# Interactive Sonification Markup Language (ISML) for Efficient Motion-Sound Mappings

James Walker, Michael T. Smith, and Myounghoon Jeon(✉)

Mind Music Machine Lab, Michigan Technological University,
Houghton, MI, USA
{jwwalker,mtsmith,mjeon}@mtu.edu

**Abstract.** Despite rapid growth of research on auditory display and sonification mapping per se, there has been little effort on efficiency or accessibility of the mapping process. In order to expedite variations on sonification research configurations, we have developed the Interactive Sonification Markup Language (ISML). ISML is designed within the context of the Immersive Interactive Sonification Platform (iISoP) at Michigan Technological University. We present an overview of the system, the motivation for developing ISML, and the time savings realized through its development. We then discuss the features of ISML and its accompanying graphical editor, and conclude by summarizing the system's feature development and future plans for its further enhancement. ISML is expected to decrease repetitive development tasks for multiple research studies and to increase accessibility to diverse sonification researchers who do not have programming experience.

**Keywords:** Design research · Interactive sonification · Sonification markup language

## 1 Introduction

The primary aim of this paper is to describe the implementation and applications of a scripting language, Interactive Sonification Markup Language (ISML), designed for expediting the creation of different parameters and specifications for interactive sonification [1] research. Sonification [2] and specifically interactive sonification is multidisciplinary by nature, including computer science, psychology, HCI, acoustics, music, and sound design, etc. We hope to promote greater efficiency in implementing experimental parameters by demonstrating a method for allowing sonification researchers, even those with no programming experience, to efficiently alter the software specifications of an interactive sonification research platform.

To this end, ISML needs to be flexible enough to support a wide variety of experimental setups, yet simple enough to be easily produced and parsed. Thus, once the initial language specification was complete, our work focused on creating a graphical user interface (GUI) that could facilitate the creation of ISML scripts by non-programmers. Given that auditory display and sonification community has grown, this type of effort [e.g., Auditory Menu Library for auditory menu research, [3]] is expected to increase efficiency and accessibility to sonification research.


M. Kurosu (Ed.): Human-Computer Interaction, Part II, HCII 2015, LNCS 9170, pp. 385–394, 2015.
DOI: 10.1007/978-3-319-20916-6_36

The present paper briefly describes our sonification research platform that gave rise to ISML, the motivation for the development of ISML, the effectiveness of ISML in actual research, and the language features and GUI. Finally, limitations of the current system and potential expansions and improvements are discussed. A description of the ISML specification is then given in the appendix.

## 2 Background and Motivation

Interactive sonification research has various applications, such as enhancing learning effects and overall user experience. For example, research [4–6] suggests that a combination of physical interaction with sonified feedback improves users' learning. Other studies [7–9] have shown that interactive sonification can also provide users with a more enjoyable experience and improve accessibility to more diverse audiences.

To conduct active interactive sonification research on those multiple applications, we have developed the Immersive Interactive Sonification Platform (iISoP) [10], which generates sounds based on users' motion – location, movements, and gestures – tracking. We have built the iISoP system using the JFugue Library (i.e., Java API for Music Programming, www.jfugue.org). iISoP leverages the university's Immersive Visualization Studio (IVS), which consists of 24 42″ monitors connected to a computing cluster of 8 machines with high-end graphics cards for display, along with a Vicon tracking system composed of 12 infrared cameras for tracking objects inside the lab. In constructing iISoP, we have developed phased projects – interactive map, virtual instrument, and dance-based sonification as a testbed [see [10], more details].

The third phase of iISoP uses parameter mapping [11] to map data features—in this case, users' physical location, movements, and gestures—into sound. This is accomplished through the following process. Users (dancers, performing artists, kids, and robots) wear special reflective markers and thus, their movements can be tracked by the Vicon system, which relays the users' positional data to the head node on the IVS cluster. There, this information is parsed to generate an abstract visualization of the users' movements on the wall of monitors. The data is simultaneously forwarded to another computer running a program which parses the positional data to dynamically generate sounds or improvise real-time music in response to the users' movements.

