Versioning and Annotation Support for Collaborative Mapping Design

Johnty Wang
Input Devices and Music Interaction Lab/CIRMMT
McGill University
Johnty.Wang@mcgill.ca

Joseph Malloch
Graphics and Experiential Media Lab
Dalhousie University
Joseph.Malloch@dal.ca

Stéphane Huot
Mjolnir Group
Inria Lille
Stephane.Huot@inria.fr

Fanny Chevalier
Mjolnir Group
Inria Lille
Fanny.Chevalier@inria.fr

Marcelo Wanderley
Input Devices and Music Interaction Lab/CIRMMT
McGill University
Marcelo.Wanderley@mcgill.ca

ABSTRACT

This paper describes a versioning and annotation system for supporting collaborative, iterative design of mapping layers for digital musical instruments (DMIs). First we describe prior experiences and contexts of working on DMIs that has motivated such features in a tool, describe the current prototype implementation and then discuss future work and features that are intended to improve the capabilities of tools for new musical instrument building as well as general interactive applications that involve the design of mappings with a visual interface.

1. INTRODUCTION

A crucial component of the digital musical instrument (DMI) [1] building process involves the mapping of sensor or gesture input signals from the musician to relevant synthesis parameters, and is described in detail by [2]. While there are many different approaches to the design and implementation of DMI mapping, for many designers, a common part of the process involves the use of graphical user interfaces where the connection of signals can be observed and manipulated visually. In this paper we use personal experiences of building DMIs in a variety of contexts to motivate two main features for a mapping tool: a deeply integrated graphical versioning and comparison tool, and rich annotation features that go beyond the text-based tagging commonly employed by existing versioning systems. While our focus here is to extend the capabilities of tools for creative design of interactive systems [3], we acknowledge the fact that a good tool is not the end-all solution in itself, but something that can help in the process. In this paper we first present background and related work, followed by the motivations for and details on the implementation of our system, and finally a discussion on the limitations of the current work and future steps.

2. EXPERIMENTATION AND ITERATION

Over the years we have been involved in the conception, design, use and evaluation of a large number of DMIs, including some that have been actively performed for 10 years and have toured internationally. Throughout the development of these instruments, *iteration* has proven to be the most important aspect. Many designers take a holistic approach their DMI concept: designing not just the sensing system or sound synthesizer but also gesture vocabularies, notation systems, and lighting or video systems that complement their idea of performance with the instrument. This approach is exciting, but also brings with it difficult challenges, since the various aspects of the instrument are interdependent - the physical instrument, sensing system, gesture, and sound cannot be changed individually without affecting the others. While the overall concept of the instrument can inform process, the actual implementation often requires iteration. Usually, the most important advances are due to a complex system *surprising* the designer, inspiring them to modify their vision of the final instrument.

It is important that our tools for supporting mapping design support iterative workflows, and also make it easy to do "exploratory" development which might result in serendipitous discovery of new directions for the instrument.

2.1 Collaboration

If iteration is important in solo development, in collaborative design contexts it is essential, especially in groups where the roles of hardware designer, composer, and performer are played by different collaborators. In this case it is no longer feasible for a single united vision of the instrument (and its performance practice and repertoire) to exist. The collaborators inevitably use different tools, have different approaches, and in interdisciplinary projects even different vocabularies for discussing interaction. During the course of several long-term projects [4], we started building approaches and tools to try to support interdisciplinary collaborations based on a few principles:

- **Don't enforce representation standards:** Encouraging each collaborator to represent their system component in the strongest way possible helps maintain the benefits of their specific domain-knowledge to the project. It also helps with modularity!

- **Make things freely connectable:** If we encourage strong but idiomatic representations of system components, flexibility must come from the ability to freely interconnect them. Our approach is for *models* to "live" only in devices rather than influencing connection or communication protocols; representations *on the wire* should be low-level, allowing basic compatibility between drastically different devices.

