XML Representation Languages as a Way of Interconnecting TTS Modules

Marc Schröder

DFKI GmbH, Saarbrücken, Germany
schroed@dfki.de

Stefan Breuer

IKP, University of Bonn, Germany
breuer@ikp.uni-bonn.de

Abstract

The present paper reports on a novel way of increasing the modularity and pluggability of text-to-speech (TTS) architectures. In a proof-of-concept study, two current TTS systems, both using XML-based languages for internal data representation, are plugged together using XSLT transforms as a means of translating from one system's internal representation to the other's. This method allows one system to use modules from the other system. The potential and the limitations of the approach are discussed.

1. Introduction

Current text-to-speech (TTS) systems are modular in principle [1, 2] but monolithic in practice. Partial processing results are stored in system-specific internal representation formats that tend not to be easily exported without loss of information, and often cannot be imported back into the system. This makes partial processing impracticable; in particular, the non-initial modules of a TTS system cannot easily be driven by externally-produced input.

The present paper explores the improvement on this state of affairs that may arise from XML-based internal representation formats found in recent TTS systems [3, 4]. Systems using XML internally can be made to export or import intermediate processing results without any loss of information. This means that one system's partial processing output can serve as another system's input at a corresponding processing step if a conversion between the two formats is possible. It will be shown that the syntactic conversion between two very different-looking XML formats is easy to achieve, as long as the information represented in the two is sufficiently similar.

The paper is organised as follows. First, the concept of XML-based representation languages is defined. The two systems used in this paper are introduced, including the properties of the respective representation languages. It is then demonstrated, using two examples, how a module from one system can be used within the other system. Finally, it is discussed what possibilities and limitations can be foreseen for a wider application of the proposed method.

2. Distinguishing markup and representation languages

The present paper is concerned with XML-based representation languages, a notion which is not yet well known and easily confused with XML-based input markup languages, which serve a very different purpose.

2.1. XML-based markup languages

XML-based markup languages provide relatively high-level markup functionality for speech synthesis input, and are intended for the use of non-experts. This group includes the upcoming W3C standard SSML (speech synthesis markup language, [5]) as well as its predecessor, SABLE. These markup languages aim at giving a non-expert user the possibility to add information to a text in order to improve the way it is spoken. They are (at least in principle) independent of any particular TTS system. Systems are assumed to parse the markup enriching their input and translate the information contained in it into a system-internal data representation format which in most cases is not XML-based.

A related markup language is VoiceXML [6], the main focus of which is speech access to the internet.

2.2. XML-based representation languages

The purpose of an XML-based representation language is to serve as the data representation format *inside* a TTS system. For that reason, the concepts represented in it are low-level, detailed, and specific to the design decisions, modules, and scientific theories underlying the TTS system. By means of the Document Object Model (DOM), a standardised object-oriented representation of an XML document, the TTS system modules can operate directly on the XML document, interpreting and adding information. The MARY [3] and BOSS [4] systems (see below) each have their own XML-based representation language.

XML representations can easily be exported to a textual form at any state of processing. As the external XML document contains the complete data, it can as easily be read back into the system, and processing can continue from that step onwards.

3. Systems overview

The present section gives a short summary of the two systems used in the current study. With some abstraction, both systems can be conceived of as following the general TTS architecture[1] represented in Figure 1.

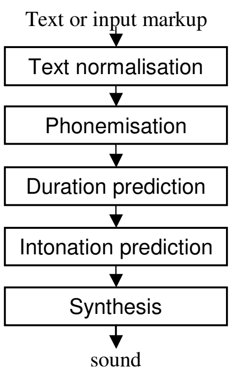

Figure 1: A simplified general TTS architecture.

3.1. The MARY system

3.1.1. Overall architecture

The MARY system [3] is a TTS server written in Java, created at DFKI with support from the Phonetics and Computational Linguistics departments at Saarland University. It is a very flexible toolkit allowing for easy integration of modules from different origins. For German, the general TTS architecture (Fig. 1) is instantiated in MARY as follows.