However, if the method the audio parser uses for generating sound were fixed, this would limit the usefulness of the platform. Therefore, the motion-to-sound mapping should be configurable, so as to facilitate various kinds of research and experiments. Because many of the researchers (psychologists, sound designers, performing artists, etc.) using iISoP may lack programming experience, expecting them to configure the system by altering the source code by hand would be an unreasonable expectation.

ISML was developed to bridge this gap between the non-technical researchers and the research platform. ISML is a simple scripting language for configuring iISoP's motion-to-sound mapping. To make generating these scripts even easier, ISML includes a web-based GUI so that researchers can create a script simply by responding to prompts, rather than needing to learn ISML's syntax and semantics. This method of configuring the mappings is far more efficient and accessible to non-technical

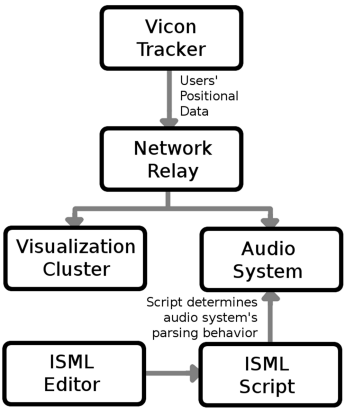


**Fig. 1.** ISML's role in the iISoP system

researchers than attempting to reprogram the system for every new experiment. Figure 1 illustrates ISML's role in the iISoP system.

The format of ISML's syntax is loosely inspired by markup languages, a system for annotating documents so that the annotations are distinguishable from the actual content [12]. Two major markup language standards are Standard Generalized Markup Language (SGML) and Extensible Markup Language (XML), the latter of which is a simplified version of the former designed to maintain its most useful aspects [13, 14]. While markup files are typically "documents" of various kinds, they can also represent arbitrary data structures [15]. ISML files, which can be thought of as an activity flowchart or state machine defining application behavior in response to external inputs, are more similar to the latter. Note that ISML is only inspired by these standards and does not make any attempt to conform to them.

ISML also bears similarity to scripting languages, since it is designed to automate the behavior of iISoP's sound generation application. Specifically, it is an example of an audio synthesis scripting language. Other languages of this type exist. A non-exhaustive list of examples includes the ChucK audio programming language [16], the C-based audio programming language Csound [17], the commercial Reaktor software, and the MPEG-4 Structured Audio standard [18]. Some of these systems use graphical interfaces for their scripting languages, while others are textual. Additionally, some of them support live coding, the ability to change the program's behavior while it is running.

By comparison, ISML scripts can be created in a graphical or textural manner. The present version of ISML does not support live coding. To the authors' knowledge, ISML is the only scripting language specifically designed to translate physical movements into dynamically generated sound, which makes it very different from other audio scripting languages.

## 3 Cost Efficiency: Programmer Time and Effort

Based on our own experience developing the system (two CS/ECE Ph.D. graduates and one CS undergraduate), creating a complete iISoP specification by programming it from scratch takes about 50 h for an experienced programmer. Reusing a preexisting specification in the creation of a new one would likely cut that time to about 20–40 %, or 10–20 h. Conversely, creating an equivalent specification for a research study using ISML takes about 5 h from scratch, or 1–2 h if reusing an existing specification, and has the added advantage of not requiring extensive training in programming. Getting ISML fully operational requires an initial time investment of approximately 70 h. Taking all of this into account, Table 1 compares the total specification development times of each approach, demonstrating that the time savings of using ISML as opposed to creating new specifications for different research studies with hard coding from scratch rapidly accumulate. It seems that the development cost of ISML "pays for itself" after approximately 3–4 specifications.

## 4 System Description

### 4.1 Overview

The iISoP system dynamically changes sound output via the following procedure. First, the system checks to see whether a set of conditions has been met; for example, "The user is currently moving at a speed equal to or greater than 1 meter per second". If these conditions are met, the system executes one or more actions; for example, "Change the key signature to C-major and the time signature to 4/4". Conditions are optional: It is allowed to specify a set of actions that is executed all the time, without any conditions being met.

