GitSonifier: Using Sound to Portray Developer Conflict History

Kevin J. North, Shane Bolan, Anita Sarma, Myra B. Cohen
Dept. of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115, USA
{knorth,sbolan,asarma,myra}@cse.unl.edu

ABSTRACT

There are many tools that help software engineers analyze data about their software, projects, and teams. These tools primarily use visualizations to portray data in a concise and understandable way. However, software engineering tasks are often multi-dimensional and temporal, making some visualizations difficult to understand. An alternative for representing data, which can easily incorporate higher dimensionality and temporal information, is the use of sound. In this paper we propose the use of sonification to help portray collaborative development history. Our approach, GitSonifier, combines sound primitives to represent developers, days, and conflicts over the history of a program's development. In a formative user study on an open source project's data, we find that users can easily extract meaningful information from sound clips and differentiate users, passage of time, and development conflicts, suggesting that sonification has the potential to provide benefit in this context.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Programming teams*

General Terms

Design, Human Factors

Keywords

version control history, sonification, conflicts

1. INTRODUCTION

Software engineers are becoming increasingly reliant on tools that aggregate and present information about the code or tasks they work on. For instance, tools exist to help identify and resolve version control conflicts before they become central to a project's critical path [9], the most comprehensive of which contextualize the information that is gathered. Other tools provide information to aid debugging or code comprehension [6] or to view code change history [9]. A majority of these tools present information through visualizations; however, visual data can be limiting if tasks are of high dimensionality or include temporal information. Take, for instance, the history of a distributed development task. Multiple users may join or leave the project, make commits, or sit idle for days at a time, especially in an open source context. When conflicts occur, they can last for days before resolution is complete. And more than a single conflict can happen at any one time. If we consider just developers, days and conflicts, this is a 3-dimensional space. If we also add the number and size of conflicts, the ability to easily view this information quickly deteriorates.

Using sound to portray data (or *sonification*) has been successfully applied in many domains including software engineering. One of the first tools, SonicFinder, utilized sound as auxiliary feedback to users. Sonification has also been used for program comprehension [5, 11], performance tuning [3], and debugging [10, 12]. McIntosh et al. create a sonification of project histories that renders an artistic orchestral composition [7]. Hussein et al. show that sonification is a good alternative to visualization for program comprehension. While these projects provide an argument for sonification, there is no work that satisfies our goal – to utilize sound to understand and improve collaborative software development through the use of historical conflict data.

Several awareness tools exist that inform developers working in parallel about emerging conflicts, via unobtrusive alerts or visualizations that present the state of the project [9]. Here we explore how sonifying this data may provide additional context. Suppose a project manager wants to "hear" her team's recent activities each morning to be aware of who made new commits and who might have version control conflicts with another. She can listen while performing other tasks, and if she notices anything interesting, e.g., a conflict, she can switch her attention for a few seconds. This provides the ability to multitask, absorbing background information. In contrast, a visualization portraying history (e.g., [8]) requires undivided attention. In addition, developers might listen to a sonification of the project to obtain a quick understanding of the intensity and pattern (*rhythm*) of commits and conflicts before joining the team.

In this paper, we propose the use of sonification and perform a formative study to evaluate the feasibility of hearing conflict management data. Our study shows that users can comprehend the information presented with high accuracy, regardless of their prior experience with music, suggesting a potential alternative for information discovery and contex-

ESEC/FSE'15, August 30 – September 4, 2015, Bergamo, Italy
© 2015 ACM. 978-1-4503-3675-8/15/08...$15.00
http://dx.doi.org/10.1145/2786805.2803199

tualization in coordinated project development. The contributions of this work are (1) the use of sonification for understanding Git conflict history, implemented as GitSonifier and (2) a user study showing that the data within GitSonifier sound clips can be easily understood, with no correlation between accuracy and prior musical experience.

