# CocoViz with Ambient Audio Software Exploration

Sandro Boccuzzo and Harald C. Gall
Department of Informatics, University of Zurich, Switzerland
{boccuzzo, gall}@ifi.uzh.ch

## Abstract

*For ages we used our ears side by side with our ophthalmic stimuli to gather additional information, leading and supporting us in our visualization. Nowadays numerous software visualization techniques exist that aim to facilitate program comprehension. In this paper we discuss how we can support such software comprehension visualization with environmental audio and lead users to identify relevant aspects. We use cognitive visualization techniques and audio concepts described in our previous work to create an ambient audio software exploration (AASE) out of program entities (packages, classes ...) and their mapped properties. The concepts where implemented in a extended version of our tool called `CocoViz`. Our first results with the prototype shows that with this combination of visual and aural means we can provide additional information to lead users during program comprehension tasks.*

## 1 Introduction

Program comprehension is a major concern in software maintenance and evolution. Without effective tool support, the amount of data, the relationships between the entities, and out of date documentation make it almost impossible for engineers to maintain an accurate understanding of an evolving system. Because of the size and complexity of nowadays systems, even effective visual representations end up being overwhelmed with all the gathered information about a project. There is need to support the observers in leading them to find relevant aspects right away.

With the `CocoViz` project[1] we aim to enhance existing maintenance and evolution analysis methods to present a software system in an intuitively understandable visualization [4]. Beyond the visual presentation we started to think about ways to support and lead observers to relevant aspects. In [5] we described ways to support visual representations of software entities with aural representations. With this approach we were able to further investigate visual representations of a particular software entity and its properties without leaving the focus from the actual visualization.

In this paper, we describe how we extended our previous audio approach from a rather investigative support of a visualization to a more exploitive approach. We achieve this by creating an ambient sound out of a set of software entities with a particular dependency or relationship. In comparison with to our previous work where we created an aural representation for one particular software entity. The main contribution is an extended `CocoViz` audio-visualization approach, where we use environmental influenced aural feedback during navigation in a visualization. The described approaches were implemented in a extended version of our `CocoViz` Tool [3].

The remainder of this paper is organized as follows. Section 2 covers aspects to support software visualization, in regard to program comprehension and navigation tasks, with audio. In Section 3 we discuss our concept of creating ambient sound out of a set of related software entities. In Section 4 we present example scenarios based on the azureus[2] project. We address related work in Section 5 and summarize with our conclusions and future work in Section 6.

## 2 Support Software Visualization with Audio

One of the main concerns in a software visualization is to find relevant aspects in a complex system as fast as possible. Explorative software visualizations as [4] offer a variety of filters and customization opportunities to limit the amount of software entities in a view. Still quite often one finds himself with hundreds of potential entities. How can we assist the observer in such situations?

Popups or tooltips are concepts that are commonly used in situations, where an observer needs additional information on entities. The pitfalls of these concepts are that with every extra popup, the observer gets a more disturbed view on the general visualization. Particularly in explorative situations where usually there are more than 10 entities to further investigate popups / tooltips become a suboptimal approach. We prefer other visualization support.

[1]This work was partially supported by the Hasler Stiftung Switzerland.

[2]http://www.azureus.com/ last checked 11.2.2009

According to Pennington's [12, 13] bottom-up theory of program comprehension *e.g.,* a programmer focuses first on the basic structural entities. A fundamental component for an appropriate visualization support therefore is an adequate highlighting of basic structure units. After further investigating on fundamental components for program understanding and suggestions for facilitating software comprehension as [11, 10], we suggested extending interaction in software visualizations with an audio visual approach [5]. By using audio we solved the previous mentioned shortcomings and we were able to improve the navigation and program comprehension capabilities of our `CocoViz` approach in general.

## 3 Ambient Audio Concept

Our primer results with audio supported visualizations [5] focused more on an investigative support of a visualization. The results showed how we were able to get additional information on one particular software entity without losing focus on the current visualization state and task.

Encouraged from those results we looked for how we could extend the audio support to a more explorative modality. Our argument is that we want to help the observer in his exploration, by conciliating him through adequate audio sources and leading him in finding relevant aspects.

Compared to our previous approach were an audio representation was triggered with the selection of one particular entity, now we use audio to find a particular entity. Like in a movie or computer game, were the audio track forewarns the audience / player of upcoming plots. *e.g.,* in a triller a strident ambient sounds warns us of what we can expect next. We create similar ambient like sounds out of set of software entities with particular dependencies or relationships.

