Software Evolution Sonification: Why and How

Pedro O. Raimundo e Sandro S. Andrade

Grupo de Pesquisa em Sistemas Distribuídos, Otimização, Redes e Tempo-Real (GSORT)
Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBa)
Av. Araújo Pinho, nº 39 - Canela - Salvador - BA - CEP: 40.110-150

{pedrooraimundo, sandroandrade}@ifba.edu.br

Abstract. *Program comprehension is one of the most challenging tasks undertaken by software developers. Achieving a firm grasp on the software's structure, behavior and evolution directly from its development artifacts is usually a time-consuming and challenging task. Software visualization tools have effectively been used to assist developers on these tasks, motivated by the use of images as outstanding medium for knowledge dissemination. Under such perspective, software sonification tools emerge as a novel approach to convey temporal and concurrent streams of information and have been proven to perform remarkably well due to the their inherently temporal nature. In this work, we describe how software evolution information can be effectively conveyed by audio streams and propose a tool for sonification of software repositories.*

Resumo. *Compreender programas de computador é uma das tarefas mais desafiadoras que desenvolvedores precisam realizar. Adquirir entendimento sólido acerca da estrutura, comportamento e evolução de um software através da investigação direta dos seus artefatos de implementação é um processo demorado e desafiador. Ferramentas para visualização de software têm sido utilizadas com sucesso para este fim, motivadas pela efetividade do uso de imagens como mecanimos para disseminação de informação. Sob esta perspectiva, ferramentas para sonorização de software emergem como uma forma inovadora de comunicação de informações temporais e concorrentes, se mostrando notadamente eficazes devido à sua natureza inerentemente temporal. Neste trabalho, são apresentados os fundamentos para representação – como fluxos de áudio – de informações sobre a evolução de um software e uma ferramenta para sonorização de repositórios de software é proposta.*

1. Introduction

Comprehending computer programs is a notoriously difficult task that involves gathering information from diverse sources (source code, documentation, runtime behavior, version history, just to mention a few) and gets progressively harder as the program's size and complexity grow [Stefik et al. 2011]. Synthesizing that information to tackle the development process effectively and efficiently is an endeavor that requires time, experience and, more often than not, peer support.

Regardless of the abstraction level (code, design, architecture) at which the developer is going to address the problem at hand, tools are usually employed in order to help the decision making and understanding. Such tools range from built-in Integrated

Development Environment (IDE) helpers and code metric viewers to complex software visualization solutions, focusing on conveying information to the user through the computer screen using tables, charts, drawings or animations. While such approaches are helpful in the comprehension process, exploring aural representations of software structure and behavior has been shown to excel at representing structural [Vickers 1999], relational [Berman 2011] and parallel or rapidly changing behavioral data [Sonnenwald et al. 1990] in an non-invasive and uncluttered fashion with high apprehension rate.

In this paper, we present a novel approach to convey information about software evolution by exploiting sound's uniquely temporal nature and aural events such as melody, harmony, rhythm and noise. Different events are used because each event has a diverse impact on the listener and they can be mixed and matched together to convey different streams of information, as long as not being confusing or intensely overwhelming.

We make two key contributions in this paper. First, we propose the foundations for a tool in which the evolutionary aspects of a piece of software can be represented as sound streams, in an unobtrusive and noninvasive manner. We build upon the existing research on the fields of auditory display and software comprehension through sonification. Second, we explore a research field that, to the best of our knowledge, has yet many research challenges, raising awareness to this particular field and possibly creating a new forum for idea exchange with a lot of untapped multidisciplinary potential.

2. Software Evolution Sonification

Software Evolution deals with the progressive changes that occur to a computer program over the time, in order to adapt itself to real world demands. That differs from software maintenance because whilst maintenance is closely related to minor tweaks and bug fixes, evolution focuses on adapting and migrating the software systems in order to keep their initial efficacy.

While significant studies have already been developed to auralize structural and behavioral software aspects, only modest contributions have been made in the way of sonifying evolutionary information. The aurally extended version of CocoViz [Bocuzzo and Gall 2008] does this by tracking how much the source code of a selected entity of interest has changed across two versions and giving the user audio feedback depending on a pre-defined threshold. Although that is a good start, it by no means explore the full potential of auralizations to represent software evolution.

Our approach takes this idea one step further, by extracting software metrics from the elected versions of the software and using sound to convey information about the changes in these metrics and overall structure and architecture of the software over time. Such information can theoretically be discretized by number of commits, timespan or a weighting of both. Studies are still being conducted to decide on a particular discretization criteria.

3. Proposed Approach

Our approach to sonifying software evolution involves three main steps: source-code retrieval, data extraction and sound synthesis, all of which are detailed on the following paragraphs.

