

Fig. 2. Recording narration while coding.

the "start recording" button and starts narrating the algorithm. While recording, she uses her mouse to select the code snippets to indicate where she is narrating (Fig. 3). By doing so, the code reviewers can easily follow her narration. She also utilizes a blackboard beside her to draw a structure graph of the project with gestures. On finishing recording, she clicks on the "start recording" button again and the multimedia comment is saved.

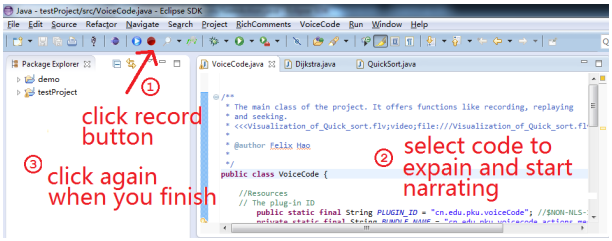

Fig. 3. Recording a multimedia narration of the code.

2) Adding Embedded Multimedia Resources: Jane feels it is not enough with only narrations; she wants to present the data structure of her program. She clicks on the "insert" button, inputs the title of the graph as "data structure" and chooses the type as "image" (Fig. 4). She can either choose a local file or use a URL to download an existing image on the web. This feature is used generally when a visual or acoustic description or example is needed, e.g., describing a complex algorithm.

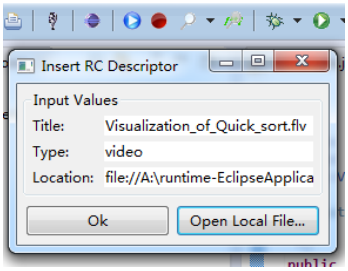

Fig. 4. Adding and embedded multimedia resource.

3) Writing Text Comments by Voice: Another feature of MCT is voice commenting, which has been thoroughly discussed by Andrew Begel [10]. MCT recognizes Jane's voice into plain texts and adds it as a comment in her code. Jane needs to click on the "voice comment" button and speak to her microphone. The voice commenting window will pop up and display what MCT has heard. When she is done, Jane

clicks on the button again and what she says becomes text comments.

4) Comments vs. Documents: Generally, comments help programmers understand source code and are strongly related to the structure of source files [2]. It is not recommended to leave messages about the whole subsystem or parts of it which consist of several source files in comments. MCT does not support Jane to leave cross-file messages, because frequent code updates make such messages obsolete easily.

B. Understanding code with Multimedia Comments

Jane is now in her new software project. She has to begin working as soon as possible after understanding the requirement as well as the detailed source code.

1) Watching the Whole Multimedia Code Narration: Jane opens the source file. She simply clicks on the "play" button and the whole code narration is played. A window pops up in the IDE and she begins watching the authors narrating their code.

When the video is playing, the original author's mouse movement reappears in the editor. The editor automatically scrolls down or up to make the current selection visible. Jane finds it quite easy to catch up to the narration following the mouse movement.

When coming to a particular function, Jane hopes to have a detailed look at the work flow so that she can understand it better. The narrator takes out a small blackboard and starts drawing and explaining the work flow diagram (Fig. 5).

Fig. 5. Watching the original author's narrating of a flow chart.

2) Retrieving Multimedia Code Narration Segments: When Jane starts to manipulate the code, she needs a narration over a certain code segment. With MCT, multimedia code narration segments can be retrieved either by code line or by method/variable identifier.

Jane clicks on the "toggle markers" button and all commented lines are highlighted. She moves her mouse to that line, and clicks on the "seek to current line" button (Fig. 6). The author immediately appears and begins narrating that part.

If Jane needs to know all about a method or a variable, she can select the identifier of that method or variable, and clicks on the "seek for current symbol" button. All the lines containing the identifier are highlighted. All narration segments about the identifier are then played. Now Jane knows all about the variable/method, including the declaration and all its usages.