The ambient sounds are constructed with the same concepts (selection of classes, filtering, metric clusters, ...) used in our previous work [5]. The metric clusters are mapped to adequate Zwicker parameters [16] or the frequency / volume of a particular sounds *e.g.,* a bubble sound. Table 1 shows a set of possible case scenarios where ambient audio can be used together with possible mapping criterias.

To trigger an ambient audio source we use common concepts like a head-up display know form video-games, a key combination or the shift key in accompaniment with a mouse. This bring up an exploration marker, that personifies an observers navigation in the visualization space and pictures the position of the observers ears. While the marker is moved around the visualization (Table 2), we take its current position and use surround sound techniques to clarify which audio representations is currently played. The observer perceives the audio sources and can adequately adjust his navigation in the visualization.

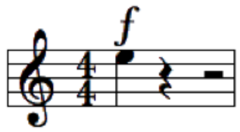
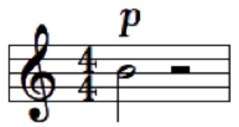
| Scenarios | Ambient sound | Mapped criteria |
|---|---|---|
| **Many changes** |  | Zwicker parameters e.g., high volume or loudness |
| **Few changes** |  | Zwicker parameters e.g., low volume or loudness |
| **Code smells** | | a particular Sound e.g., a bubble sound with low frequency |
| **Disharmonies** | | Zwicker parameters or a particular sound e.g., a bubble sound with high frequency |

**Table 1. Case scenarios and possible mappings for ambient audio**

Based on the ambient sound we are able to lead an observer on a trail, we call audio exploration path (AEP). We preserve an audit trail with keyframes of this followed AEP, for an eventual backtracking to interesting exploration positions.

An ambient sound currently depicts either a package or a set of entities. The first is useful to call attention on present code smells or deficiencies in one package compared to other packages. We explain this case during an example in section 4. The second deals about case scenarios where attention is brought to entity sets with some sort of cohesion (*e.g.,* change couplings [8], number of bug reports, ...) between each others. Suppose we are looking for certain instabilities in our system or critically affected system components. We can build our visualization with a layout involving similarities as used by Fischer *et al.* in [7]. Using ambient audio on top of that layout, mapped to metrics like *number of bug reports* or *number of recent changes* offers us a walkthrough on our software visualization were we simply hear about our critical hot spots. In particular thanks to ambient audio wound up a set of entities with a particular dependencies, we get attentive as well on cases where small individual entities, because of their relation, represent a critical situation to a component.

The AASE approach enables us to extend the audio support for software visualizations, with assisted navigation, a space annotated with sounds (AEP) and perception of entities with dependencies to a more explorative modality.

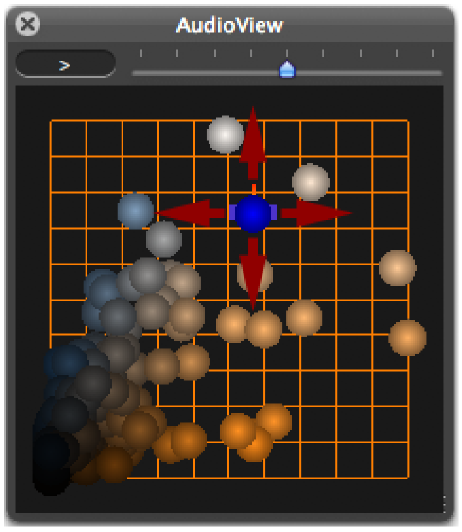


**Table 2. Head-up display with exploration marker**

## 4 Example Scenarios

To demonstrate our AASE concepts we implemented them in our `CocoViz` tool. For the evaluation we use an ambient sound similar to a bubble sound, everyone is familiar with from their daily live. We alter the volume and frequency based on the mapped metrics of each set of software entities. The different acoustic feedbacks would sound like the different stages from still to boiling water. The higher the volume and frequency the more the perceived acoustic feedback resembles the one of boiling water.

We are currently evaluating our approach with different versions of the azureus dataset and versions of a commercial web framework. In the following we demonstrate on a simple understandable task how ambient audio exploration can help leading the user to the relevant entities. The used evolutionary data set consists of the azureus 2 project. We choose 3 major releases from v3.0.5 to v3.1.1. The metrics were calculated per release.

