

SSML und Sprachsynthese

Artikel und Referat ausgearbeitet von [Hannes Restel](#) im WS06/07

Sinn und Zweck von Sprachsynthese bzw. [SSML](#) ist die computerseitige Umwandlung von Text in Sprache; der Computer vermag es also, Texte mit Hilfe synthetischer Stimmen vorzulesen. Dies ist insbesondere für interaktive Telefonesysteme oder den barrierefreien Zugang zum Internet wünschenswert.

Sprachsynthese vs. [SSML](#)

Unter Sprachsynthese wird die reine Umwandlung eines schriftlichen Texts in eine akustische Form verstanden. Da maschinelle Sprachsynthese-Systeme jedoch keinerlei semantische Kenntnisse über den Text besitzen - also kein Textverständnis besitzen - wurde die SSML (*Speech Synthesis Markup Language*) entworfen. Mit ihr ist es möglich einen zu sprechenden Text mit Metainformationen zu versehen, welches die korrekte Aussprache für das Sprachsynthese-System vereinfacht bzw. ermöglicht.

SSML ist jedoch nicht Teil eines Sprachsynthese-Systems, sondern steuert bereits vorhandene Synthese-Systeme an, stellt also eine neue Ebene zwischen Texteingabe und Sprachausgabe dar.

Einfaches Beispiel für die Relevanz von Textverständnis:

Wie wird der geschriebene Ausdruck $1/4$ ausgesprochen? Je nach Kontext könnte es

- "ein Viertel"
- "Eins zu Vier"
- "Erster April"
- oder nach amerikanischem Format gar "Vierter Januar"

sein. Unter Verwendung der SSML wird das Ausspracheverhalten festgelegt.

Es folgt nun eine Erklärung der Sprachsynthese und der Technik aktueller Sprachsynthese-Systemen und anschließend eine Erläuterung der [SSML](#).

Sprachsynthese

Geschichte der Sprachsynthese

Bereits sehr früh hat sich die Menschheit Gedanken über Sprachsynthese gemacht. Dabei stand im Vordergrund, den menschlichen Körper mittels Maschinen nachzuahmen.

Die ersten Sprachsynthese-Systeme soll es angeblich bereits um 1000 n.Chr., wo jedoch keine Artefakte erhalten sind.

1791 schließlich gelang es Wolfgang von Kempelen, eine Maschine zu bauen, die den menschlichen Vokaltrakt nachahmte. Diese Gerät besaß künstliche (i.e. mechanische) Lungen (*Blasebalg*), Stimmbänder (*Gummiband*), Lippen (*Gummitrichter*), Mund und Rachen. Das Gerät konnte wie ein Rucksack auf dem Rücken getragen werden und wurde mit beiden Händen und Armen bedient. Es war bereits möglich, Wörter und einige kurze Sätze zu synthetisieren. Dieses und alle folgenden Beispiele haben gemein, dass die Synthese in Echtzeit geschieht, eine Eingabe also umgehend in eine Aussprache umgewandelt wird. Man möge hier an ein Klavier denken, wo ein Tastendruck unmittelbar zu einem hörbaren ton führt.

