AN OBJECT ORIENTED MODEL FOR THE REPRESENTATION OF TEMPORAL DATA IN THE INTEGRA FRAMEWORK

Jamie Bullock
Birmingham Conservatoire
Birmingham City University
james.bullock@bcu.ac.uk

Henrik Frisk, Ph.D.
Malmö Academy of Music
Lund University
henrik.frisk@mhm.lu.se

ABSTRACT

In this paper we describe a model for the representation and storage of time-related data in the context of the Integra framework. We highlight the need for a portable, sustainable data format that can be shared between common environments for audio and multimedia processing. Since the storage of spectral and gestural data has been covered by the SDIF and GDIF formats, we focus on the storage of multimedia processing module state, and changes in state over time. After a review of existing research in this area we propose an object-oriented approach to both the storage format and the runtime-handling of module state in keeping with the Integra design paradigms. We also show how an XML-based format can lead to a semantically-rich, flexible and robust approach to storage of module state and interpolated or non-interpolated state sequences.

1. INTRODUCTION

The primary motivation for this research is to attempt at complementing the concept of the Integra module [4, 3]; a self contained and largely synchronous sound processing module, with the ability to store and edit time based, asynchronous data. The address space design and storage format specification of Integra modules makes way for a generalized and interchangeable time format in which sequences of events are dispatched to module instances. As a result of the generic and abstract nature of the Intgera framework and the abstraction layer it contains, any time-related data stored according to the model proposed below can be shared by any module instance that has a compatible interface.

1.1. Related work

The dominant mechanism for the representation and storage of time-based control data in a musical context is the MIDI protocol and its related formats. The file format we propose draws upon the strengths of MIDI whilst providing an alternative that is more suited to the complex demands of interactive and generative music. The facilities offered by the format are discussed in sections 2 to 3. We will first proceed with a review of existing developments in the field that are related to our current work.

1.1.1. SDIF

SDIF is an extensible standard for the storage and interchange of 'sound description' information including, but not limited to spectral data and 'higher-level' audio features[8, 6]. SDIF is a portable binary file format that is currently supported in a number of widely-used programming environments including Max/MSP, OpenMusic and CLAM. The GDIF format extends SDIF by adding the capability to store gesture-related data including controller data, sensor data and meta-data about the gesture capturing session [7]. One of the limitations of SDIF as a *general purpose* format for time-related data is that it trades semantic richness for efficiency of representation. This is a sensible design choice given that the original purpose of SDIF was to standardise the storage of spectral data frames. We have decided in Integra to use SDIF and GDIF where appropriate, supplementing these standards with our own XML-based format, described in section 3.

1.1.2. MetriXML

MetriXML is a format developed by Xavier Amatriain for the original CLAM software, and can be thought of as a Music-N style Generator/Score language implemented in XML[1]. A MetriXML score is an XML file containing a header tag, which is used store meta data about the score, and a body tag, which contains a series of event tags representing time stamped messages destined for Instruments declared in the header. An example MetriXML event is shown below.

```
<Event>
    <Time type="temporal">
        <Seconds>01</Seconds>
    </Time>
    <Receiver>clarinet</Receiver>
    <Message>
        <Parameter>Gain</Parameter>
        <Value>0,1 1,0.5</Value>
```

```
    </Message>
</Event>
```

The Integra IXD sequence format is similar to MetriXML in both syntax and semantics, particularly in its notion of 'events'. MetriXML also incorporates the SDIF format by reference using the $< SDIFDefinitionFile >$ tag.

1.1.3. SMIL

The Synchronized Multimedia Integration Language, or SMIL [5], is a general purpose multi media format. Supported by W3C it is an XML-based language with which authors may write interactive multimedia presentations among other things. It had some support in InternetExplorer 5.5 and 6, Apple's QuickTime player, RealAudio and Helix players, as well as the reference Ambulant player. SMIL has sophisticated methods for synchronizing multiple multi media streams and it was an inspiration in the early stages of the development of the Integra time format. For example, the *seq* and *par* element in SMIL loosely correspond to the *sequence* and *preset* element in the IXD format. With the modularization of SMIL 3.0 it is feasible to imagine incorporating SMIL elements in IXD files.

An important difference between SMIL and the proposed format is that SMIL is a *language* that may be used to construct content whereas the primary purpose of the Integra format is to store time based information created elsewhere.