- **Don't dumb it down, but...:** Musicians possess highly technical knowledge, and are usually capable of learning surprising amounts of electrical engineering, digital signal processing, and software design in short periods of time. This means that instrument designers do not need to hide low-level technical properties of the system from the performer, however the nature of the system representation should be questioned: is there a technical representation from the performer's perspective that might serve better?

The software library *libmapper* [4] and the surrounding tools are the result of this design philosophy and have supported a large number of interdisciplinary research/creation projects. These tools form a discoverable, distributed network of mappable devices; they encourage (but do not enforce) modular, strong representations of system components; they glue together different programming languages and environments; and they aid the process of rapid experimentation and iteration by making every signal compatible and allowing drag-and-drop style mapping design.

3. RELATED WORK

There are a fairly large number of existing tools for supporting the task of mapping DMIs or similar interactive systems. Some, like the Mapping Library for Pure Data [5] and the Digital Orchestra Toolbox [1] try to provide building blocks for assembling mapping structures; others, like MnM [6] and LoM [7] support specific multidimensional mapping schemes, usually learned from examples. A variety of platforms and environments are also used for designing and implementing DMI mapping, including ICon [8], junXion [9], and Jamoma [10]. Finally, some commercial DMIs have dedicated instrument-specific mapping systems, such as the Continuum Editor [11], 2PIM [2] , Karlax Bridge [3] , and EigenD [4]

3.1 Version Control Systems

The core features of this work is related to two particular aspects of version control: first, the comparison between two different versions of a document [12] and second, the management of previous versions. While the workflow of most modern source control systems are centred around these two issues, they mostly focus on text based data. When the data has a graphical view, or is based on a translation of text data, a text-based "diff" tool is no longer sufficient. Dragicevic et al. [13] describes the process of implementing a comparison system via computing the visual output and then rendering the transition between versions using animations, which preserves the mental model of the user better when there is a translation process is involved.

The environment implemented by Rekall [14], designed to serve both as a "score" as well as documentation for a theatrical work, contains features that are relevant to our application since it contains a visual timeline as well as a rich annotation interface, but is intended for a very different purpose of documenting the timeline progression of a piece of work, as opposed to being able to preview/revert to prior states.

4. SUPPORTING EXPLORATION, BUILDING COMPLEXITY

4.1 Problems Identified

Through the process of working with DMIs in a variety of contexts as described earlier, we have identified the following issues when it comes to the design of mappings:

- **Fear of losing current state:** By not keeping a constant stack of actions and having to manually manage incremental versions, the fear of moving away from a desirable configuration when experimenting inhibits exploration.

- **Difficulty indexing, saving, finding, restoring and talking** about previous mapping designs limits number of options that can be reasonably compared.

- **Support for complexity:** Without effective comparison between mapping configurations, it is difficult to create complex mappings by combining mapping vignettes generated in workshop settings.

4.2 Proposed Solutions

Based on the issues identified above, we have proposed the following solutions:

- ***Distributed Components*:** Using the software library *libmapper* [4] enables simple composition of new or existing system components, and its distributed nature supports collaboration (For example, an arbitrary number of user interfaces can be actively viewing and editing the system at the same time, with different visual representations of the system).

- **Version tracking:** Ability to store previous states as to encourage experimentation while ensuring that no valuable work is lost.

[1] http://idmil.org/software/dot
[2] http://www.pucemuse.com/
[3] http://www.dafact.com/
[4] http://www.eigenlabs.com/