Figure 1 shows a Hotspot-View as explained in [4] of all the packages in azureus version 3.0.5. In our example scenario we are looking for packages which include classes that where recently changed. Solving the task in a visual approach we can change the color mapping to represent where changes were maid. By triggering the ambient audio though we can simply leave the entities in the current color mapping and just moving around the exploration marker, while listen to the audio feedback. In Figure 1 moving the exploration marker towards (a) we would here a ambient sound similar to a soft sea, approaching (b) we would perceive a more stormy ambient sound, and close to (c) the ambient sound resembles the one of boiling water. In other words we can easily argue that in (a) there where only little changes since the last release, some where made in (b) and quite some changes were made to the classes in package (c).

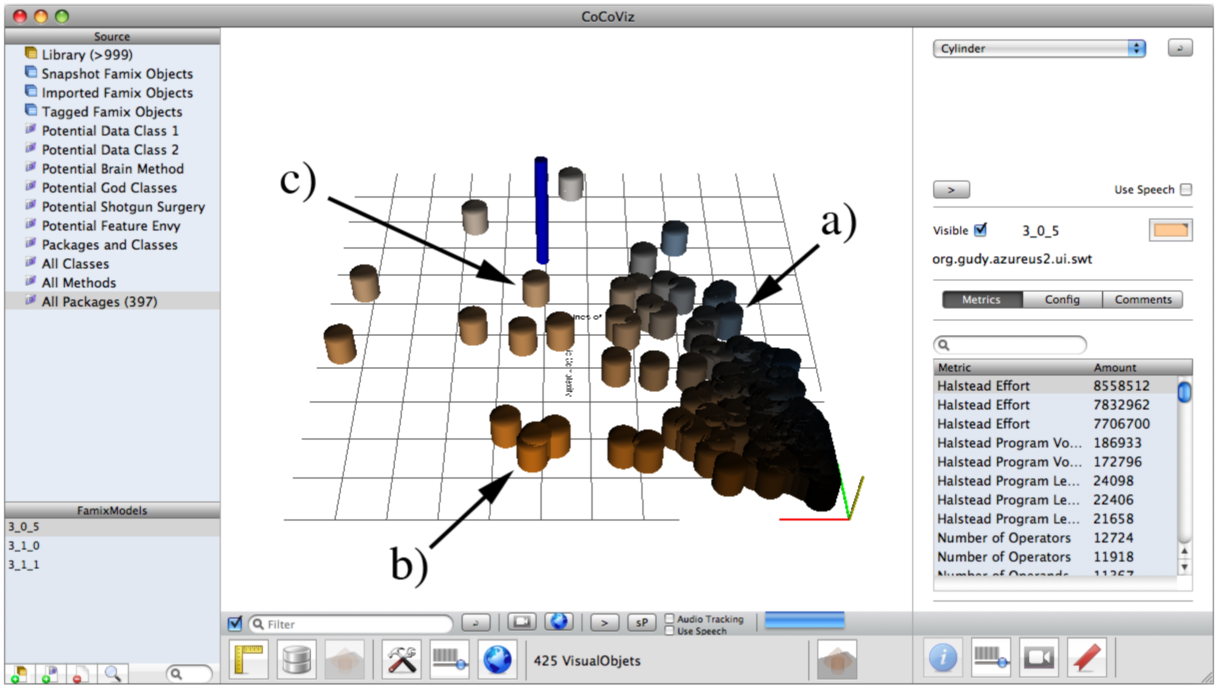
## 5 Related Work

The goal of Software visualization is to represent the complex context of today's software projects. Most visualization methods use a graphical representation of data. In the past few years a variety of approaches dedicated to software visualization and software reengineering emerged.

**Metrics visualization** describe a software state or situation. Metrics describe a specific software entity and are not part of a hierarchy. The goal of these approaches is to show aspects of a software by visualizing the representing metrics. An exponent of this approach is Lanza and Ducasse's Polymetric Views [9]. In their concept they display the software entities based on their metric values as the position, the height, the width and the color of a rectangular shape.
AASE uses similar concepts to create the ambient audio environment and extends those works with an interactive approach where a viewer analyses the software in walking through the views to identify relevant aspects.

**Audio supported visualization** To our best knowledge we found only little work related to our audio approach in supporting software visualization. However there is work done in the context of software analysis and auditory display.
A good introduction to the various approaches present in the field of auditory representation is provided by Vickers in [15].
Brown and Hershberger in [6] use audio for algorithm animations. They enhanced the Zeus algorithm animation systems using a MIDI synthesizer and introduced the use of colour and sound in algorithm animations.
Baecker *et al.* in [1] use audio to provide programmers with debugging and profiling feedback without disturbing the integrity of the graphical interface. According to them audio may be a more salient representation for certain types of program information like repetitious patterns in control flow and nonlinear sequences of variable values.
Berman and Gallagher in [2] present techniques to listen to program slices that help software developer in undertaking program comprehension activities.
Work with audio in software analysis has been done by Stefik *et al.* in [14]. In their work they use aural feedback to sonify computer code as an aid to non-sighted programmers.