2. BACKGROUND

There are several techniques for designing sonifications. Audification plays (as sound) data that is already in the form of a physical wave. Mild transformations can be applied to the data. For example, seismologists audify earthquake records, but since the waves have a frequency lower than humans can detect, the records are sped up, making them easier to hear [4]. We do not use audification in this work, but instead use other sonification techniques.

Auditory icons are sounds representing pieces of data. They are selected so that their meanings are intuitive. For example, the SonicFinder program added sound effects to a file explorer, such as the sound of an object scrapping on the ground while a user dragged a file. The sound effect is intuitive because dragging an object in the real world produces a scrapping sound [2, 4].

Parameter mapping sonification (PMSon) maps different dimensions of a data set to a characteristic sound wave, which is then manipulated over time to represent the data. Consider using a thermometer in a pot of boiling water. The thermometer's reading can be shown by a visual graph of temperature over time. Analogously, a PMSon could map the temperature to a sound wave's pitch, increasing the pitch as the temperature gets higher [4]. SonicFinder used PMSon to indicate the size of a file as it was dragged [2].

Like auditory icons, *earcons* are sounds associated with data. Earcons use sounds that have an abstract relationship with the data. As a result, interpreting earcons requires training, but earcons can be used when data has no natural auditory metaphor available. Earcons are usually musical *motifs*, or short musical phrases. Related earcons can share musical characteristics so their relationship is easier to recall [4]. Motifs have several characteristics. Pitch refers to which notes are played. Rhythm is the timing of the notes. Timbre is the sound of the instrument playing the motif. A *measure* is a group of notes representing all or part of a motif, and is the basic unit of time in music.

In this work we use earcons to represent our data, since we do not have a natural sound for software commits or conflicts. This is consistent with other recent sonification work in software engineering [5–7, 10–12]. We also use PMSon to signal the intensity of conflicts.

3. GITSONIFIER

Figure 1 shows how GitSonifier works[1]. GitSonifier parses Git data (#2), flattens the data into a timeline (#3), adds a layer indicating the number of conflicts (#4), and renders the music clip (#5). To start (#1), data is collected from a team's Git repository. The Git tree is then walked to identify commits that introduce and resolve conflicts.

Next, the Git data is passed to the sonifier module (step #2), which parses three key data elements: commit authors, days on which commits were made, and when conflicts occur.

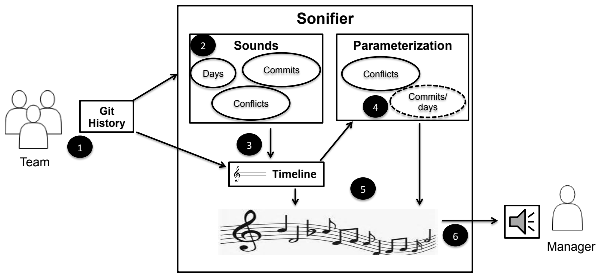

Figure 1: GitSonifier

Each developer is assigned a musical motif as an earcon that is created using unique timbre, pitches, and rhythm. This makes them easier to differentiate [4]. Each commit is represented by a measure of music because untrained listeners can intuitively tell when a measure begins and ends.

As a result, the amount of time the sonification plays to represent a day's worth of commits is longer or shorter depending on how many commits were made during the day. Accordingly, each day is represented by an earcon called a day separator. The day separator plays for one measure when one day ends and another begins. For instance, if developer A and developer B commit in that order one day, and then developer A makes another commit the next day, then developer A's earcon, developer B's earcon, a day separator, and developer A's earcon would play in that order for one measure each. On days when no activity occurs, the day separator is played multiple consecutive times to show the number of days that pass. These steps processing the Git data into a timeline are shown as step #3.