Text normalisation consists of an optional input markup parser converting SSML into MaryXML; a tokenizer; a preprocessing component converting numbers, abbreviations etc. into pronouncable form; a part-of-speech tagger and chunker (local syntactic parser); and an information structure module recognising givenness and contrast based on text structure, optionally using a semantic database.

Phonemisation is performed using a custom DFKI pronunciation lexicon compiled into a finite state transducer, complemented with letter-to-sound rules.

Duration prediction is carried out using a version of the Klatt rules [7] manually adapted to German.

Intonation prediction is carried out in two rule-based steps. First (actually before duration prediction), symbolic GToBI labels [8] are predicted; second, these symbolic labels are translated into frequency-time targets.

The synthesis module is instantiated using several synthesis engines, among them MBROLA [9].

For English, the Mary system uses a number of modules from the open-source FreeTTS system derived from FESTIVAL [2]. Use of these modules is made possible by mapping MaryXML to the multi-layered "Utterance" structure used in FreeTTS and vice versa.

3.1.2. MaryXML syntax

The syntax of a MaryXML document reflects the information required by the modules in the TTS system. Concepts which can also be encoded in speech synthesis input markup languages, such as sentence boundaries and global prosodic settings, are represented by the same tags as used in the W3C SSML specification [5].

Most of the information to be represented in MaryXML, however, is too detailed to be expressed using tags from input markup languages. Specific MaryXML tags represent the low-level information required during various processing steps.[2]

The MaryXML syntax was designed to maintain a certain degree of readability for the human user, by keeping information redundancy at a minimum.

3.2. The BOSS system

3.2.1. Overall architecture

The Bonn Open Synthesis System [4] is an open source client/server architecture for non-uniform unit selection synthesis implemented in C++ under Linux. BOSS was designed at IKP with contributions from IPO, Eindhoven. BOSS relates to the general TTS architecture as follows.

Text normalisation is performed by a user-supplied client application, which also creates the BossXML structure from plain input text (TTS) or text enriched with markup (CTS). Network-enabled demonstration clients for TTS purposes exist for Windows and Linux.

Phonemisation is supplied by the boss_transcription module, which uses the Bonn Machine-Readable Pronunciation Dictionary (BOMP) [10] to generate the syllabic and phonetic structure from input graphemes. boss_transcription handles unknown words by attempting morpheme decomposition, or, if this fails, by grapheme-to-phoneme conversion using decision-trees. The latter are also used for the assignment of lexical stress.

Duration prediction is done by means of Classification and Regression Trees (CART).

The intonation module is based on the Fujisaki model [11] for the parameterisation of F0 contours. These parameters are predicted by a neural network at syllable level. An alternative module for F0 prediction is under development at IKP.

The synthesis module in BOSS consists of two parts: The unit selection module assigns costs to words, syllables, phones and, if available, half-phones from the

[1]The separation into modules is made in order to structure the following presentation. No claim is made that this Figure adequately describes all existing TTS systems.

[2]A full XML Schema-based definition of MaryXML is available online at `http://mary.dfki.de/lib/MaryXML.xsd`.

database and selects the segments; these are retrieved and concatenated in the final module, which is also responsible for prosodic manipulation. At present, only boundary smoothing is applied.

3.2.2. BossXML syntax

BossXML was designed for efficient processing at run time. It maps the linguistic levels word, syllable and phoneme onto a hierarchical element structure. In the course of synthesis, these levels are added to the XML structure, as soon as their contents are known. For words, this is the case after text normalisation. Syllables and phonemes are added by the transcription module. Every node contains all the information pertaining to it, thus no recourse to higher or lower levels has to be taken. In contrast to MaryXML, redundancy is high in BossXML, with the advantage that the programmer of a module only has to care about adding the information generated by the module, while the retrieval of pre-existing information is straightforward.

4. Plugging system components together

In order to demonstrate the feasibility of the proposed method, two use cases were implemented in which a module from one system is used in the other system: a) use of the BOSS phonemisation in MARY; and b) use of the MARY duration prediction in BOSS.

