Text-to-Speech Scripting Interface for Appropriate Vocalisation of e-Texts

Gerasimos Xydas and Georgios Kouroupetroglou

University of Athens, Department of Informatics and Telecommunications,
Division of Communication and Signal Processing
{gxydas, koupe}@di.uoa.gr

Abstract

Electronic texts carry important meta-information (such as tags in HTML) that most of the current Text-to-Speech (TtS) systems ignore during the production of the speech. We propose an approach to exploit this meta-information in order to achieve a detailed auditory representation of an e-text. The e-Text to Speech and Audio (e-TSA) Composer has been designed and developed as an XML based scripting framework that can be adopted by existing TtS, with minor or major modifications. It provides a mechanism to create scripts using combined elements from e-texts and TtS systems. The e-TSA Composer can manipulate the behaviour of a TtS (e.g. the applied prosody) in order to define a finest vocalisation in response to specific e-texts.

1. Introduction

Spoken information provision in emerging environments, like in auditory-only interfaces, virtual reality and e-commerce raise new issues for text-to-speech (TtS) systems. Apart from ordinary plain text, novel elements, most of them appear in a repeatable way, are set for vocalisation. For example, an HTML page has `<paragraph>`s and `<table>`s under `<body>`. An MS-Word document has sections where different styles are defined. These are elements that support the visualisation of the documents and in most cases carry important meta-information about the text (e.g. bold letters usually imply emphasis). The introduction of eXtended Markup Language (XML) [1], as well as other mark-up languages, makes this meta-information even more perceivable. How this meta-information can be audibly embedded in the vocalisation of the text information comprises an important research issue.

Furthermore, several works have dealt with the importance of mixed speech and non-speech signals to support the presentation of structures, such as tables, lists, etc. This becomes more essential in cases of auditory-only interface [2]. Other works (e.g. [3]) propose the insertion of non-speech audio signals, such as beeps, while other focus on the prosodic variations and speaker-style changes [4].

In order to efficiently interpret etexts in a vocalised manner, there should be a TtS mechanism that would allow declarations like: *For every <title> you come across in an HTML document, insert a beep and read it out loudly and in a staccato way.* This differs from the scope of VoiceXML [5] and other speech-aware mark-up languages (e.g. [6]) in that VoiceXML: (a) has its own format, (b) presents an abstraction of a part of the functionality of a TtS and (c) it is already speech-aware. However, the majority of the existing e-texts lack speech awareness.

Most of the TtS systems don't incorporate a facility to associate specific meta-information of the source document with customised speech and audio behaviour for enabling a detailed auditory representation of an e-text. They mainly deal with properly syntactical structured texts and there is not provision for handling other kinds of meta-information. In this work, we make an effort to accommodate that through the *e-TSA Composer*.

As we need a way to refer to the parts that constitute a formatted document (like the elements on XML), we introduce the notion of *cluster*. A cluster is a representative of a part of an e-text. In this paper essentially we target to a scheme where TtS related properties and source's e-text clusters could be combined on a recursive base. This will enable the description of the auditory behaviour of the TtS on response to specific clusters.

The general requirements to achieve that goal in TtS systems should be:

1. Identification and handling of clusters in the source e-text and in a way that can be further combined with TtS system's properties.
2. Declaration of TtS system's properties in an appropriate way.
3. Combination of the above descriptions into an instruction set that can control TtS system's operation.
4. Interpretation the above instruction set.

Proc. Of EUROSPEECH 2001, Sept. 3-7, Aalborg, Denmark, pp. 2247-2250

In the next paragraph we briefly present the e-Text to Speech and Audio (e-TSA) Composer. Then, we propose the format of the scripts and how a TtS can adopt this approach. In section 4 we discuss the flexibility as well as the perspective of our work.