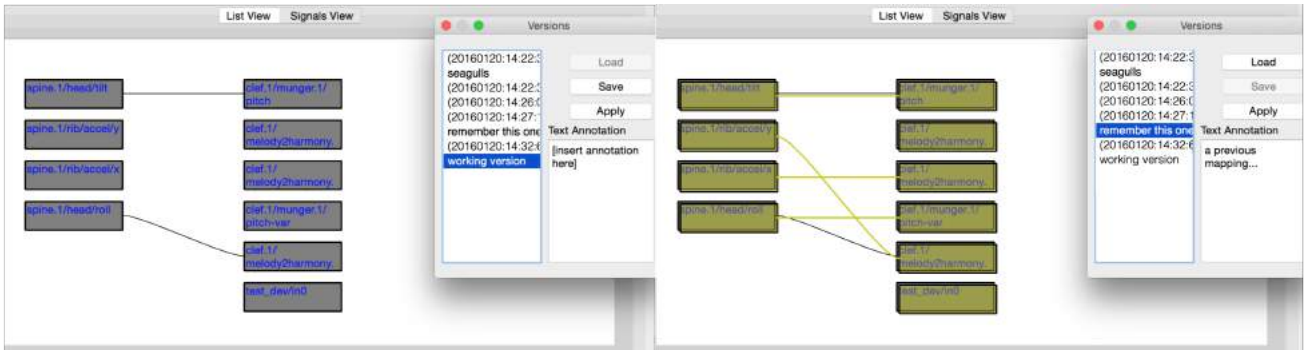

Figure 1. Screenshot showing working version (left), and a comparison of a previous version (right) when it is selected from the interface showing. Note: the actual rendering of the preview overlay has been offset in the application to show up better on a printed page in grayscale.

- **Tagging:** For identifying previous versions. Can be basic text annotations, but also multimedia content that may reflect the creative process, or even recordings of the output of the mapping.
- **Graphical tools:** To realize the above features of tagging, looking at and searching through versioning history, loading and previewing previous versions and creating new ones.

The following section describes the first software application we have implemented to realize the above.

5. IMPLEMENTATION

We have built a graphical tool for *libmapper* that manages distributed mappings between signals. The C++ QT Framework was chosen for its widespread usage, cross platform support, user interface and data model libraries as well as easy interface with the *libmapper* codebase that was written in C/C++.

5.1 Software Features

The mapping tool has the following main features:

- A *libmapper* interface that allows querying of available devices and signals on the network, and interactively modify the mapping configurations
- A user interface that displays the device signals and allows the user to create and break mappings between output and input signals
- A user interface for displaying and previewing previously stored versions of the mapping, along with annotations as well as saving new versions of the mapping

Figure 1 shows screenshots of the application in use and displays two states of operation. The left shows the current "working version" of the mapping, and the right when a previous version is being previewed. Since the *libmapper* network is a distributed system, a third layer, showing the actual status of the live network, can be overlaid on top and synchronized with the working version (not currently shown in the screenshot).

6. FUTURE WORK

In addition to continuous development in refining the implementation described above, there are a few additional features we are currently considering which would increase the capabilities of the tool, and provide interesting possibilities.

- **Constant storage of state changes**: In addition to explicitly stored states, a stack of all user input could be stored into the versioning system which essentially creating an undo-redo loop that is always stored. Versioning system backends such as **.git** [5] provides efficient ways of handling this kind of incremental changes in data.
- **More featured tagging with data snippets, other media**: by recording segments of input signal streams and associating them with particular versions, it would be possible to "replay" the input at a given point in time. This also allows alternative mappings to be applied after the fact, which allows experimentation to occur without having direct access to live inputs, which may be a limited resource in collaborative projects involving many individuals. Using media tags allow more expressive annotation options.
- **Advanced Query features**: by applying analysis on the input stream and classifying/segmenting relevant gestures and activities, a prior version could be queried using gestural input. Also, queries using recordings of the output can also be very useful in a collaborative setting as it does not require in depth technical knowledge of the mapping itself.
- **Additional visualizations of the mapping**: Using different data visualization techniques to visually represent the connections between signals may provide additional insights not seen by one particular view. This also may include revealing changes in the parameter processing and transformation features of libmapper, in addition to signal connections themselves.