Software evolution is an inherently temporal phenomenon, as such it's fundamental to track the changes in a program's structure and code-metrics across a period of time to achieve a proper representation of it. This involves retrieving snapshots of the software's source code at different versions. At the time of this study, no technology seems more appropriate to handle this particular task than batch processing with source control tools such as Git and Subversion. By doing that, we automatically maintain and store changes across software revisions on a repository. By using such tools, multiple versions of the software's source code can be stored locally for further analysis.

Once local repositories are created from retrieved versions of the program's code, it's possible to parse the source files and retrieve the software's logical structure using one of the many available tools for this purpose. In our preliminary studies we considered GCC-XML [GCC-XML 2012] and Clang [Clang 2013] as potential tools for this task. Further experimentations and impressions make us lean towards Clang as the superior tool, mainly due to the fact that GCC-XML doesn't extract information on the method's bodies, necessary for calculating many of the commonly used code-metrics. However, both tools are still under consideration due to factors such as reliability, ease-of-use and performance.

After extracting the chosen metrics from the retrieved versions, a comprehensive mapping can be elaborated between the software metrics collected and the various aural events that will be defined according to the guidelines already present in the software sonification literature. Both musicality [Vickers and Alty 1998] and comprehensibility [Stefik et al. 2011] of the auditory cues should be taken into account.

Once the mapping process is complete, a musical score will be generated according to one of the available sonification technologies. Exploratory research indicates that CSound [Vercoe 1992], the Structured Audio Orchestra Language / Structured Audio Score Language (SAOL/SASL) [Scheirer 1998] suite, and Lilypond are relatively proven and well-developed technologies for audio synthesis and rendering, while in-depth research brought up NSound [Hilton 2014] as a viable alternative to the first three. While being relatively new and obscure, NSound has the advantage of being completely object-oriented and built from the ground-up with C++ support. Again, the choice between one of these technologies is going to involve a compromise between reliability, ease-of-use and performance, with the added criteria of extensibility.

4. Conclusions and Future Work

The technologies and the methodology described in this paper lay the foundations for a tool that will be able to generate comprehensive auralizations that convey meaningful information about the software's evolution. Such information, previously displayed only in a spatial medium via complex graphical representations, can be auralized as a musical score. Studies [Berman 2011, Vickers and Alty 2000] have already shown that even those with little musical background can quickly grasp the concepts conveyed by aural representations after proper training. Thus, the apprehension and retention rates for this approach may be inferred as superior to that of a visual-based approach.

A secondary contribution of this work is that it touches upon a field that's still not explored on the current body of knowledge. With good visibility and promotion, it fosters the beginning of a new research area and opens up a new forum of discussion

that welcomes researchers from areas such as computer science, music, education and communication.

Along with this paper, a systematic literature review is on its finishing stages and future publications are already in sight, building upon this work with the final methodological steps and validation experiments to assert the tool's effectiveness and efficacy.

References

[Berman 2011] Berman, L. (2011). *Program Comprehension Through Sonification*. PhD thesis, Durham University.

[Bocuzzo and Gall 2008] Bocuzzo, S. and Gall, H. (2008). Software visualization with audio supported cognitive glyphs. In *2008 IEEE International Conference on Software Maintenance*.

[Clang 2013] Clang (2003-2013). C language family frontend for llvm. http://clang.llvm.org/index.html. Accessed: 08/04/2014.

[GCC-XML 2012] GCC-XML (2002-2012). Xml output for gcc. http://gccxml.github.io/HTML/Index.html. Accessed: 08/04/2014.

[Hilton 2014] Hilton, N. (2014). Nsound. http://nsound.sourceforge.net. Accessed: 08/04/2014.

[Scheirer 1998] Scheirer, E. D. (1998). The mpeg-4 structured audio standard.

[Sonnenwald et al. 1990] Sonnenwald, D. H., Gopinath, B., Haberman, G. ., III, W. M. K., and Myers, J. S. (1990). Infosound: An audio aid to program comprehension. In *System Sciences, 1990., Proceedings of the Twenty-Third Annual Hawaii International Conference on*.

[Stefik et al. 2011] Stefik, A., Hundhausen, C., and Patterson, R. (2011). An empirical investigation into the design of auditory cues to enhance computer program comprehension. *International Journal of Human-Computer Studies*.

[Vercoe 1992] Vercoe, B. (1992). The canonical csound reference manual.

[Vickers 1999] Vickers, P. (1999). *CAITLIN : implementation of a musical program auralisation system to study the effects on debugging tasks as performed by novice Pascal programmers*. PhD thesis, Loughborough University.

[Vickers and Alty 1998] Vickers, P. and Alty, J. (1998). Towards some organising principles for musical program auralisations. In *Proceedings of the Fifth International Conference on Auditory Display*.

[Vickers and Alty 2000] Vickers, P. and Alty, J. L. (2000). Musical program auralisation: Empirical studies. In *In Proceedings of International Conference on Auditory Display (ICAD*, pages 157–166.