Each set of conditions and actions is organized into an activity. Activities serve as a mechanism for grouping conditions and actions together. By having multiple activities, it is possible to have different sets of conditions and actions which can be checked and executed. Within an activity, all of the conditions must be satisfied in order for the actions to be executed; outside of that activity, its conditions do not matter.

Thus, considered collectively, activities and their conditions form a disjunction of conjunctions.

**Table 1.** Comparison of spec development times with and without ISML

| Specifications | Hours (code) | Hours (ISML) | Total (code) | Total (ISML) |
|---|---|---|---|---|
| 1 | 50 | 75 | 50 | 75 |
| 2 | 10–20 | 1–2 | 60–70 | 76–77 |
| 3 | 10–20 | 1–2 | 70–90 | 77–79 |
| 4 | 10–20 | 1–2 | 80–110 | 78–81 |
| 5 | 10–20 | 1–2 | 90–130 | 79–83 |

Lastly, activities are organized into *items.* Items provide a scoping mechanism for variables. ISML has 26 variables (a–z) available for use. However, each variable exists independently within the item in which it is used; that is, each variable has item scope. Within each item, all of that item's activities are executed in sequential order from top to bottom. Each item, and all of its activities, are executed (all conditions are checked and the corresponding actions are executed) every cycle of the system. There is no limit on the number of items an ISML script may contain.

The current version of ISML allows for two basic types of conditions: *object* and *comparison.* An *object* condition applies only to an object in the Vicon tracking system that possesses the user-specified name; for example, "left_foot". A *comparison* condition compares whether two values are equal to, greater than, or less than each other. Values that can be compared include constants, variables, current beats per minute, current velocity (in any of the X, Y, or Z axes, or the composite velocity on all three axes), average velocity, acceleration, proximity (the average distance between all tracked objects), current position, and elapsed time (which may be repeatable for conditions that are checked at regular intervals).

Additionally, the current version of ISML allows for the following types of actions:

- Assignment. Sets one value equal to an expression. An expression may be another value, or two other values with some arithmetic operation applied on them (e.g., a = b*c).
- Set the key signature.
- Set the time signature.
- Set the instruments being used.
- Play a specific sequence of notes.

### 4.2 Graphical Editor

Since ISML is designed to be usable by non-programmers, it includes a GUI for the creation of scripts via prompts. The GUI is written entirely in Javascript, so it can be run in a Web browser; and because it does not use any server-side code, it can be run from any computer.

The GUI's appearance is shown in Fig. 2. Upon startup, the user is presented with a blank script and can either access a comprehensive help guide, load an existing script, or begin creating a new script by clicking the "Add Item" button. From here, the GUI dynamically generates additional buttons for adding and deleting the various components of the script under development. To prevent mishaps, selecting a checkbox is required before deleting script components.

To aid user comprehension, each component of the GUI is indented and color-coded so that distinguishing different activities, condition sets, and so on, is more intuitive. Additionally, a "mapping summary" is generated on the side of the screen which summarizes the motion-to-sound mapping that has been created so far. These features help lighten the cognitive burden on the user by keeping the visual representation of the script organized.

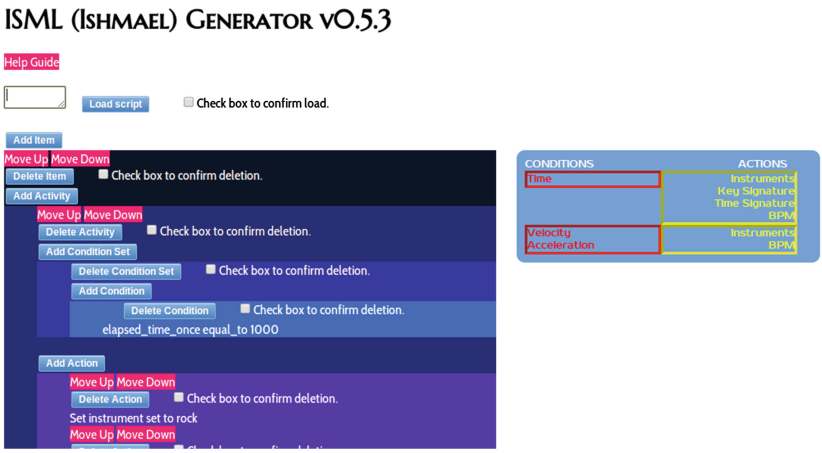


**Fig. 2.** Screen capture of the ISML graphical editor (Color figure online)

Once editing is complete, the user can download the valid ISML-formatted file by clicking the "Download ISML File" button. This file can be further tweaked by hand if desired, or loaded back into the GUI editor at any time for later revision.