[5] `http://www.git-scm.com/`

7. CONCLUSION

We have discussed the motivation for integrating a versioning control and annotation system for visual mapping tools for DMI design based on prior experience working in a variety of contexts. Being able to effectively store, preview, and reload previous configurations of a system addresses some of the challenges of this creative process by reducing risks, encouraging exploration, and improving the management of configuration files. A newly developed mapping tool for *libmapper*, a well used system for connecting signals employed for the design and implementation of DMIs, is implemented with these features and described. These solutions, while motivated by our own experience, also correspond to the findings by others in the general case [3] for "rethinking user interfaces" for general creativity support tools in general as well as which includes *history keeping, exploratory search, visualization, collaboration and composition*. Such views are also echoed when exploring the development of tools to support computer music [15].

Acknowledgments

The primary author is supported by a NSERC-FRQNT Industrial Innovation Scholarship with Infusion Systems, and the Centre for Interdisciplinary Research in Music Media and Technology.

8. REFERENCES

[1] E. R. Miranda and M. M. Wanderley, *New digital musical instruments: control and interaction beyond the keyboard*. AR Editions, Inc., 2006, vol. 21.

[2] A. Hunt, M. M. Wanderley, and M. Paradis, "The importance of parameter mapping in electronic instrument design," *Journal of New Music Research*, vol. 32, no. 4, pp. 429–440, 2003.

[3] B. Shneiderman *et al.*, "Creativity support tools: Report from a U.S. national science foundation sponsored workshop," *International Journal of Human-Computer Interaction*, vol. 20, no. 2, pp. 61–77, 2006.

[4] J. Malloch, S. Sinclair, and M. M. Wanderley, "Distributed tools for interactive design of heterogeneous signal networks," *Multimedia Tools and Applications*, vol. 74, no. 15, pp. 5683–5707, 2014.

[5] H. C. Steiner, "Towards a catalog and software library of mapping methods," in *Proceedings of the 2006 conference on New interfaces for musical expression*. IRCAM—Centre Pompidou, 2006, pp. 106–109.

[6] F. Bevilacqua, R. Müller, and N. Schnell, "Mnm: a max/msp mapping toolbox," in *Proceedings of the 2005 conference on New interfaces for musical expression*, 2005, pp. 85–88.

[7] D. Van Nort and M. M. Wanderley, "The LoM mapping toolbox for Max/MSP/Jitter," in *Proceedings of the International Computer Music Conference*, New Orleans, USA, 2006.

[8] P. Dragicevic and J.-D. Fekete, "The input configurator toolkit: towards high input adaptability in interactive applications," in *Proceedings of the working conference on Advanced visual interfaces*. ACM, 2004, pp. 244–247.

[9] Steim Foundation, "junXion v4 manual," Online, 2008, available: `http://www.steim.org/steim/support/manuals/jX`
`Manual.pdf`. Accessed October 16, 2009.

[10] T. Place and T. Lossius, "Jamoma: A modular standard for structuring patches in max," in *Proceedings of the International Computer Music Conference*, 2006, pp. 143–146.

[11] E. Eagen and L. Haken, *EagenMatrix User Guide*, 2013.

[12] J. W. Hunt and D. M. Mcllroy, "An algorithm for differential file comparison," Bell Laboratories, Tech. Rep., 1976.

[13] P. Dragicevic, S. Huot, and F. Chevalier, "Gliimpse: Animating from markup code to rendered documents and vice versa," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '11. New York, NY, USA: ACM, 2011, pp. 257–262.

[14] C. Bardiot, "Rekall," *Performance Research*, vol. 20, no. 6, pp. 82–86, 2015.

[15] R. A. Fiebrink, D. Trueman, C. Britt, M. Nagai, K. Kaczmarek, M. Early, A. Hege, and P. R. Cook, "Toward understanding human-computer interaction in composing the instrument," in *Proceedings of the International Computer Music Conference*, 2010, pp. 135–142.