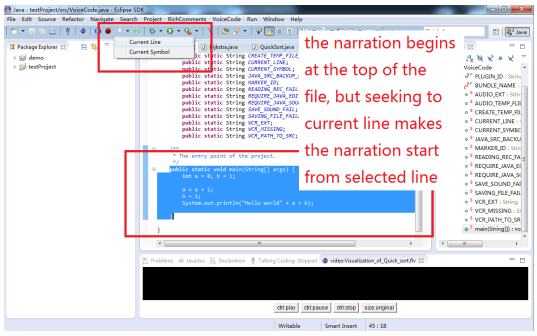

Fig. 6. Retrieving narrations segments by code line.

3) Playing Embedded Multimedia Resources: Another type of multimedia comments is embedded multimedia resources. A descriptor is embedded in the source file indicating a multimedia resource. These descriptors are in the form of text comment themselves. When Jane needs to play any multimedia resource, she can click on the descriptor. For an audio clip, it will be played directly; for other resources, a new window pops up where Jane can watch images, videos or something else.

C. Updating Code and Multimedia Comments

Jane soon understands the new project, and now she finds a bug in the source code. She starts worrying "what happens if I modify the code?" Fortunately, MCT automatically retains consistency between code and multimedia comments. Details of this feature are described in Implementation section.

Code changes include code insertion, deletion and modification, a combination of deletion and insertion. If a line of code is deleted or modified, it becomes no longer available and the multimedia comment on this line should be skipped. The positions in recorded mouse movements should be adjusted accordingly or ignored when code changes.

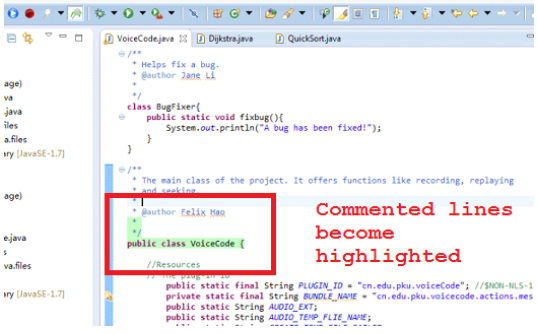

Fig. 7. Toggle multimedia comments markers.

By giving Jane the opportunity to watch and make multimedia comments, MCT makes it possible for an engineer to leave a multimedia narration on his code for Jane to watch. Reading the whole code may take at least 10 minutes for Jane and Jane may get stuck on some unimportant details in this code. But a three-minutes-long video tour on this class is enough

to understand how this class works. In addition to that, Jane can take advantage of MCT for her own sake by adding a video narration to her code, so she can just leave critical text comments and let the video do the rest.

When a code narration is already contained in a source file, Jane can replace it by recording another one. It is hard to edit existing code narrations with an IDE plugin efficiently for obvious reasons. However, she can use any external video or audio editors when applying small modifications to code narration. We may implement features to separate the existing video or audio into several clips by recognizing spaces between sentences and allow users to modify it just like modifying text comment in the future.

IV. Implementation

MCT is realized as an Eclipse plugin (tested on INDIGO and JUNO). A multimedia comment basically contains an audio or a video clip, a file recording mouse movements, a copy of the source file when the comment is created and optionally extra files due to code changes.

When a user clicks on the "start recording" button, we first check whether there are any multimedia comments already. If so, the newly recorded comment will be appended to the tail of the old ones. If not, a new comment will be created. We record mouse movements and audio with a microphone and video with a camera. The user can choose to use audio or video feature or not to control source code size. The tool is still a prototype and only supports the .wav audio format and the .avi video format. Any code modifications during the recording are not encouraged and may cause uncertain consequences.

When a user clicks on the "replay" button, we load the recorded mouse movement into memory and compare between the currently opened file and the original one when the comments are recorded. The differences tell the MCT which line in one file corresponds to which line in another. Therefore, MCT would not replay a comment written for deleted code. Then the recorded mouse movements are performed and audio or video clips are played. If a video clip is available, a view window will show up to display the video.