4.1. BOSS phonemisation in MARY

In order to use the BOSS phonemisation in MARY, the phonemisation input format must be translated from MaryXML into BossXML, and the phonemisation output format must be translated back from BossXML into MaryXML. In the following, the output of the phonemisation module corresponding to the word “Hallo” in the sentence “Hallo Welt.” (engl. “Hello World.”) is shown in MaryXML and in BossXML.[3]

```
BossXML (source):
<WORD Orth="Hallo" ExtInfo="pos:ITJ"...>
<SYLLABLE TKey="ha" Stress="1"...>...</SYLLABLE>
<SYLLABLE TKey="lo:" Stress="0"...>...</SYLLABLE>
</WORD>

MaryXML (target):
<t pos="ITJ" sampa="'ha-lo:">Hallo</t>
```

The conversion from the MaryXML to the BossXML structure is performed using an XSLT stylesheet. As MaryXML contains part-of-speech information that cannot be represented in BossXML, the `ExtInfo` attribute is used as a simple feed-through mechanism to preserve the external information (see also 5 below).

[3]most BossXML attributes and some substructure was omitted for space reasons; the full documents can be found at `http://www.dfki.de/~schroed/maryboss2004`

After phonemisation, the BossXML <`WORD`> element contains a substructure of syllables and phonemes, richly annotated with features relevant for unit selection. In MaryXML, phonemiser output consists of a compact “sampa” attribute added to the <`t`> element. Again, an XSLT stylesheet performs the conversion from BossXML back into MaryXML. Information about syllable boundaries and stress, represented by the XML element structure and attributes in BossXML, is converted into sampa diacritics for MaryXML. The part-of-speech information transparently “fed through” the BOSS system by means of the `ExtInfo` attribute is converted back into a MaryXML attribute.

4.2. MARY duration prediction in BOSS

Another example of XML-based module integration is the use of the MARY duration prediction module in the BOSS system. Again, two conversions are necessary: The input to the duration prediction module must be converted from BossXML to MaryXML, and the module output must be converted back from MaryXML to BossXML.

The latter step is somewhat more complicated than the other conversions required so far because of the rich sub-structure of <`WORD`> elements in BossXML. It provides information about the context explicitly which in MaryXML must be deduced from the surrounding XML structure. The following example shows the duration prediction output for the syllable [lo:] of “Hallo Welt”.

```
MaryXML (source):
<syllable sampa="lo:">
<ph d="60" end="206" p="l"/>
<ph d="106" end="312" p="o:"/>
</syllable>

BossXML (target):
<SYLLABLE Stress="0" PMode="" PInt="0"
  CCRight2="LAB" CCRight="v" CRight="v"
  CCLeft2="CEN" CCLeft="a" CLeft="a"
  TKey="lo:" Dur="166">
<PHONEME Stress="0" PMode="" PInt="0"
  CCRight2="BAC" CCRight="o" CRight="o:"
  CCLeft2="CEN" CCLeft="a" CLeft="a"
  TKey="l" Dur="60"/>
<PHONEME Stress="0" PMode="" PInt="0"
  CCRight2="LAB" CCRight="v" CRight="v"
  CCLeft2="ALV" CCLeft="l" CLeft="l"
  TKey="o:" Dur="106"/>
</SYLLABLE>
```

The XSLT stylesheet performing the conversion needs to analyse the syllable and phoneme contexts in the MaryXML document and add the information required. Because of the flexibility of XSLT transforms, this can be done with reasonable effort.

5. Discussion

The method proposed for converting one TTS system’s internal data representation into another’s is powerful insofar as syntactic conversion is concerned, which may include complex inference algorithms. It allows researchers and system developers to connect systems, provided that the information used in one system can either be directly converted or at least be generated from the information used in the other system.