### 4.3 Feature Development

The ISML specification and accompanying editor have gone through several small revisions. After initial development, the specification was changed from infix to prefix notation for easier parsing, and semantic error prevention was added to the GUI editor. Various other tweaks were applied to the specification to simplify parsing, and then the "mapping summary window" (Fig. 2, right side) feature was added. Finally, the color scheme was changed to be more visually pleasing. Both the ISML specification and the editor are in a fluid state and will undergo iterative development cycles with end users.

## 5 Discussion and Future Work

Faste and Faste [19] proposed four categories of design research. With ISML, we have seized on two of those categories: "research on design" by investigating our research process for possible enhancements; and "design of research" by planning and preparing for future research on the system. From our experience, the ability to rapidly generate new sonification configurations has already shown benefits, and the time savings will continue to accumulate as more research is conducted with the system. This approach will decrease repetitive development tasks for various experiments and increase accessibility to researchers from various disciplines, who do not necessarily have programming experience.

In the future, we intend to strengthen our claims of ISML's effectiveness with an empirical study and to continue developing ISML to improve its efficacy; for example, by adding features for volume control, panning, and audio filters. Additionally, the

current version of ISML only supports the configuration of audio. In the future, we plan to add the ability to configure the visualization parameters and emotional parameters through ISML as well.

## Appendix: ISML Language Format

ISML uses a format that encloses sections between tags, with the closing tag indicated by a slash (e.g., <tag> and </tag>). ISML supports the following tags:

```
<item></item>
<activity></activity>
<if><or><then></if>
```

Everything within a single item should be contained between <item> and </item> tags. The execution order of items is not defined; thus, items should be written to be independent of one another. Everything within an activity is contained between corresponding <activity> and </activity> tags. All activities within an item are checked and executed in sequential order. The content of each activity is in the following form:

```
<if>
condition_a1
condition_a2
...
<or>
condition_b1
condition_b2
<or>
...
<then>
action_1
action_2
...
</if>
```

Or alternatively, simply:

```
action_1
action_2
...
```

A condition set is defined as all the conditions contained between two tags. Any of the following forms are possible:

```
<if>
conditions
<or>

<or>
conditions
<or>

<if>
conditions
<then>

<or>
conditions
<then>
```

Conditions may have either of the following 2 forms:

```
object name_of_object
comparator value_1 value_2
```

Note that, for ease of parsing, ISML uses prefix rather than infix notation.
Valid comparator tokens include:

```
equal_to not_equal_to greater_than less_than
```

Values may be any valid number, a variable name (a-z), or any token from the following list:

```
cur_velocity_x cur_velocity_y cur_velocity_z cur_velocity_composite
avg_velocity_x avg_velocity_y avg_velocity_z avg_velocity_composite
acceleration_x acceleration_y acceleration_z acceleration_composite
avg_proximity x_position y_position z_position elapsed_time_once
elapsed_time_repeatable bpm
```

Actions may have any of the following forms:

```
assignment
set key signature
set time signature
set instruments
play notes
```

Set key signature is denoted by any token from the following list:

```
c_major g_major d_major a_major e_major b_major fsharp_major csharp_major
a_minor e_minor b_minor fsharp_minor csharp_minor gsharp_minor dsharp_minor
asharp_minor
```

Set time signature is denoted by any token from the following list:

```
1/2 2/2 3/2 4/2 2/4 3/4 4/4 2/8 3/8 4/8 6/8 8/8
```

Set instruments is denoted by any token from the following list, which is slated for future expansion:

```
electronica rock orchestral
```

Assignments take the following form:

```
= value_1 operator value_2 value_3
```

The third value is optional. In the case of simply transferring one value to another, assign is used as the operator; e.g.,

```
= a assign cur_velocity_x
```

Values 2 and 3 may be any valid number, a variable name (a-z), or any token from the token list for conditions (cur_velocity_x and so on). Value 1 may only be a variable or bpm because it does not make sense to assign to other kinds of properties, such as velocity, which are read-only.