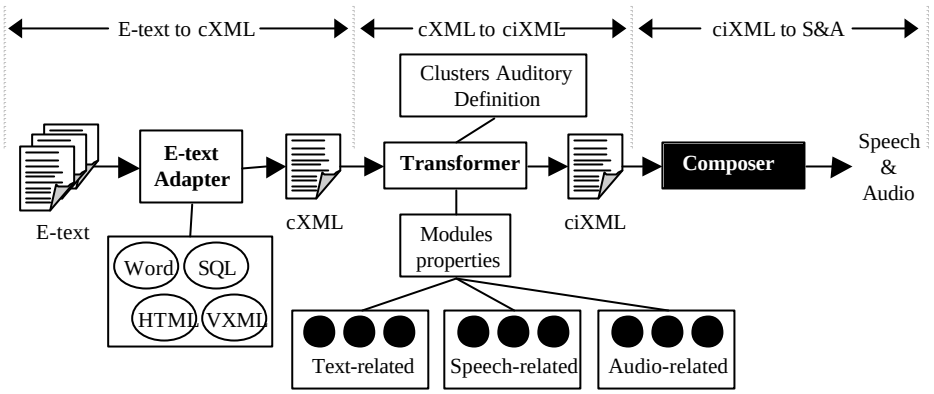

Figure 1: The architecture of the e-TSA Composer. Black cycles refer to the modules of the underlying TtS.

2. The e-TSA Composer

To fulfil the basic requirements described above, we have designed the e-Text to Speech and Audio (e-TSA) Composer framework (Figure 1). Its mainly feature is to allow clusters' and TtS' properties to be mixed under a single scripting language. Actually, we do not propose a new scripting language, but we gain from the power of XSLT [7] to transform XML documents. Thus, the e-TSA Composer is mainly based on XML, which furthermore is widely used on cross-platform applications and also there are many related authoring tools freely available.

The e-TSA Composer assumes an underlying generic TtS and can be seen as an extension or an add-on to it. This approach targets to a cross-TtS use.

Figure 1 presents the architecture and the flow of information in the eTSA Composer. The parts that are closely coupled with the underlying TtS have been marked with black colour. Three major steps are identified:

1. The translation of e-texts to an appropriate composer-XML format (cXML). Specialised modules handle specific e-text formats (e.g. HTML to cXML, MS-Word to cXML).
2. The transformation of the cXML document to a set of instructions for the underlying TtS, in the form of a composer-interface XML (ciXML). We will present this in detail later.
3. The interpretation of the ciXML document. This depends heavily on the underlying TtS, as it should provide means to handle such documents, and this is one of the modifications that need to be done in the TtS. The second one is to properly export its properties.

2.1. Exporting the TtS properties

TtS properties consist of parameters that customize its functionality (e.g. the ratio that is applied to the duration of a syllable at the end of a phrase). We further assume that the functionality of a TtS is controlled by a set of modules. This is usually the case in most modern TtS systems (eg. [8] and [9]).

We have adopted the Document Type Definition (DTD) format, which suits perfectly to the XML scheme of the Composer in order to have a uniform definition of the TtS properties. Each module should provide a DTD defining a unique namespace and a set of parameters accommodated with valid values that customize it. The namespace act as the identification of the module in order to be referred during scripting.

Figure 2 shows the way that properties are being exported to a DTD format. The algorithm is as follow:

For each parameter P_{xi} of type T_x with a range of valid values $V_{x1}, V_{x2}, \ldots, V_{xN}$ define:

```
<!ELEMENT Pxi (…)>
<!ATTLIST Pxi (Vx1|Vx2|…|VxN)>
```

The first three dots (…) refer to specific implementation issues (can be EMPTY, #PCDATA, …), which is out of the scope of this paper.

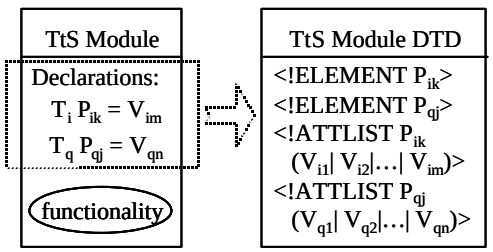

Figure 2: The export of module's parameters to a DTD document. On the left, parameter P_{ik} is of type T_i and

has been assigned the value V_{im}. On the DTD, parameter P_{ik} can been assigned a value among the valid V_{i1}, V_{i2},…V_{im}.

For example, assume a RHYTHM module capable of (a) applying several tempo patterns and (b) modify the duration of syllables based on their position in a phrase. This module can be represented by the following DTD:

```
<!ELEMENT Rhythm (Tempo*, Syllable*)>
<!ELEMENT Tempo (#PCDATA?)>
<!ATTLIST Tempo speed
(lento|andante|allegro)>
<!ATTLIST Tempo acc_var (legato|staccato)>
<!ELEMENT Syllable (#PCDATA?)>
<!ATTLIST Syllable place (begin|end)>
<!ATTLIST Syllable ratio (#PCDATA)>
```

The above does not imply that both elements should be used at the same time. Instead, one can use what suits better her/his case.

Figure 3 presents the cycle of customisation of a TtS module's properties. The role of the "Transformer" and the "Composer" will be presented in detail in next section.