2. THE PLAYER MODULE

The Integra framework seeks to address the issue of interchange between environments by defining a set of protocols and file formats for portable development, along with a shared library, which implements these protocols and provides transparent IXD file parsing (see section 3). Integra modules consist of three components: an interface definition; an implementation and instance data representing the module's state at 'save time'. The Player module is a special module in the sense that it is responsible for instigating synchronous state changes in other modules. Time and timing in the Integra framework are almost always handled asynchronously. The Player module is different, in that it is expected to implement an internal clock, which can be used to schedule future events. These events can be sequenced in time as Sequences, or bundled up into Presets, which enable multiple attributes of a given module to be changed simultaneously. The Player interface supports the following features:

- looped and reverse-looped playback of sequenced data
- random access to sequence data
- non-linear sequences:
  - sequences that contain other sequences
  - sequences that control the playback of other sequences
- relative representation of time (non-absolute)
- non-track based: each time location can have an arbitrary number of 'events' with an arbitrary number of 'targets'

2.1. The Event interface

'events' can be thought of as class instances that conform to the Event interface. Event inherits from the Integra base class, adding the attributes for the event *address*, *value* and *interpolation* setting.

'events' are useful for containing data for a single attribute, or small sets of uncorrelated attributes, e.g. recorded sensor data, BPF data, note data. However, often we will want to store a snapshot of all of the attributes of a module at a given point in time. For this we have the Preset class. Preset inherits from the Event class, and can be thought of as a special type of event: an event that contains other events in a set where each event is guaranteed to have a different address. Presets have certain implicit rules associated with them:

- They contain no time information (all events in a preset can be said to coexist simultaneously)
- All event address strings must be unique within a given preset.

Since the Preset class definition conforms to the Event interface, Preset and Event instances can be used interchangeably in contexts such as sequences.

3. STORAGE: THE IXD FILE FORMAT

The IXD file format is an XML-based format that is already used within the Integra framework for storing Module definitions and Collections of modules[3]. The time-based data format described in this paper will form an extension to the existing IXD formats and the various Schemas can be reused extensively.

3.1. Sequences

For maximum flexibility and re-usability both sequences and presets are stored as separate entities and linked to in the collection using standard XLink syntax [2]. Should the included file contain multiple sequences or presets an additional selector argument may be supplied. In the example below, the sequence with id 2 from the file mysequence.ixd will be embedded.

```
<state>
  <value title="mysequence"
         xlinktype="simple"
         href="mysequence.ixd"
         show="embed"
         selector="2" />
</state>
```

Sequence files contain list of events according to the following trivial example which sets delay.time to 400 at tick 0, starts another Player, and changes delay.time to 800 at tick 100:

```
<sequence id=0>
  <event tick="0" id="1"
         marker="Foo Bar">
    <address class="delay"
             attribute="time"/>
    <value>400</value>
  </event>
  <!-- we deleted event id="2"
       at some point -->
  <event tick="0" id="3"
         marker="">
    <!-- this goes to 'another'
         player -->
    <address class="player"
             attribute="play"/>
    <value>1</value>
  </event>
  <event tick="100" id="4"
         marker="Baz Bam">
    <address class="delay"
             attribute="time"/>
    <value>800</value>
  </event>
</sequence>
```

It is important to note that the implication of sending the value '1' to another Player is that events can trigger sequences, which can in turn contain further events. This provides an implicit arbitrary nesting of sequences and events through the mechanism of Player instances 'playing' other Player instances.

3.2. Preset

If an event is of type 'preset' it contains a link (again using standard XLink syntax [2]) to a preset file which may contain one or many presets. Selection of individual presets may be achieved using the same selector attribute used in the sequence link.

```
<event tick="100"
       type="preset"
       href="mypreset.ixd"
       id="5"
       marker="Load delay preset 1"
       selector="2">
```

A stored preset defines events with address/value pairs. These events have no time information associated with them but are merely 'snapshots' and there must be no duplicate addresses in any given preset. Below is a simple example:

```
<preset class="delay" name="delay preset 1">
  <event>
    <address attribute="frequency"/>
    <value>800</value>
  </event>
  <event>
    <address attribute="phase"/>
    <value>0.5</value>
  </event>
</preset>
```

3.3. Tagging and meta-data

Because our temporal model uses the XML-based Integra IXD format for data storage it inherits the ontological possibilities provided by our existing schema. This means that sequences and presets can be given 'tags' containing semantic information, and that 'relations' can be created between different sequences, or for example presets and module instances. We also provide the possibility to embed sequence descriptions, and links to documentation about a given sequence or preset. This could have applications in the musicology of live electronic music for example where multiple versions of performances could be encoded and given additional semantic markup to inform future study.