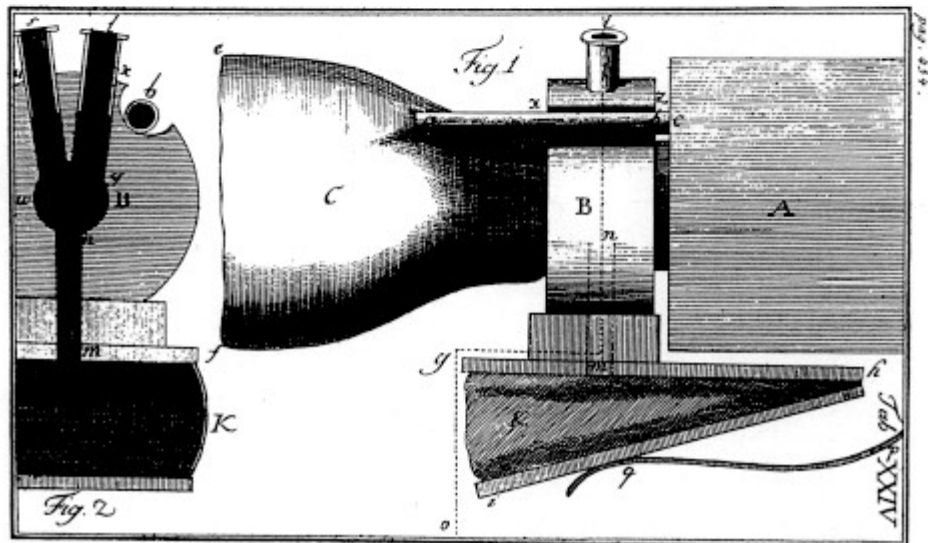
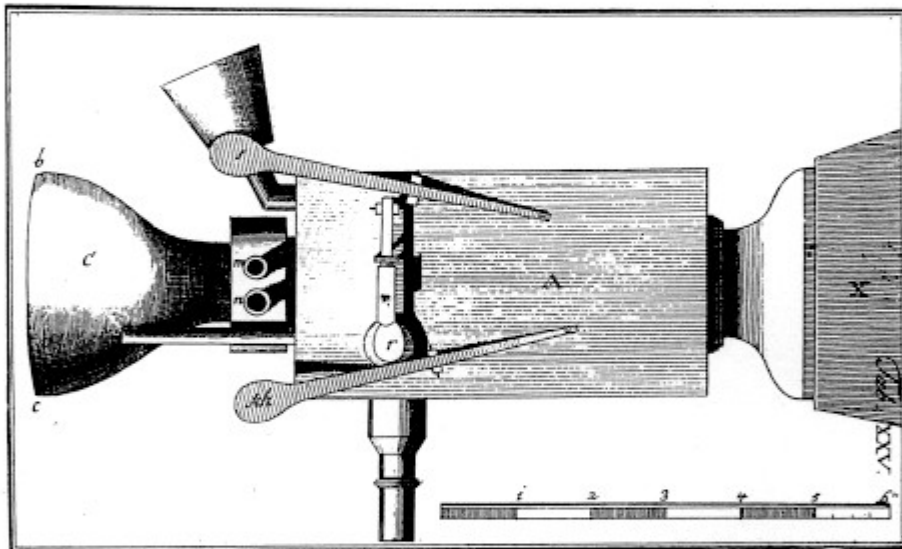
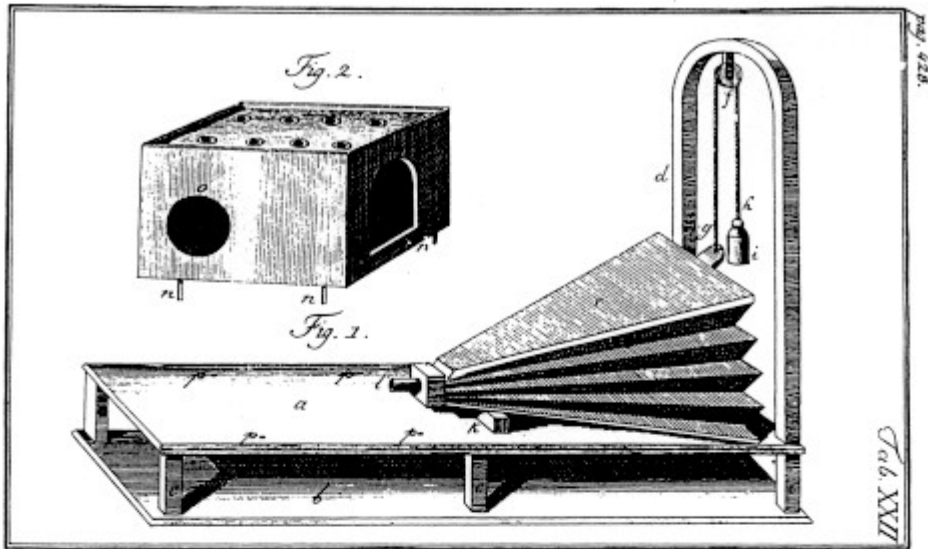


