SignalML: metaformat for description of biomedical time series

Piotr J. Durka *, Dobiesław Ircha

Laboratory of Medical Physics, Institute of Experimental Physics, Warsaw University, ul. Hoża 69, 00-681 Warszawa, Poland, http://brain.fuw.edu.pl.

Abstract

This paper introduces a complete and elegant solution to the problem of inherent incompatibility of different formats used for digital storage of biomedical time series (in particular EEG) and their annotations.

We define a simple XML-based language, in which information on the structure of binary data files can be simply and efficiently coded. In most cases, the description of an existing format takes relatively few lines of XML code. Once written, this information can be used by any software, which, owing to this meta-description, may read the *original data files*, thus eliminating the need for conversions and duplication of data.

This proposition is hereby submitted to an open discussion within the community involved in relevant research, clinical and commercial applications. Links to the current version of the XML Schema defining the language and the mailing list `signalml-l@fuw.edu.pl` dedicated to this topic are located at `http://eeg.pl/SignalML/`. This site offers also the first implementation: an Open Source multiplatform software for display/annotation of the biomedical time series, which, owing to the idea presented in this paper, can be easily adopted to new data formats.

Key words: biomedical time series, compatibility, EEG, digital storage, annotations, tags, European Data Format, PhysioNet

* *Corresponding author*: Piotr J. Durka, Laboratory of Medical Physics, Institute of Experimental Physics, Warsaw University, ul. Hoża 69, 00-681 Warszawa, Poland. tel. (48 22) 5532126, fax (48 22) 6226154

Email addresses: `durka@fuw.edu.pl` (Piotr J. Durka), `rircha@fuw.edu.pl` (Dobiesław Ircha).

URL: `http://durka.info` (Piotr J. Durka).

1 Introduction

Incompatibility of formats used for digital storage of biomedical time series (and related annotations) is one of the major problems of contemporary electroencephalography (c.f. [1,2]). It almost limits the exchange of data and processing algorithms to clinics and laboratories using equipment and software from the same manufacturer, which is one of the causes of the lack of a coherent progress in clinical encephalography in the last decades [3]. The situation looks similar e.g. for the digital EMG (electromyographic) data; Jabre and Salzsieder [4] write:

> "Different electromyography (EMG) machines store their data in different formats that vary from manufacturer to manufacturer and even between different EMG machines from the same manufacturer. As advanced as these machines are today, it is necessary in most cases to use faxes, scanners, or a common interface such as Adobe's file format to exchange data between them."

The European Data Format was proposed in 1991 as a "simple format for exchange of digitized polygraphic recordings" [5]. Aimed primarily at polysomnograms consisting of signals stored with different sampling frequencies, it was optimized for their efficient storage by introducing the extra notion of a "record", containing sequences of equal time length from different signals. In 2002 its extension to the EDF+ was proposed in a way preserving compatibility with previous versions, extending possibilities of annotating signals and storing interrupted recordings. Authors of this standard write: "studying the specification and then programming an EDF import/export unit typically takes a few days" (`http://www.hsr.nl/edf/`). However, not every neuroscientist is also a programmer, and up till now we still lack of a freely available and user friendly software for browsing and annotating EDF files.

A different set of formats was implemented in the largest Web-based research resource for complex physiological signals `PhysioNet.org` [6]. This site offers Open Source programs for display, conversions, annotation and processing of data in these formats.

One could continue enumerating existing solutions (a review is given by [1]); the standard thinking about these problems leads to a longing for a "universal standard". Even supposing that all the interested parties (neurophysiologists, clinicians, engineers and companies) would *agree* (!) on any particular solution as a standard, it would work only for the newly created recordings and systems. All the existing data would have to be *converted* to this new/chosen format. But conversions between different data formats in certain cases may lead to a loss of information or accuracy. And because the existing software systems

are designed to work only with the "old" formats, original files would have to be preserved and this conversion would actually mean *duplicating* the data.