Compared to our AASE approach the others distinguishes itself mainly as they focus on tracking the value of state variables and control flow during debugging or visualizing algorithms. Meanwhile AASE's focus is more in supporting the interaction within a visualization.



Visible ✓ 3_0_5

org.gudy.azureus2.ui.swt

| Metric | Amount |
|---|---|
| Halstead Effort | 8558512 |
| Halstead Effort | 7832962 |
| Halstead Effort | 7706700 |
| Halstead Program Vo... | 186933 |
| Halstead Program Vo... | 172796 |
| Halstead Program Le... | 24098 |
| Halstead Program Le... | 22406 |
| Halstead Program Le... | 21658 |
| Number of Operators | 12724 |
| Number of Operators | 11918 |

**Figure 1. Hotspot view of the packages present in azureus v3.0.5 with ambient audio exploration marker**

## 6 Conclusions & Future Work

In this paper we discussed improvements to the perception of relevant aspects in evolving software projects. We proposed an ambient audio software exploration (AASE) approach that assists in the software visualizations investigation and perception of situation where individually irrelevant entities with dependencies become relevant.

We are currently preparing a user study to get substantial data on the benefits of audio-supported software exploration. The evaluation is performed on a large set of software projects and against other visualization approaches to document situations where audio feedback offers substantial advantages over more traditional approaches.

Future work aims to consider more sophisticated audio algorithms and tailor the audio feedbacks more towards specific general program comprehension tasks

## References

[1] R. Baecker, C. DiGiano, and M. Aaron. Software visualization for debugging. *Commun. ACM*, 40(4):44–54, 1997.

[2] L. I. Berman and K. B. Gallagher. Listening to program slices. In *Proc. Int'l Conf. on Auditory Display*, 2006.

[3] S. Boccuzzo and H. C. Gall. Cocoviz: Supported cognitive software visualization. In *Proc. Working Conf. on Reverse Eng.*, 2007.

[4] S. Boccuzzo and H. C. Gall. Cocoviz: Towards cognitive software visualization. In *Proc. IEEE Int'l Workshop on Visualizing Softw. for Understanding and Analysis*, 2007.

[5] S. Boccuzzo and H. C. Gall. Software visualization with audio supported cognitive glyphs. In *Proc. Int'l Conf. on Softw. Maintenance*, 2008.

[6] M. Brown and J. Hershberger. Colour and sound in algorithm animation. In *Proc. IEEE Workshop on Visual Languages*, page 5263, 1991.

[7] M. Fischer, M. Pinzger, and H. Gall. Populating a release history database from version control and bug tracking systems. In *Proc. Int'l Conf. on Softw. Maintenance*, pages 23–32, 2003.

[8] H. Gall, K. Hajek, and M. Jazayeri. Detection of logical coupling based on product release history. In *Proc. Int'l Conf. on Softw. Maintenance*, pages 190–198, Nov 1998.

[9] M. Lanza and S. Ducasse. Polymetric views — a lightweight visual approach to reverse engineering. *IEEE Trans. on Softw. Eng.*, 29(9):782–795, 2003.

[10] R. Mosemann and S. Wiedenbeck. Navigation and comprehension of programs by novice programmers. In *Proc. Int'l Workshop on Program Comprehension*, page 79, 2001.

[11] M. J. Pacione. Software visualisation for object-oriented program comprehension. In *Proc. Int'l Conf. on Softw. Eng.*, pages 63–65, 2004.

[12] N. Pennington. Comprehension strategies in programming. In *In G. M. Olson, S. Sheppard & E. Soloway, Eds. Empirical Studies of Programmers: Second Workshop*, pages 100– 113, 1987.

[13] N. Pennington. Stimulus structures and mental representations in expert comprehension of computer programs. In *Cognitive Psychology*, pages 295–341, 1987.

[14] A. Stefik, R. Alexander, R. Patterson, and J. Brown. Wad: A feasibility study using the wicked audio debugger. In *Proc. IEEE Int'l Conf. on Program Comprehension*, pages 69–80, 2007.

[15] P. Vickers. External auditory representations of programs: Past, present, and futurean aesthetic perspective. In *Proc. Int'l Conf. on Auditory Display*, 2004.

[16] E. Zwicker, H. Fastl, and W. M. Hartmann. "psychoacoustics: Facts and models". *Physics Today*, 54:64–65, 2001.