Abbildung 1. Apparat

von Wolfgang von Kempelen

Knapp 140 Jahre später - im Jahre 1835 - baute Joseph von Faber ein Gerät, welches gesprochene, gesungene und geflüsterte Sprache synthetisieren konnte. Dieses Gerät war stationär und wurde



Abbildung 2.

über eine Art Klaviatur bedient.
Apparat von Joseph von Faber

1939 war es Homer Dudley, der mit seinem *Voder* eines der ersten elektronischen Sprachsynthese-Systems baute. Hervorgegangen war der *Voder* aus einer Entwicklung des MIT (Massachusetts Institute of Technology). Dieses Gerät wurde zur Weltausstellung 1940 in Paris präsentiert, wo folgendes Hörbeispiel erhalten ist: [AUDIO1 - Hörbeispiel Voder](#)



Abbildung 3. Voder von Homer Dudley

1950 erfand Frank Cooper sein *Pattern Playback*, ein optoelektronisches Synthese-System. Dieses Gerät kann eventuell die Urstimme des Roboters gelten, was an folgendem Hörbeispiel deutlich wird: [AUDIO2 - Hörbeispiel Pattern Playback](#). Sprache wurde generiert, indem bestimmte Bereiche einer sich drehenden Scheibe (auf denen jeweils Laute gespeichert wurden) mit einem Lichtprisma fokussiert wurden, so dass die resultierenden modulierten Lichtwellen von einer Photozelle in elektrische Signale umgewandelt und anschließend von einem Lautsprecher ausgegeben wurden.

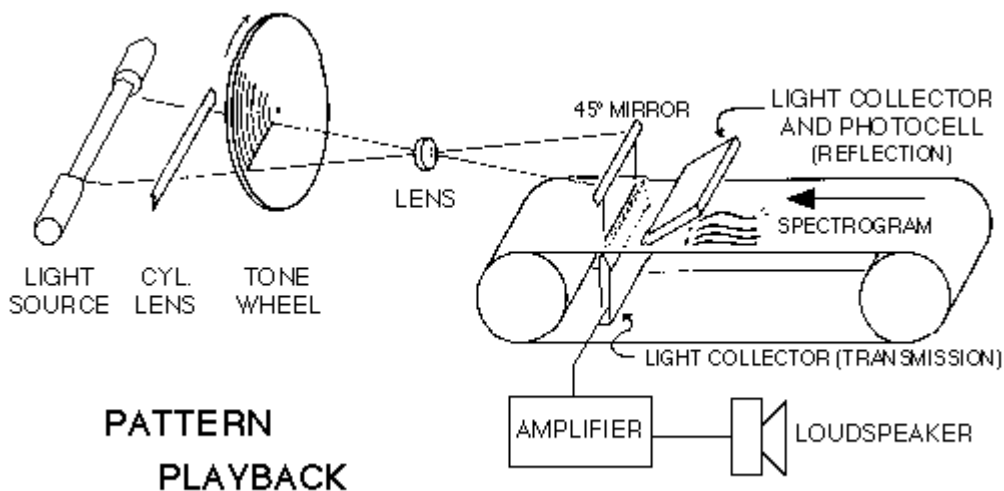


Abbildung 4. Pattern

Playback von Frank Cooper

Der ebenfalls in den 50er Jahren des 20. Jahrhunderts entwickelte *Ove* konnte bereits eine viel größere Bandbreite an verschiedenen Lauten und besaß eine Stimmmelodie, wie folgendes Beispiel zeigt: [AUDIO3 - Hörbeispiel Ove](#).