The deeper meaning of the word *duplicating* is the most important—trivial, but to date unexplored—observation. In many cases conversions mean just changing the format of the header or footer, while the numbers (actual values of the time series) remain the same: usually 16-bit integers, sometimes 32-bit floating points. It would be irrational to invent new formats for this. So if we would just know how to interpret the different headers [1], i.e. read from them at least the crucial signal's parameters like the number of channels and the sampling frequency, we could use the *original* datasets**!** All it takes is an efficient and universal metaformat for description of their structures.

Transition from paper to digital recordings created also the issue of annotations (tags). To make them as flexible as the pencil marks on paper EEG, digital annotations need to be:

(1) Easy to place at an arbitrary point or epoch of any given channel(s), and
(2) Uniquely linked to the datafile to which they refer, and even more: also to the way the signal was presented to the algorithm or expert who created the annotations (montage).

1.1 The Solution

We propose a general solution in terms of a universal and simple XML-based language (SignalML) for meta-description of existing formats. Using this metainformation, we may adopt any software to work with the *original datasets*, just as easily as e.g. programming a "classical" import unit for a new format. This eliminates the need for conversions and multiplying the data.

Definition of SignalML includes a universal and flexible schema for defining and creating arbitrary annotations (tags), which can be uniquely (secured by checksums) related to any arbitrary point or epoch of any channel(s) of the recording. These definitions can include e.g. the traditional hypnograms, artifact tags, arbitrary textual annotations related to any instant of the recording or exact markings of start, end and channel of transients.

[1] Or footers, or auxiliary files containing the necessary information

2 SignalML

The XML Schema formally defining the language is `http://eeg.pl/SignalML.xsd`. The web page `http://eeg.pl/SignalML/` contains wider description, examples, compliant software and a link to the mailing list devoted to discussion on this proposition. Below we briefly present its main features.

2.1 Files

"Everything is a file", so let's start explaining the idea by a clear designation of the involved types of files:

(1) **definition of the XML Schema**, formally defining the syntax of the language, is contained in the file `http://eeg.pl/SignalML.xsd`.
(2) **metainformation files** should be unique for each existing format (or its version), so they should be possibly identified by a URL, e.g. for the European Data Format it is `http://eeg.pl/meta_EDF.xml` (Table 1)
(3) **annotation files** are related uniquely to a given data file, and contain:
- complete information necessary for a proper interpretation of the data in a given file. Reference to a proper metainformation file may be sufficient, or, alternatively, explicit values of parameters (sampling, number_of_channels etc.) defined in section 2.2 can be included,
- filename and control sum identifying uniquely the data file to which the annotations relate,
- montage (transform) applied to the signal when creating annotations,
- definitions of tags/annotations,
- tags/annotations based upon the above definitions.

As a special case, annotation files with a complete set of parameters describing a "raw" (containing only data samples) datafile may constitute an autonomous format—this particular case would partially correspond to the translation proposed in [4]. Content of an example annotation file is given in Table 2.

2.2 Parameters and properties

We define a minimum set of parameters, common to all the digitally stored (multichannel) physiological signals:

data_format with following attributes:
- *frame_type* defines how the data from different channels (different signals) are mixed in the file. Current options include standard full multiplexing

(*'multiplex'*), *'edf_frame'* (arbitrary) and *'frames'* as in PhysioNet.org,
- *sample_type*—the format of samples within the file, e.g 16-bit integer, 32-bit IEEE floating point etc.
- *offset*—the number of bytes to skip (header size) before reading data samples

number_of_channels denotes the number of signals stored in file

sampling_frequency has an attribute `units` (e.g. 'Hz') and can be scalar or vector[2]. In the former case a uniform sampling frequency is assumed for all the channels (signals) in file

calibration_gain defines the number by which sample values must be multiplied to obtain values in *units* (e.g. *'microVolts'*, attribute of *'calibration_gain'*)

calibration_offset numerical value to be substracted from each sample value before multiplying by *calibration_gain*. If not defined explicitly, assumed zero

channel_names is a vector of the names for signals stored in the file—usually names of electrodes from which signals were recorded.

These values constitute the minimum information, necessary to display the file's data. If *frame_type* is diffrent from *multiplex*, the minimum information contains some more parameters.