A natural limitation of the method is the science underlying the different TTS systems. If the approaches to a given phenomenon pursued in the two systems are so different that no mapping between them is known, then all the syntactic power of XSLT will obviously not be able to solve the underlying scientific question. For example, if one system models prosody in terms of superimposed intonation contours [11] and the other uses a model based on frequency-time targets [8], it will not sensibly be possible to exchange prosody-related data between the two systems. It may nevertheless be possible to interconnect most of the modules from these systems: The modules prior to the "intonation prediction" module (see Fig. 1) are unaffected by the incompatibility, and subsequent modules may be able to operate with an approximation of the required information.

The second use case presented above (see 4.2) required such an approximation. The Klatt-rule-based MARY duration module uses the concept of "accented syllable", in the sense of phrase accent as opposed to word stress, in order to predict segment duration. This information is not provided by the BOSS system. The module must therefore run on limited information and will predict shorter durations for "accented" syllables. In the current use case, where the module output is fed into the BOSS synthesis, this effect may not actually be very damaging, given the fact that the BOSS modules do not take "accent" into account.

On the other hand, if appropriate "feed-through" mechanisms exist in an XML-based representation language, it is possible to preserve information from one system while processing data with another system in which this information cannot be represented. A first crude approximation of such a mechanism is the `ExtInfo` attribute, available in BossXML, which may contain an arbitrary string value. During XSLT transformation from system A to system B, incompatible information can be stored within such a tag, which is ignored by system B but preserved in its output, so that it can be decoded by the XSLT transformation of the processing result back to system A. In the first use case (see 4.1) this method was used for preserving part-of-speech information. In the future, it may be necessary to devise more elaborate feed-through mechanisms which can also represent the sub-structure of words. In the second use-case (see 4.2), such a mechanism would have made it possible to avoid re-creating the complex BossXML structures.

6. Acknowledgements

Part of this research is supported by the EC Projects NECA (IST-2000-28580) and HUMAINE (IST-507422).

7. References

[1] T. Dutoit, *An Introduction to Text-to-Speech Synthesis*. Dordrecht: Kluwer Academic, 1997.

[2] A. Black, P. Taylor, and R. Caley, "Festival speech synthesis system, edition 1.4," CSTR, University of Edinburgh, UK, Tech. Rep., 1999. http://www.cstr.ed.ac.uk/projects/festival

[3] M. Schröder and J. Trouvain, "The German text-to-speech synthesis system MARY: A tool for research, development and teaching," *Intl. J. Speech Technol.*, vol. 6, pp. 365–377, 2003. http://mary.dfki.de

[4] E. Klabbers, K. Stöber, R. Veldhuis, P. Wagner, and S. Breuer, "Speech synthesis development made easy: The Bonn Open Synthesis System," in *Proc. Eurospeech*, Aalborg, Denmark, 2001, pp. 521–524. http://www.ikp.uni-bonn.de/boss

[5] M. R. Walker and A. Hunt, *Speech Synthesis Markup Language Specification*, W3C, 2001. http://www.w3.org/TR/speech-synthesis

[6] *VoiceXML 2.0 Specification*, VoiceXML Forum, 2004. http://www.voicexml.org

[7] D. H. Klatt, "Synthesis by rule of segmental durations in English sentences," in *Frontiers of Speech Communication*, B. Lindblom and S. Öhman, Eds. New York: Academic, 1979, pp. 287–299.

[8] M. Grice, S. Baumann, and R. Benzmüller, "German intonation in autosegmental-metrical phonology," in *Prosodic Typology*, S.-A. Jun, Ed. Oxford University Press, 2002.

[9] T. Dutoit, V. Pagel, N. Pierret, F. Bataille, and O. van der Vrecken, "The MBROLA project: Towards a set of high quality speech synthesisers free of use for non commercial purposes," in *Proc. 4th ICSLP*, Philadelphia, USA, 1996, pp. 1393–1396.

[10] "Bonn Machine-Readable Pronunciation Dictionary (BOMP)." http://www.ikp.uni-bonn.de/dt/forsch/phonetik/bomp/BOMP.en.html

[11] H. Mixdorff and H. Fujisaki, "The influence of focal condition, sentence mode and phrase boundary location on syllable duration and the F0 contour in German," in *Proc. 14th ICPhS*, San Francisco, USA, 1999, pp. 1537–1540.