Conflict earcons are represented by a drum motif. Although still abstract, these are close to auditory icons, since there is a meaning behind the choice. We first identify when conflicts are introduced and resolved on the timeline. Then, we portray the number of unresolved conflicts during each commit and day separator by parameterizing the earcons (step #4). The more unresolved conflicts there are during a commit or day separator, the louder the earcons will be during the corresponding measure. Conflicts are played as an overlay onto the commit timeline. We do this because conflicts can exist in a Git project concurrently with commits being added. In future work, we can parameterize (step #4) and overlay additional data, such as the size of commits.

Finally, in step #5 we combine the timeline data and the conflicts into a music clip, which is exported and played (step #6) for a manager or software development professional.

To implement GitSonifier, we created `.wav` files for our earcons. Each file is one measure long, making it easy to combine them. We wrote a program to parse the Git data. We then use Beads, a Java library, to combine the sounds on the timeline [1]. It can combine sounds both sequentially, playing commits and day separators in order, and concurrently, playing commits and conflicts at the same time. Songs generated by GitSonifier are played at a tempo of 100 beats per minute. In our informal experience, this is fast enough to be enjoyable, but slow enough to be understood.

4. USER EVALUATION

To evaluate whether sonification is an effective means for portraying development history, there are several questions that need to be answered. For example, are users able to dis-

[1]GitSonifier sounds and experimental data are on our website: `http://cse.unl.edu/~myra/artifacts/NIER15/`

tinguish information encoded in a sonification? How can we leverage the multi-dimensionality of music to portray complex data? How much complexity can be encoded in a sonification so that users are able to parse the different types of information? Are there any length restrictions on the sonifications, so that users are not overwhelmed?

As an initial step, we perform an exploratory study to evaluate the basic assumption that developers can distinguish a set of information encoded in a sonification. Before we evaluate whether users can listen and comprehend the music while performing other tasks, we need to first determine whether sonification is feasible. We ask the following three research questions.

RQ1: How well do participants interpret the sounds representing a Git history?

RQ2: How efficient is the use of sonification for understanding a Git history?

RQ3: What was the participants' evaluation of sonification?

Participant Characteristics. We recruited six participants who have experience working in teams and using version control systems. They have experience in Git, SVN, and CVS, ranging from 1.5 months to 10 years. Most have experience working in small teams. Three have no music background, whereas others have between 10 - 25 years of experience in music. Four are male and two are female.

Study Design. We used GitSonifier to create a soundclip of the history of a GitHub project, Voldemort[2], which spans a 2-day development period and includes 3 developers and 1 conflict. This resulted in a 26-second long clip. We then created ten variants of this clip by adding or removing users, days, and conflicts from the history. Some variants change only one parameter, while others change two or all three parameters. Each variant includes 1 to 3 days, 2 to 4 developers, and 0 to 3 conflicts. Each sonification (variant) is 24-29 seconds long so that there is enough data encoded, but at the time same it is feasible to complete a set of 10 clips in the study time period. Of the 10 clips, one is the same as the original, and two are duplicates.

Participants were trained on what earcons meant in the sonification of the original data (from here on, we call it the training clip). They then had to complete a brief quiz about the data in the training clip. They could review the training clip and the earcons multiple times. Participants had to answer all the questions correctly before proceeding.

Next, the participants completed a set of ten tasks (one per variant). Each task had 3 questions that asked if the training clip and the variant clip had the same, more or fewer (1) developers, (2) days and (3) conflicts. The training clip was included in the task description for review. We measured the times-to-completion of tasks and correctness scores per question. Tasks were presented in a random order to minimize learning effects. At the end of the experiment, participants filled out a survey on the usability of the sonification. Finally, there was an exit interview in which the researchers could ask about specific (anomalous) actions.

We had users compare clips to see whether they can understand and differentiate the meaning of the clips. This design lines up with our use case of a manager listening to a sonification each morning, listening for unusual patterns rather than just listening to isolated data.