In some idealized case of an 'example' dataformat, all these parameters could be read from the file's header by simply supplying offset in bytes[3] and the type of the variable to be read, e.g.

```
<number_of_channels type='byte' offset='16'/>,
```

and the whole metainformation would contain just a few of such clearly readable lines. However,

(1) apart from these generally understood parameters, datafiles (in particular headers/footers etc.) may contain various kinds of information and it's difficult to propose a general naming/ontology for all these cases,
(2) these parameters are sometimes coded in an indirect way, e.g. in the EDF format `sampling_frequency` is derived from the duration of a data record and the corresponding number of samples.

Therefore, we introduce a `<property>` tag, where the name, defined freely within the metainformation file, is coded in the `id` attribute. E.g. a string identifying the patient in an EDF file (Table 1) can be specified as

[2] If the attribute `index` is absent in the defining tag, we assume scalar.
[3] Some parameters can be stored in different files than the data. Unless explicity identified by the `location` attribute, offset values relate to the datafile and are counted from its beginning.

```
<property id='patient_ident' type='ascii' width='80' offset='8'/>
```

These names (ids), contained within the {} parenthesis, can be used as variables in expressions evaluating parameters values. Such expressions, which contain mentioned variables, numbers and signs of the four basic mathematical operations (+-*/), are given in the attribute eval:

```
<sampling_frequency evaltype='float'
                    eval='{nr_of_samples}/{duration_of_data_record}'/>
```

Finally, some of the parameters will be vectors rather than scalars. This property is coded by the property index—if this property is absent in the definition, we assume a scalar. {index} as a number can be used explicitly in the expressions evaluating the vector's values, as in the above example from Table 1 defining the location of the names of electrodes:

```
<channel_names type='ascii' width='16'
               index='1..{number_of_channels}' offset='256+16*{index}'/>
```

If this vectorial notation and four basic operations are not enough to specify decoding of some complicated format, low level routines (currently in Java) can be contained within <code> tags.

Table 1 exemplifies of some of the above constructs. Other examples are available from http://eeg.pl/SignalML/.

3 Examples and implementation

3.1 Format definition

In this section we present an example of a format's definition: a file http://eeg.pl/meta_EDF.xml, containing the metainformation needed and sufficient to decode data stored in the European Data Format (EDF). We hope that this notation, together with the brief description contained in this paper, will be self-explanatory enough to give a general idea of the proposed language. Additionally, metainformation for the EDF format (Table 1) can be compared to the definition of the EDF header available from http://www.hsr.nl/edf. A more extensive definition of the SignalML language (including the Schema) and further examples are available from http://eeg.pl/SignalML.

To properly interpret data stored in the European Data Format (Table 1), we have to read several parameters separately for each signal stored in the

file. It requires an extensive use of the `index='1..{number_of_channels}'` attributes, indicating that the given property or parameter is a vector rather than a scalar.

3.2 Annotations

An imaginary file exemplifying some tag definitions and actual marks (annotations) is presented in Table 2. Section `<datafile_identification>` contains enough information to uniquely link the annotations to a given data file. Section `<signal_transform>` records the possible transforms applied to the signal when the annotations were created, that is, how the signal was presented to an expert or algorithm. It may contain sections `<montage>` and `<filters>`. Section `<tag_definitions>` contains in this example definitions of diverse groups of tags: sleep stages are marked in blocks of fixed length (pages) and quantized starting points (at a page boundary). The definition of "transients in C3" allows for arbitrary starting points and durations of tags, but contains restriction to the channel number 10. Finally, tags of type `'events'` allow to mark any epoch in any channel(s). In routine applications like e.g. sleep stages scoring or marking artifacts, standard content of this section can be automatically copied into this file.

Section `<tag_data>` contains the actual marks, in this example limited to just a few. We observe that each of these tags can be assigned an arbitrary length textual `<annotation>`.

3.3 Implementation

Together with the defining Schema, we provide an Open Source multiplatform viewer, capable of displaying multichannel time series based upon the above discussed metainformation. As a minimum, it allows to effectively share the datasets and annotations between otherwise incompatible laboratories. The system is based upon the General Public License (`http://www.gnu.org/licenses/gpl.html`), with an exception allowing for a restricted linking of a proprietary code, like e.g. signal processing plugins compliant with the program's API.