Ab den 1970er Jahren schließlich erfolgte die Synthese mittels Mikrocomputern. Heutzutage gibt es zwei große Methoden der Sprachsynthese: Erstens eine vollständige Synthese der Stimme seitens der Maschine (die *Physiologische Modellierung*) und Zweitens die *Signalmodellierung*, in welcher bereits im Vorfeld aufgenommene Sprachsamples geeignet kombiniert und moduliert werden. Die zweite Methode der Synthetisierung wird von dem meisten modernen Systemen präferiert.

Exkurs: Linguistik

Die Linguistik ist die Wissenschaft von der Sprache. Im Folgenden wird ein sehr kurzer Überblick über den Aufbau von Sprache gewährt - insbesondere von gesprochener Sprache - da das Verständnis auf diesem Gebiet essentiell für eine erfolgreiche moderne Sprachsynthese ist.

Gesprochene Sprache ist analog zu den Buchstaben der geschriebenen Sprache aus Einzelementen aufgebaut, welche sich zu Silben, Wörtern und schließlich Sätzen konkatenieren lassen. Diese hierarchische Gliederung von klein (Einzelement) nach groß (Satz) ist folgendermaßen aufgebaut:

Phoneme

Phoneme sind die kleinsten bedeutungsunterscheidenden, jedoch bedeutungstragenden Einheiten von Sprache. Sie entsprechen so den abstrahierten Lauten. Die Deutsche Sprache besitzt an die 40 Phoneme. Beispiele wären die Laute "p", "k", oder "t", nicht jedoch andere Laute wie "arrglll" oder "hmpfff", da diese nicht zu einem aussprechbaren bedeutungstragenden Wort konkateniert werden können.

Diphone (Lautübergänge)

Die Verbindung zweier Phoneme wird Diphon ("zweiklingend") genannt und wird meist von der Mitte des ersten Lauts zur Mitte des zweiten Lauts gemessen. Diphone bilden somit den Grundstein für die Konkatenation von Phonemen zu Silben und schließlich zu Wörtern. Je besser die Synthese zwischen den Phonemen, desto natürlicher klingt das Ergebnis.

Morpheme

Morpheme sind die kleinsten bedeutungstragenden Einheiten von Sprache, bilden also die kleinsten möglichen bedeutungstragenden Konkatinationen von Phonemen. Beispiel ist das Wort "Tatze", dessen Bedeutung sich durch Änderung jeweils nur eines Morphems ändert: "Tatze" -> "_K_atze" -> "Ka_ss_e" -> "_T_asse".

Silben

bekannt.

Wörter, Phrasen, Sätze

klar.

Technik moderner Sprachsynthese-Systeme

Die Umwandlung von Text zur Sprachausgabe bedarf dreier Schritte:

1. **Eingabe des Texts** (ggf. mit Metainformationen der [SSML](#))
2. **Synthese des Texts**
 1. Vorverarbeitung (i.e. *Textnormalisierung*)
 2. Synthese
3. **Ausgabe als Audio** (die sog. *waveform*)

die Vorverarbeitung der Eingabe durch Textnormalisierung

Bevor die eigentliche Sprachsynthese durchgeführt werden kann, wird der Eingabetext *normalisiert*. Insbesondere werden dabei Zahlen in eine **aussprechbare Form** gebracht (z.B. "12" zu "zwölf") und **Abkürzungen werden expandiert** (z.B. "d.h." zu "das heißt"). Weiterhin wird der Text einer ersten Analyse unterzogen und in die sog. **Inventarelemente** (eng. *token*) unterteilt. Diese Zerstückelung des Texts bewirkt, dass dieser nach Phonemen, Diphonen, Silben, Wörtern oder gar ganzen Sätzen unterteilt wird.