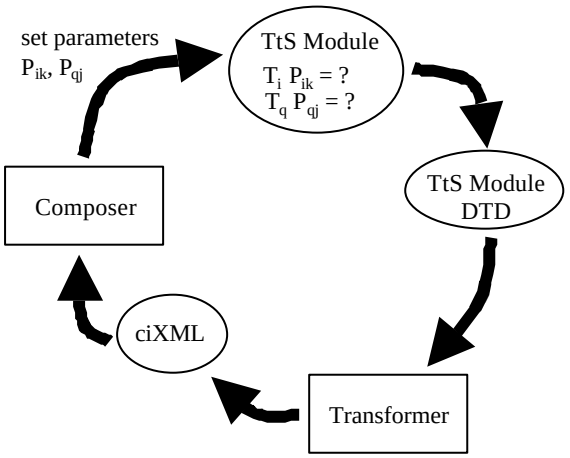

Figure 3: The cycle of modules' customisation.

3. Cluster Auditory Definition (CAD)

The cXML document produced during step 1 in Fig. 1 includes elements (like `<book>`, `<title>` and `<chapter>`) either just one or many times. These elements represent the clusters of the source e-text. In order to associate a customised auditory behaviour to the clusters, we introduce the notion of the Cluster Auditory Definition (CAD), which refers to the speech and audio behaviour of the TtS for the specific cluster. A CAD consists of:

1. A pointer to a cluster of the source e-text and
2. A script that describes the speech and audio behaviour of the TtS.

The CAD actually forms a link between the cluster and a detailed programmed behaviour of the TtS when dealing with that cluster.

The mechanism for making this association is based on transforming the cXML document. This transformation takes into account the modules' DTDs and has recursive properties.

3.1. CAD scripts

A script in a CAD must be able to make use of any service and data available in the TtS. Thus, the TtS should have properly exported its features in corresponding DTDs.

Assume that we have an HTML document that among other things contain the lines:

```
<h1>A new therapy on the way</h1>
<p>blah blah
<a href="http://news.com/article1">
Click here</a></p>
```

The "Text Adapter" of Figure 1 can transform this to a cXML document:

```
<headline>A new therapy on the
way</headline>
<details>blah blah
<link href="http://news.com/article1">
Click here</link></details>
```

This cXML document defines three clusters (we refer to them using their XPath in the cXML):

a. `/headline`
b. `/details`
c. `/details/link`

One can now describe the auditory features of her/his preference for each cluster by writing a CAD script to vocalise it. For example:

1. `/headline`: apply a "staccato" rhythm pattern
2. `/details`: apply normal prosody
3. `/details/link`: insert a beep at the beginning, read it out loudly and expand syllables by 1.2 times at the end of a phrase or by 1.4 times at the beginning.

To create a CAD script, we use transformation rules in XSLT template formats. So, the template for the case of the CAD number 3 above (`/details/link`) would be as follows:

```
<template match="/details/link">
   <SOUND:Insert file="beep.wav"/>
   <ENERGY:Loud>
```

```
   <RHYTHM:Syllable place="end"
ratio="*1.2">
   <RHYTHM:Syllable place="begin"
ratio="*1.4">
      <apply-template/>
   </ENERGY:Loud>
   </RHYTHM:Syllable>
   </RHYTHM:Syllable>
</template>
```

This template implies that whenever a `/link` cluster is offered in the source HTML document, under a `/details` cluster, it will be vocalised using the behaviour described in the template. Furthermore, this inherits to any instance of the above cluster is found in the source HTML document.

3.2. Interface with the underlying TtS

After applying the above template, the produced document embeds instructions for customising the functionality of the TtS. The result is formatted in composer-interface XML (ciXML) type and looks like the following:

```
<SOUND:Insert file="beep.wav"/>
<ENERGY:Loud>
<RHYTHM:Syllable place="end" ratio="*1.2">
<RHYTHM:Syllable place="begin"
ratio="*1.4">
Click here.
</ENERGY:Loud>
</RHYTHM:Syllable>
</RHYTHM:Syllable>
```

This description is a complete instruction set for the underlying TtS. It can be interpreted in a row to set up the parameters that control it.

One important feature that supports the safe interpretation of the scripts is that the TtS can ignore everything it does not understand. This way the TtS can perform its best that suits the script. Another note here is that there is the option for the TtS to rename a namespace, if it is defined otherwise locally.