3.4. Evaluation

Whilst the IXD format alluded to in this paper has a formally defined schema, and is currently used for a range of example module definitions provided with the Integra framework, the temporal model and storage constructs have not yet been implemented and tested in practice. We therefore defer evaluation of the proposed model to future research. However, it is worth noting at this point the similarity of aspects of IXD with the RDF and OWL specifications. RDF is a W3C recommendation that defines a language for describing resources using subject-predicate-object expressions[9]. Whilst RDF is primarily useful for encoding objects and object properties, OWL places a greater emphasis on relationships *between* objects to create descriptive ontologies. This capability is provided by Integra IXD through the 'relation' construct and although OWL is currently more exhaustive in the types of relationship it can define, there is no reason why the Integra Relation class could not be extended in future versions of the schema. In general we decided to develop our own XML-based format rather than use RDF or OWL because on the one hand with specific requirements that can't be met by these standards and on the other, Integra IXD provides enough semantic richness and potential

for ontological description along with extensibility to make OWL unnecessary.

4. CONCLUSION

In this paper, we have outlined a proposal for an extension to the Integra framework that allows the representation and storage of temporal data. We have acknowledged the significance of existing formats, such as SDIF and GDIF for the storage of high-rate data such as spectral analysis frames and gesture capture data. However, we have highlighted a need for an environment neutral format for the storage of multimedia processing module state, and state change over time, even though such format may well reference SDIF, GDIF and even SMIL formats. Hence we have proposed an XML-based extension to the existing Integra file formats, which addresses some of these problems, and opens new possibilities for file-based transformation of temporal data for multimedia processing modules. Finally, we argue that using an XML-based format provides a level of semantic richness not possible with binary, and simpler text-based formats.

5. REFERENCES

[1] X. Amatriain, "An object-oriented metamodel for digital signal processing with a focus on audio and music," Ph.D. dissertation, Departament de Tecnologica, Universitat Pompeu Fabra, Barcelona, Spain, 2004.

[2] Arbouzov, L et al., "Xml linking language (xlink) version 1.1," Web resource. [Online]. Available: http://www.w3.org/TR/2006/CR-xlink11-20060328/

[3] J. Bullock, H. Frisk, and L. Coccioli, "Sustainability of 'live electronic' music in the integra project," in *The 14th IEEE Mediterranean Electrotechnical Conference Proceedings*. Ajaccio, Corsica: IEEE, 2008, iEEE Catalog Number: CFP08MEL-CDR.

[4] J. Bullock and H. Frisk, "libIntegra: A System for Software-Independent Multimedia Module Description and Storage," in *Proceedings of the International Computer Music Conference 2007*. Copenhagen, Denmark: ICMA, 2007.

[5] D. e. a. Bulterman, "Synchronized multimedia integration language (smil 3.0)," Web resource. [Online]. Available: http://www.w3.org/TR/2008/REC-SMIL3-20081201/

[6] M. W. A. Chaudhary, "Audio applications of the sound description interchange format standard," in *In Proceedings of the International Computer Music Conference, Ann Arbor, Michigan*, 1999, pp. 276–279.

[7] A. R. Jensenius, T. Kvifte, and R. I. Godøy, "Towards a gesture description interchange format," in *NIME '06: Proceedings of the 2006 conference on New interfaces for musical expression*. Paris, France, France: IRCAM — Centre Pompidou, 2006, pp. 176–179.

[8] D. Schwarz, I. Centre, G. Pompidou, and M. Wright, "Extensions and applications of the sdif sound description interchange format," in *In Proceedings of the International Computer Music Conference*, 2000, pp. 481–484.

[9] S. Staab and R. Studer, Eds., *Handbook on Ontologies*, ser. International Handbooks on Information Systems. Springer, 2004.