Zum Durchführen der Textnormalisierung zieht das Sprachsynthese-Programm Datenbanken heran. Das Abbilden von Originaltext hin zu Inventarelementen geschieht anhand bestimmter für jedes Synthese-System individuellen **Heuristiken und Entscheidungsbäumen**. Hierbei gilt: Je umfangreicher die **Datenbank**, desto besser kann der Text Abkürzungen expandieren und desto spezieller werden die Inventarelemente.

So muss beispielsweise ein in der Datenbank bereits vorhandener vollständiger Satz nicht erst vom Synthese-System mittels Algorithmen aus dem Originaltext erzeugt werden, sondern er kann direkt in einer bereits vorhandenen Datenform aus der Datenbank ausgelesen werden.

die Synthese

Im Anschluss an die Textnormalisierung erfolgt die eigentliche **Synthese** (engl. *rendering*). Hier werden die normalisierten Inventarelemente in Sprache **konvertiert**. Diese erste Form der Sprachausgabe ist jedoch noch ohne eine *menschliche Komponente*, sie würde wie eine typische Roboterstimme klingen. Deshalb wird der Sprachausgabe zusätzlich eine **Sprachmelodie** (*Prosodie*) hinzugefügt sowie eine **Stimmqualität** (engl. *voice*).

Die **Sprachmelodie** wird mittels Hinzufügen von auditiven Metainformationen erreicht, welche eine lebhaftere Aussprache des Textes bewirkt, aus der wir Menschen zusätzliche Informationen gewinnen. Merkmale der Sprachmelodie sind unter anderem das Absenken/Heben der Stimme bei Auftreten von Satzzeichen oder eine kurze Pause nach Punkten. Die ursprüngliche Aussprache wird

dazu in den Dimensionen *Dauer*, *Grundfrequenz*, *Variation der Tonhöhe* und *Amplitude* moduliert, wobei hier meist ein **PSOLA-Algorithmus** (*Pitch Synchronous Overlay Add*) zum Tragen kommt.

Die **Stimmqualität** bestimmt das Geschlecht, Alter, Sprechgeschwindigkeit und die Klangfarbe (singend, schreiend, flüsternd) des Sprechers.

SSML

Anwendungsgebiete

- Dialog-Steuerung von interaktiven Telefonie-Systemen
- Generierung von Hörbüchern (zum Glück sehr selten!)
- Generierung barrierefreier Benutzeroberflächen
- GPS-Navigationssysteme („in 100m links abbiegen“)

(Einschränkung: Meist sind zur Realisierung der Anwendungen noch weitere Sprache-verarbeitende Technologien erforderlich)

Herkunft

Die erste Version der [SSML](#) wurde im September 2004 als Recommendation der *W3C Voice Browser Working Group* verabschiedet und basiert auf der von Sun Microsystems entwickelten *JSML* und *JSpeech*.

Die [SSML](#) ist neben *VoiceXML 2.0* und der *Speech Recognition Grammar Specification* Teil des *W3C Speech Interface Framework*. Erst alle Elemente des Frameworks zusammen ermöglichen oben genannte Anwendungen. Die [SSML](#) ist dabei für die eigentliche Aussprache verantwortlich, wohingegen sich die anderen Komponenten der Logik von Sprachanwendungen ermöglichen (z.B. die Dialogsteuerung eines Telefonsystems). Das *Speech Interface Framework* ist die momentan aktivste Gruppe des [W3C?](#) und wird von vielen großen Unternehmen wie IBM, Apple, Microsoft, Sun, HP, France Telecom, Scansoft oder Voxpilot unterstützt.

die Elemente von SSML

Jedes SSML-Dokument ein XML Dokument. Es folgt eine Auflistung plus Beschreibung aller in der Recommendation des [W3C?](#) definierten Elemente der SSML 1.0.

Element speech

Ist das Wurzelement (root-Element) jeden SSML-Dokuments. Alles, was von diesem Element umschlossen ist, wird an das anschließende Sprachsynthese-System gesendet und wird von diesem interpretiert.