4 Conclusion

SignalML provides a simple and effective way of encoding the metainformation needed for a proper interpretation of digital time series, stored in different for-

mats. Unlike the actual software (programs), created in thousands for conversions, display or analysis of data in particular dataformats, SignalML encoding requires only one instance (metainformation SignalML file) for a given format to make it readable by any compliant software. E.g. if a hardware producer wants to make the data, stored by his signal acquisition systems, accesible by a (future) wealth of software of other providers (including Open Source), he does not need to supply low-level input/output/conversion routines for any imaginable programming language and operating system. All it takes are up to a few dozen lines of standard XML.

This approach greatly simplifies also the task of writing software capable of accessing data in more than one format. Classically, programmers needed to write separate low-level routines for each different format or format feature. After a software project was closed, it was extremely difficult to add support for another dataformat. With SignalML we can write just *one* routine for reading *any* dataformat based upon its meta-description in SignalML.

Finally, for any interested scientist or clinician, writing such a meta-description for most of the formats is simpler than programming low level I/O routines, and opens access to the wealth of compliant software. The availability of the multiplatform browser/annotator, mentioned in section 3.3, should be seens as the first step in creation of this "wealth", since such an open and user-friendly program was, at least in the field of EEG, needed and missing for years. But above all we hope that this approach will be accepted by the community. This would expectably lead to development and/or adaptation of wide variety of compliant software. Adopting any existing software to use SignalML is a task comparable to writing just one more I/O routine, since standard XML parsers are readily available for most of the programming languages.

The presented idea was exemplified on the electroencephalographic and polysomnographic recordings, but it is not limited to these. Most of the formats for biomedical time series can be efficiently described within the proposed Schema—maybe after some enhancements resulting from the open discussion. This paper, the definition of the SignalML language and the accompanying software implementation provide the critical mass needed to start a discussion and collaborative effort of interested parties, leading to the development of SignalML and compliant software. We hope it will lead to a universal, elegant and widely accepted markup language, which in due time will be submitted as a standard to the World Wide Web Consortium.

5 Acknowledgements

This work was partially supported by the grant of Committee for Scientific Research (Poland) to the Institute of Experimental Physics, Warsaw University.

References

[1] A. Värri, B. Kemp, T. Penzel, A. Schlögl, Standards for biomedical signal databases, IEEE Eng Med Biol Mag (2001) 33–37.

[2] J. Olivan, Formats in clinical neurophysiology: The point of view of a user, http://neurotraces.com/views/formats.html (2003).

[3] P. Durka, K. Blinowska, A unified time-frequency parametrization of EEG, IEEE Eng Med Biol Mag 20 (5) (2001) 47–53.

[4] J. F. Jabre, B. T. Salzsieder, Defining extensible markup language standards for electromyography data transmission across the world-wide web, Muscle & Nerve 999 (2002) S72–S76.

[5] B. Kemp, A. Värri, A. Rosa, K. Nielsen, J. Gade, A simple format for exchange of digitized polygraphic recordings, Electroencephalogr Clin Neurophysiol 82 (1992) 391–393.