Valid operators include:

```
+ - * / % ^ abs
```

"Play notes" has the following form:

```
play_notes:note_list
```

## References


1. Hermann, T., Hunt, A.: The discipline of interactive sonification. In: Proceedings of the International Workshop on Interactive Sonification (ISon), Bielefeld, Germany (2004)
2. Kramer, G., et al.: The sonification report: status of the field and research agenda. Report prepared for the National Science Foundation by Members of the International Community for Auditory Display (ICAD), Santa Fe, NM, USA (1999)
3. Raman, P., Davison, B.K., Jeon, M., Walker, B.N.: Reducing repetitive development tasks in auditory menu displays with the auditory menu library. In: Proceedings of the International Conference on Auditory Display (ICAD), pp. 229–237, Washington, D.C., USA (2010)
4. Antle, A.N., Droumeva, M., Corness, G.: Playing with the sound maker: do embodied metaphors help children learn? In: Proceedings of the International Conference on Interaction Design and Children, pp. 178–185, Chicago, IL, USA (2008)
5. Ferguson, S.: Learning musical instrument skills through interactive sonification. In: Proceedings of the International Conference on New Interfaces for Musical Expression (NIME2015), pp. 384–389 (2006)
6. Howison, M., Trninic, D., Reinholz, D., Abrahamson, D.: The mathematical imagery trainer: from embodied interaction to conceptual learning. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI2011), pp. 1989–1998, Vancouver, BC, Canada (2011)
7. Walker, B.N., Godfrey, M.T., Orlosky, J.E., Bruce, C.M., Sanford, J.: Aquarium sonification: Soundscapes for accessible dynamic informal learning environments. In: Proceedings of the International Conference on Auditory Display (ICAD 2006), pp. 238–241, London, UK (2006)
8. Jeon, M., Winton, R.J., Henry, A.G., Oh, S., Bruce, C.M., Walker, B.N.: Designing interactive sonification for live aquarium exhibits. In: Stephanidis, C. (ed.) HCII 2013, Part I. CCIS, vol. 373, pp. 332–336. Springer, Heidelberg (2013)
9. Jeon, M., Winton, R.J., Yim, J.B., Bruce, C.M., Walker, B.N.: Aquarium fugue: interactive sonification for children and visually impaired audience in informal learning environments. In: Proceedings of the 18th International Conference on Auditory Display (ICAD 2012), Atlanta, GA, USA (2012)
10. Jeon, M., Smith, M.T., Walker, J.W., Kuhl, S.A.: Constructing the immersive interactive sonification platform (iISoP). In: Streitz, N., Markopoulos, P. (eds.) DAPI 2014. LNCS, vol. 8530, pp. 337–348. Springer, Heidelberg (2014)
11. Hermann, T.: Taxonomy and definitions for sonification and auditory display. In: Proceedings of the 14th International Conference on Auditory Display (ICAD 2008), Paris, France (2008)
12. Coombs, J.H., Renear, A.H., DeRose, S.J.: Markup systems and the future of scholarly text processing. Commun. ACM **30**(11), 933–947 (1987)

13. World Wide Web Consortium: Extensible Markup Language (XML) 1.0 (5th edn.) 2008. Web. 4 Oct 2014
14. Clark, J.: Comparison of SGML and XML. World Wide Web Consortium, 1997. Web. 4 Oct 2014
15. Philip, F.: Extremes of XML. XML London, 15–16 June 2013
16. Wang, G.: The ChucK Audio Programming Language. Dissertation, Princeton University, Princeton (2008)
17. Boulanger, R.: The Csound Book. MIT Press, Cambridge (2000)
18. Scheirer, E.D.: The MPEG-4 structured audio standard. In: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal processing 1998, vol. 6. IEEE (1998)
19. Faste, T., Faste, H.: Demystifying 'design research': design is not research, research is design. IDSA (2012)