Attribut `xml:lang`

Setzt die Sprache, in welcher das Dokument ausgegeben werden soll. Beispiel für Attributwerte sind `xml:lang = de, en-US, fr, ...`

`xml:lang` kann auch in den Elementen `voice`, `p` und `s` gesetzt werden, wobei das globale `speech:xml:lang`-Attribut dann für das jeweilige innere Element überschrieben wird.

Attribut `xml:base` (optional)

`xml:base` setzt die Basis-URI. Nur die Elemente `lexicon` und `audio` verwenden dieses Attribut. Weitere Informationen über URIs, URLs und Pfade: [siehe Ausarbeitung zu XPATH](#)

Element `lexicon`

Mit diesem Element lassen sich (eine oder mehrere) zusätzliche spezielle Aussprache-Lexika (i.e. Datenbanken) in beliebigen Formaten angeben, beispielsweise für chemische oder medizinische Fachbegriffe. Der Pfad zu diesen Lexika wird entweder absolut oder relativ mit Hilfe des in `speech` definierten `=xml:path=-` Attributs angegeben.

Attribut `uri` (required)

Die Adresse (URI) zu der externen Datenbank

Attribut `type` (required)

Der MIME-Typ der Datenbank

Bsp:

```
<lexicon uri="sp.fu.de/lex4.lex" type="vnd.example.lexicon" />
```

Elemente `meta` und `metadata`

Die Elemente sind optional und geben zusätzliche Metainformationen z.B. über den Autor des [SSML](#)-Dokuments an. Ich werde mich nicht weiter zu diesen Attributen äußern, da sie für das Verständnis und die Anwendung von SSML nicht relevant sind.

Element `voice`

Legt fest, wie die synthetische Stimme (*voice quality*) klingen soll. Für die *voice quality* lassen sich jedoch nur sehr wenige Attribute festlegen. Insbesondere Geschlecht und Alter der synthetischen Stimme können gesetzt werden. Für weitere Modifikationen der Stimme: siehe Element `prosody`. Innerhalb eines SSML-Dokuments darf das `voice` Element mehrmals vorkommen, wobei jeweils der umschlossene Block eine andere Stimme erhält.

Attribut `name` (optional)

Anstatt umständlich einzelne Stimmattribute angeben zu müssen, kann das Sprachsynthese-System einzelnen Stimmen oder Stimmvarianten mnemonische Namen zuweisen, wie beispielsweise `Anna` oder `Abraham`.

Attribut `gender` (optional)

Legt das Geschlecht der synthetischen Stimme fest. Erlaubte Attributwerte sind `female`, `male` und `neutral`.

Attribut `age` (optional)

Setzt das ungefähre Alter des Sprechers. (Bsp. `age = 31`) Allerdings ist nicht jede Stimme in jedem Alter verfügbar. Die meisten Sprachsynthese-Systeme weisen einer Stimme immer nur ein oder

wenige fixe Alter zu, so dass im Falle nur eines Alters das setzen dieses Attributs keine Auswirkung hat. Ansonsten wird diejenige Stimme genommen, wessen Alter am nächsten am angegebenen Alter liegt.

Attribut `variant` (optional)

Wählt vom Sprachsynthese-System zulässige Varianten einer Stimme aus. Genauere Erklärungen bzw. Beispiele können hier mangels Information leider nicht gegeben werden.

Ebenfalls möglich ist es, innerhalb eines `voice` Elements die `xml:lang` neu zu setzen.

Element `prosody`

Prosody ist eines der komplexesten Elemente von [SSML](#) und ermöglicht eine Manipulation der synthetischen Stimme.

So können etwa Stimmhöhe und Sprechgeschwindigkeit verändert werden. Für die meisten Attribute sind sowohl absolute Angaben (z. B. 500Hz) als auch relative Attributwerte (z. B. +7%) möglich. Werden relative Angaben gemacht, so beziehen sich diese stets entweder auf die immer vorhandene Prosodie der Standardstimme (bzw. der vom Benutzer gesetzten `voice`) oder aber auf die soeben im `prosody` geänderten Attributwerte.