[2]`https://github.com/voldemort/voldemort`

5. RESULTS

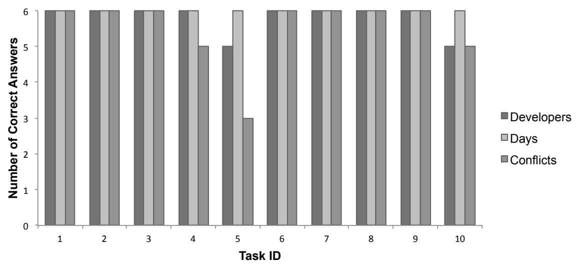

Figure 2: Correct Answers by Category

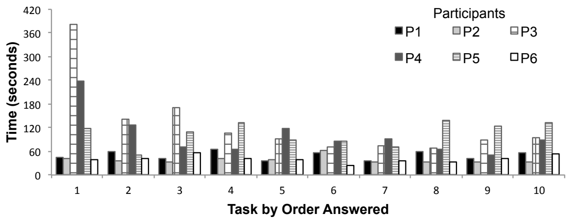

Figure 3: Amount of Time to Answer Each Question

Our results indicate that participants can easily understand the information encoded in a sonification (RQ1). Each participant answered 27 to 30 questions correctly, with an average of 29 correct answers per participant; four out of six got all answers correct. Figure 2 shows participants' correctness scores for each question per task. There were no discernable learning effects.

Figure 3 shows the time to completion of each task, sorted in the order in which tasks were attempted. Participants were able complete the tasks quickly (RQ2), and became faster as they progressed through the tasks. This is especially true for P3 and P4. The overall average time to complete each task across all participants was 1 minute, 11 seconds. The longest time for a task was 6 minutes, 21 seconds, which was participant's (P3) first task; he/she rapidly improved as he/she progressed through the tasks. On average participants took 40.3 seconds across all tasks. A key factor in how long it took to perform a task depends on how many times a participant listened to the sonifications. Three of the participants answered most of the questions (per task) after only listening to the (task) sonification once or twice, and without listening to the training clip. They were able to answer questions consistently in about 50 seconds.

The exit survey included questions on the usability of the sonification on a Likert scale from 1 to 5, with 5 being the highest score. For most questions, the median score is 5. This shows that participants liked the sonifications and found it easy to understand (RQ3) - see Table 1 for the mean (Avg) and median (Med) scores.

5.1 Discussion

Listening to Sonification(s). Four out of the six participants listened to the training clip at least once during the study. Two re-listened on their first and third tasks, respectively, to ensure that they remembered the original sonification. Another re-listened when he/she realized the

Table 1: Questionnaire Results

Question	Avg	Med
It was easy to tell the different sounds apart	4.8	5.0
It was easy to hear who each developer was	4.8	5.0
It was easy to hear the number of conflicts	4.7	5.0
It was easy to hear when days passed	4.7	5.0
The sound helped me understand the data	4.2	4.0
I would be interested in hearing the development data of my own teams' projects	4.5	4.5

current clip was the same as the training clip and wanted to confirm it. The fourth re-listened on questions 1 and 6 to double-check their work. When we asked participants why they did not listen to the training clip more often, they said that they had memorized the important information from the training clip. For example, one participant said, "*I kind of memorized the number and type of developers, the conflicts, and the number of days, so I didn't really need to [listen to the training clip again].*"

One participant (P5), with ten years of version control experience, but no musical background always listened to the current task both before and after answering each question (in a task) in order to be confident of his/her answers. This significantly added to the completion time.

Incorrect Responses. In Task 5, three participants got the question about conflicts wrong. The task's sonification had two conflicts, and the conflict that was introduced first was resolved before the second conflict was introduced. As a result, drums appeared, then stopped, then restarted. This was likely the most challenging task for the participants because they failed to notice the silence (for one measure) between the two conflicts' drums. This suggests that a single measure of silence may not be enough.