[6] G. Moody, R. Mark, A. Goldberger, Physionet: A web-based resource for the study of physiologic signals, IEEE Eng Med Biol Mag (2001) 70–75.

```
<?xml version="1.0"?>
<meta_format>
<header>
 <format id='EDF'/>
 <text_info>EDF data format</text_info>
 <url> http://www.hsr.nl/edf</url>
</header>

<data_format frame_type='edf_frame'
             offset='{header_size}'
             record_size='{duration_of_data_record}'
             sample_size='{nr_of_samples}'
             sample_type='int16'
             />
<parameters>
  <property id='version' type='ascii' width='8' offset='0'/>
  <property id='patient_ident' type='ascii' width='80' offset='8'/>
  <property id='record_ident' type='ascii' width='80' offset='88'/>
  <property id='start_date' type='ascii' width='8' offset='168'/>
  <property id='start_time' type='ascii' width='8' offset='176'/>
  <property id='header_size' type='ascii' width='8' offset='184' evaltype='int32'/>
  <property id='reserved' type='ascii' width='44' offset='192'/>
  <property id='number_of_data_records' type='ascii' width='8' offset='236' evaltype='int32'/>
  <property id='duration_of_data_record' type='ascii' width='8' offset='244' evaltype='float'/>
  <property id='transducer_type' type='ascii' width='80' index='1..{number_of_channels}'
          offset='256+16*{number_of_channels}+80*({index}-1)'/>
  <property id='physical dimension' type='ascii' width='8' index='1..{number_of_channels}'
          offset='256+96*{number_of_channels}+8*({index}-1)'/>
  <property id='physical_minimum' type='ascii' width='8' index='1..{number_of_channels}'
          offset='256+104*{number_of_channels}+8*({index}-1)'/>
  <property id='physical_maximum' type='ascii' width='8' index='1..{number_of_channels}'
          offset='256+112*{number_of_channels}+8*({index}-1)'/>
  <property id='digital_minimum' type='ascii' width='8' index='1..{number_of_channels}'
          offset='256+120*{number_of_channels}+8*({index}-1)'/>
  <property id='digital_maximum' type='ascii' width='8' index='1..{number_of_channels}'
          offset='256+128*{number_of_channels}+8*({index}-1)'/>
  <property id='prefiltering' type='ascii' width='80' index='1..{number_of_channels}'
          offset='256+136*{number_of_channels}+80*({index}-1)'/>
  <property id='nr_of_samples' type='ascii' width='8' index='1..{number_of_channels}'
          offset='256+216*{number_of_channels}+8*({index}-1)'/>
  <property id='reserved2' type='ascii' width='32' index='1..{number_of_channels}'
          offset='256+224*{number_of_channels}+32*({index}-1)'/>

  <sampling_frequency index='1..{number_of_channels}'
          evaltype='float' eval='{nr_of_samples}[{index}-1]/{duration_of_data_record}'
          units='Hz'/>
  <number_of_channels type='ascii' width='4' offset='252' evaltype='int32'/>
  <calibration_gain evaltype='float' index='1..{number_of_channels}'
          eval='1.0F/({physical_maximum}[{index}]-{physical_minimum}[{index}])'
          units='microVolts'/>
  <calibration_offset evaltype='float' index='1..{number_of_channels}'
          eval='{physical_minimum}[{index}]-{digital_minimum}[{index}]*{calibration_gain}[{index}]'/>
  <channel_names type='ascii' width='16' index='1..{number_of_channels}'
          offset='256+16*{index}'/>
</parameters>
</meta_format>
```

Table 1
Metainformation for the European Data Format. Original definition of the EDF header is available from http://www.hsr.nl/edf.

```
<?xml version="1.0"?>
<annotations>

<datafile_identification>
 <format id='EDF'/>
 <name>example_datafile.edf</name>
 <signature method='crc32' offset='111' length='512' value='07af0b0d'/>
</datafile_identification>

<signal_transform>
</signal_transform>

<tag_definitions>
   <def_group name='HYPNOGRAM'  fixed_length='2560' offset_quant='2560'
             channels='all' ref_channel_number='1'>
     <tag_item name='4' description='stage IV'/>
     <tag_item name='3' description='stage III'/>
   </def_group>

   <!-- marking transients in C3 -->
   <def_group name='C3_transients' channels='10'>
       <tag_item name='S' description='sleep spindle'/>
       <tag_item name='W' description='slow wave'/>
   </def_group>

   <def_group name='events'>
       <tag_item name='0' description='other event in arbitrary channel'/>
   </def_group>

</tag_definitions>

<tag_data>
   <text_info>
     Hypnogram and transients by dr. X,  artifacts detected automatically
     by http://EEG.pl/artifacts/ with parameters: threshold1=50%, threshold2=50%, ...
  </text_info>
<tags>
 <tag name='4' quantized_offset='136'/>

 <tag name='S' position='1927' length='86'>
     <annotation>I'm not quite sure whether it's really a sleep spindle...</annotation>
 </tag>

 <tag name='0' position='3234' length='25600' ref_channel_number='1'>
     <channels number='7,9'/>
     <annotation>Electrodes not contacting</annotation>
 </tag>
</tags>
</tag_data>

</annotations>
```

Table 2
Example annotations to an EDF file