Attribut `pitch` (optional)

Verändert die Tonhöhe des Sprechers, i.e. die Basisfrequenz des Sprechers wovon ausgehend der Tonhöhenumfang moduliert wird. Mögliche Attributwerte sind 500Hz, +4%, `medium` oder `x-high`.

Attribut `range` (optional)

Stellt den Tonhöhenumfang des Sprechers bezüglich der Basisfrequenz (`pitch`) dar. Mögliche Attributwerte sind 500Hz, +4%, -20%.

Attribut `rate` (optional)

Die Sprechgeschwindigkeit. Z. B. `rate = slow, fast, 0.7` (also etwas langsamer), `2.3` (viel schneller), +34% oder 12%.

Attribut `duration` (optional)

Ein interessantes Attribut, in der man die Aussprache-Dauer des umschließenden Paragraphen bzw. des umschließenden Satzes in Sekunden/Millisekunden einstellen kann. Bsp: `duration = 20s` oder `400ms`.

Attribut `volume` (optional)

Ist die "Stimmgewaltigkeit"/Lautstärke. Beispielattribute `silent, soft, loud, medium, 0, +50%` oder `100`. Die ganzen Zahlen repräsentieren das Volumen von 0 (nix) bis 100 (seeehhr laut und kräftig).

Attribut `contour` (optional)

Dieses Attribut ist sehr kompliziert, so dass ich es nicht eingehend erläutert wird. Es wird eine Liste von Eigenschaften der Prosodie bezüglich einiger Referenzpunkte des

umschließenden Blocks angegeben. Die einzelnen Punkte werden interpoliert, so dass sich die Prosodie über den gesamten Block hinweg regelmäßig ändert. So ließe sich etwa ein immer dramatischer werdendes Gedicht in der Spannung steigern, indem am Anfang ein Referenzpunkt mit leiser `volume` gesetzt wird und am Ende die `volume` auf einen hohen Wert bei gleichzeitig langsam werdender Sprechweise gesetzt wird.

Bsp:

```
...
<prosody pitch="5200Hz" range="40Hz" volume="90" duration="2s">
  <s>
    Dies ist ein unglaublich kreischig hoher, sehr lauter und äußerst schnell
    ausgesprochener Satz!
  </s>
</prosody/>
...
```

Element s

Umschließt einen Satz. Beispiel siehe folgenden Absatz.

Element p

Umschließt einen Absatz bzw. Paragraphen. Gemeinsam mit dem `s` Element erlaubt das `p` Element eine semantische Gliederung des Texts.

Bsp:

```
...
<p>
  <s>
    Dieser Satz steht am Beginn des Texts.
  </s>
  <s>
    Dieser Satz steht wegen des umschließenden "p" im Zusammenhang mit dem
    obigen Satz.
  </s>
</p>
<s>
  Dieser Satz ist eigenständig.
</s>
...
```

Element break

Lässt eine Sprechpause bestehen, deren Länge durch zwei optionale Attribute definiert werden kann. Ist kein Attribut angegeben, so wird die Länge der Pause vom zu Grunde liegenden Sprachsynthese-System festgelegt.

Attribut `time` (optional)

Erzeugt eine Pause einer definierten, absoluten Länge. Erlaubte Werte sind z. B. `300ms` oder `2s`

Attribut `strength` (optional)

Erlaubte Attributwerte sind `none`, `weak`, `medium`, `strong` und `x-strong`. Diese geben die Stärke der Pause an, wobei `none` eine Pause der Länge `0ms` ist, also keine Pause.

Bei Angabe beider Elemente wird zuerst eine Pause der Länge `time` und anschließend eine Pause der Länge der Stärke `strength`.

Element `emphasis`

Dieses Element ist mit dem aus HTML bekannten Hervorhebungselementen `b` oder `i` vergleichbar. Mit `emphasis` umschlossene Stellen werden in der Sprachausgabe besonders betont. Das optionale Attribut `level` gibt den Grad der Betonungsstärke an. Ohne gegebenes Attribut wird der Grad `moderate` verwendet.