Effects of Background. All participants who had experience playing an instrument or composing music got a perfect score on each task, despite the task's difficulty. This suggests that people with music backgrounds had more success in interpreting sonifications. However, the effect size of this is not high: the lowest overall score that any participant got was 27 out of 30 questions correct.

We did not find any connection between experience and using sonification and there was no correlation between the participants' music experience and the time to completion for tasks. The two slowest participants (P3 and P4 in Figure 2) were non-native English speakers. They quickly improved their times as they progressed. We also did not find any effect of experience in version control on using sonification.

Sonification Effects. When considering developer sounds, two participants missed one question each. One of them misread the question and realized this mistake on his/her own later. Questions regarding conflicts had the most incorrect answers. One task (discussed above) had a short gap between two sets of conflict drums. 3 participants missed the "new" conflict. Two other questions regarding a conflict were missed as well, but these were on a case by case basis. All participants correctly answered questions about the number of days. The day separator sound may have stood out among the rest of the sounds because it was written in a different key and had a distinct motif. This might indicate that earcons that are very distinct from the rest of the sounds may work better. However, we need to consider how the different notes sound when put together, a series of distinct earcons may not be aesthetically pleasing. We will experiment overlaying different kinds and lengths of earcons in the future, especially for conflict earcons.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a novel approach and built a tool, called GitSonifier, for sonifying conflict history data so that project managers or development team members can easily comprehend it. We have applied this to conflict data from a Git repository and performed a formative user study to evaluate whether or not users can understand the content by listening to the resulting sound clips. Our study shows that the majority of users were able to differentiate the data between clips and thought that differences were easy to hear. As future work, we will expand our implementation of GitSonifier and will provide it as an Eclipse plug-in. We plan to perform larger user studies to compare with visualizations and to isolate the multitasking use case. We also plan to apply our sonification approach to additional areas where it has shown to have potential, such as testing and debugging.

7. ACKNOWLEDGMENTS

This work was supported in part by NSF grants CCF-1253786, IIS-1110916 and CCF-1161767.

8. REFERENCES

[1] Beads, 2015. `http://www.beadsproject.net/`.

[2] W. W. Gaver. The SonicFinder: An interface that uses auditory icons. *HCI*, 4(1):67–94, Mar. 1989.

[3] C. Henthorne and E. Tilevich. Sonifying performance data to facilitate tuning of complex systems: Performance tuning: Music to my ears. In *OOPSLA*, pages 35–42, 2010.

[4] T. Hermann, A. Hunt, and J. G. Neuhoff. *The sonification handbook*. Logos Verlag Berlin, 2011.

[5] K. Hussein, E. Tilevich, I. Bukvic, and S. Kim. Sonification design guidelines to enhance program comprehension. In *ICPC*, pages 120–129, May 2009.

[6] A. Kuhn, D. Erni, P. Loretan, and O. Nierstrasz. Software cartography: thematic software visualization with consistent layout. *J. of Soft. Maint. and Evo.: Research and Practice*, 22(3):191–210, 2010.

[7] S. McIntosh, K. Legere, and A. Hassan. Orchestrating change: An artistic representation of software evolution. In *CSMR-WCRE*, pages 348–352, Feb 2014.

[8] M. Ogawa and K.-L. Ma. Code_swarm: A design study in organic software visualization. *IEEE VCG*, 15(6):1097–1104, Nov 2009.

[9] A. Sarma, D. Redmiles, and A. van der Hoek. Categorizing the spectrum of coordination technology. *Computer*, 43(6):61–67, 2010.

[10] A. Stefik, K. Fitz, and R. Alex. Increasing fault detection effectiveness using layered program auralization. In *SERP*, Jan 2006.

[11] A. Stefik, K. Fitz, and R. Alexander. Layered program auralization: Using music to increase runtime program comprehension and debugging effectiveness. *ICPC*, pages 89–93, 2006.

[12] P. Vickers and J. L. Alty. Siren songs and swan songs debugging with music. *Com. ACM*, 46(7), July 2003.