Consistency between multimedia comments and code is maintained in the following way: A file difference comparison will be done each time a multimedia comment is loaded into memory. MCT detects whether a line is removed or moved based on the differences, e.g., deleted lines and added lines. If a line is moved during code maintenance due to code insertion or deletion, mouse movements will be adjusted to fit the corresponding lines. If a line is deleted, corresponding mouse movements will be ignored but the audio or video clip related to that line will be remained because the multimedia clip may still be useful. Newly added audio or video clips are appended at the end of the existed clips.

MCT uses CloudGarden TalkingJava SDK[2] for speech recognition as an implementation of Java Speech API[3]. MCT will automatically detect an installed English speech recognition engine. If there is none available, the user will be warned and may not use the speech comment input feature.

MCT uses VLC player[4] as our video decoder, so users may need to install a free VLC player to experience video features.

The MCT plug-in can be downloaded at
http://www.sei.pku.edu.cn/∼lige/MCT/plugins.7z

V. Discussion

In this paper, we introduce the idea of multimedia comments and present an IDE plug-in named MCT, Multimedia Commenting Tool. Using MCT, programmers can record an audio or video code narration to explain their code. They can also insert embedded multimedia resources into source files. The correspondence between code and multimedia comments will be retained by MCT automatically.

In the future, we will conduct a survey on software developers in indutry to see how MCT works in real development. We will deploy our system to university students and teaching assistants to see how MCT can help in education.

VI. Acknowledgment

This research is sponsored by the National Basic Research Program of China (973) No. 2011CB302704, the National Natural Science Foundation of China No. 61232015, the National 863 Program of China No. 2012AA011202, and the Science Fund for Creative Research Groups of China No. 61121063.

References

[1] T. Pearse and P. Oman, "Maintainability measurements on industrial source code maintenance activities," in *Software Maintenance, 1995. Proceedings., International Conference on*, 1995, pp. 295–303.

[2] D. Haouari, H. Sahraoui, and P. Langlais, "How good is your comment? a study of comments in java programs," in *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, 2011, pp. 137–146.

[3] A. Carbone, J. Hurst, I. Mitchell, and D. Gunstone, "Principles for designing programming exercises to minimise poor learning behaviours in students," in *Proceedings of the Australasian conference on Computing education*, ser. ACSE '00, pp. 26–33.

[4] S. P. R. R. Douglas Riecken, Jurgen Koenemann-Belliveau, "What do expert programmers communicate by means of descriptive commenting?" in *Empirical Studies of Programmers: Fourth Workshop*, 1991, p. 177C195.

[5] J. Vogel, "Six ways to write more comprehensible code," *http://www.ibm.com/developerworks/linux/library/l-clear-code/*.

[6] Javarevisited, "10 best practices to follow while writing code comments," *http://javarevisited.blogspot.com/2011/08/code-comments-java-best-practices.html*.

[7] D. Kramer, "Api documentation from source code comments: a case study of javadoc," in *Proceedings of the 17th annual international conference on Computer documentation*, ser. SIGDOC '99, pp. 147–153.

[8] L. J. Najjar, "Multimedia information and learning," *Educational Multimedia and Hypermedia*, pp. 129–150, 1996.

[9] "Differential effects of home literacy experiences on the development of oral and written language," *Reading Research Quarterly*, vol. 33, no. 1, 1998.

[10] A. Begel, "Program commenting by voice," *http://www.cs.berkeley.edu/ abegel/cs294-1/voicecomments.pdf*, 2002.

[2] TalkingJava SDK with Java Speech API implementation, http://www.cloudgarden.com/JSAPI/

[3] Java Speech API - http://en.wikipedia.org/wiki/Java_Speech_API.

[4] VLC media player - http://www.videolan.org/vlc/index.html.