4. Discussion

The e-TSA Composer embeds features of a speech mark-up language, as it produces ciXML, a marked-up document that includes information about the vocal representation of a text. However, ciXML is more flexible (but on the other hand complicated) as it allows a detailed programming of the TtS. So, instead of simply saying "slow rate" one can further define properties of "slow rate" (e.g. stretching the final syllables by a factor, hot to slow down pitch transition etc), though a TtS can provide a separate "slow rate" instruction as well. In VoiceXML, the perception of "slow rate" is left to the view of the TtS. Thus, the e-TSA Composer allows finest and more predictable description of the prosodic and other non-speech features.

In [10] XML has been used as a mean to compute and represent the hierarchical linguistic structures of a text in a TtS and further effort has been made on its use during speech generation [11][12]. In contrast, eTSA Composer mainly deals with non-speech-aware applications and how to use XML as a scripting language to associate elements of the source with auditory details. Furthermore, CAD scripts can be used to deal with speech marked-up documents or in concept-to-speech systems.

A further advantage of our methodology is that one can now control the TtS using XML authoring tools.

The e-TSA Composer has been implemented as a dynamically linked module in DEMOSTHeNIS Speech Composer [13] to enable the audible parsing of HTML documents. Though we achieved the basic requirements, more effort should be employed on standardising the definitions of modules, thus allowing e-TSA to work in a cross-TtS manner.

5. Conclusions

In order to fulfil the need for exploiting the meta-information of e-texts during their auditory representation, we presented a scripting framework that allows the recursive association of text's elements and TtS speech and audio events. The e-TSA Composer has been proposed based on XML for being adopted by existing TtS systems.

6. Acknowledgements

The work described in this paper has been partially supported by the M-PIRO project of the IST Programme of the European Union under contract no IST-1999-10982.

7. References

[1] http://www.w3c.org/XML

[2] Mitsopoulos, E., *A Principled Approach to the Design of Auditory Interaction in the Non-Visual User Interface*, Submitted for the degree of Doctor of Philosophy, University of York, UK, 2000

[3] Hakulinen, J., Turunen, M. and Raiha, K., *The Use of Prosodic Features to Help Users Extract Information*

from Structured Elements in Spoken Dialogue Systems, In Proceedings of ESCA Tutorial and Research Workshop on Dialogue and Prosody, Eindhoven, The Netherlands, pp.65-70, 1999

[4] Shriver, S., Black, A. and Rosenfeld, R., *Audio Signals in Speech Interfaces*, In Proceedings of International Conference on Spoken Language Processing (ICLSP-2000), Beijing, China, 2000

[5] *Voice eXtensible Markup Language (VoiceXML™) version 1.0*, W3C Note 05 May 2000, http://www.w3.org/TR/voicexml/

[6] Sproat, R., Taylor, P., Tanenblatt, M. and Isard, A., *A markup language for text-to-speech synthesis*, In Proceedings of Eurospeech97, Rhodes, Greece, pp. 1747-1750, 1997

[7] *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation 16 November 1999, http://www.w3.org/TR/xslt

[8] Taylor, P., Black, A. and Caley, R., *The architecture of the Festival Speech Synthesis System*, 3rd ESCA Workshop on Speech Synthesis, Jenolan Caves, Australia pp. 147-151, 1998

[9] Dutoit, T., Bagein, M., Malfrere, F., Pagel, V., Ruelle, A., Tounsi, N. and Wynsberghe, D., *EULER : an Open, Generic, Multi-lingual and Multi-Platform Text-To-Speech System*, In Proceedings of LREC'00, Athens, Greece, pp. 563-566, 2000.

[10] Huckvale, M., *Presentation and Processing of Linguistic Structures for an All-Prosodic Synthesis System Using XML*, In Proceedings of Eurospeech99, Budapest, Hungary, pp 1847-1850, 1999

[11] Huckvale, M., *The Use and Potential of Extensible Mark-up (XML) in Speech Generation*, Working paper, COST258 Naturalness of Synthetic Speech, (to appear), 2001

[12] Hitzeman, J., Black, A., Mellish, C., Oberlander, J., Poesio, M. and Taylor, P., *An annotation scheme for concept-to-speech synthesis*, In Proceedings of European Workshop on Natural Language Generation, Toulouse, France, pp. 59-66, 1999

[13] Xydas, G. and Kouroupetroglou, G., *DEMOSTHeNIS Composer*, Technical Report, University of Athens, Department of Informatics and Telecommunications, Greece, 2001