Attribut `level` (optional)

Erlaubte Attributwerte sind `strong`, `reduced`, `moderate` und `none`.

Element `sub`

`sub` steht für `substitute` und ersetzt eine gegebene Zeichenkette durch eine andere, welches insbesondere im Bezug auf zu expandierende Abkürzungen sinnvoll ist.

Attribut `alias` (required)

Enthält den Ausdruck, in den expandiert werden soll, also den Ausdruck der vom Synthese-System gesprochen wird.

Bsp:

```
...  
Eine nützliche Technologie ist das <sub alias="World Wide Web">WWW</sub>.  
...
```

Das im Text enthaltene "WWW" wird "World Wide Web" ausgesprochen.

In einer Bibliothek bereits enthaltene Abkürzungen werden automatisch expandiert. Das `sub` Element ist auch sinnvoll, um identisch geschriebenen Wörtern mit unterschiedlicher Aussprache (wie etwa "to read" im Englischen: "reed" oder "red") eine eindeutige Aussprache zu geben. In diesem Falle würde das `alias` Attribut die korrekte Aussprache enthalten (also "reed" oder "red"). Dieser Trick jedoch ist nicht hinreichend sauber! In Fällen der mehrdeutigen Aussprachen sollte auf das `phoneme` Element zurück gegriffen werden.

Element `say-as`

Das `say-as` Element stellt Regeln für die Aussprache auf, was insbesondere bei nicht eindeutigen Formaten wie Datums- und Zeitangaben sinnvoll ist. Das im Beginn dieses Artikels eingeführte Beispiel mit den Ausspracheproblemen zu " $=1/4=$ " kann mit Hilfe dieses Attributs gelöst werden.

Als Attribut (`interpret-as`) muss ein bestimmtes Format angegeben werden, welches Regeln zur Interpretation des Inhalts des `say-as` dient. Formate kann es beliebig viele geben, weshalb das [W3C?](#) hier auch keine Liste erlaubter Attributwerte für `interpret-as` angibt.

Attribut `interpret-as` (required)

Gibt den Inhaltstyp an, nach dem der Inhalt von `say-as` geparkt werden soll. Jedes Sprachsynthese-System bzw. jede zusätzliche Datenbank ("lexicon") kann eigene Inhaltstypen definieren.

Attribut `format` (optional)

Dient zur weiteren Verfeinerung/Unterscheidung von `interpret-as`.

Attribut `detail` (optional)

Mit diesem Attribut lässt sich der Detailgrad der Aussprache angeben. So ließe sich beispielsweise definieren, ob eine Datumsangabe lediglich "Erster Zehnter Zweitausendsechs" oder "Erster Oktober des Jahres Zweitausendundsechs" ausgesprochen werden soll.

Bsp:

```
...
<p>
  Verschiedene Aussprachemöglichkeiten von "1/4": <break/>
  als Bruch: <say-as interpret-as="fraction"> 1/4 </say-as> <break/>
  als amerikanisches Datum: <say-as xml:lang="en-US" interpret-as="date"> 1/4
</say-as> <break/>
  als deutsches Datum: <say-as xml:lang="de" interpret-as="date"> 1/4 </say-as>
<break/>
</p>
...
```

Achtung: Bei den Datumsangaben lässt sich mittels `xml:lang` bequem das gewünschte Format einstellen, allerdings wird der Text dann für die Dauer der Datumsangabe in der jeweiligen Sprache ausgesprochen.

Element `audio`

Erlaubt das Einfügen von Audio-Dateien. Bei Angabe eines nicht-leeren `audio` Elements wird der Inhalt gesprochen, wenn die angegebene Audiodatei nicht verfügbar ist.

Attribut `src` (required)

Der Pfad zur gewünschten Audiodatei.

Bsp:

```
Es wird nun ein Stück von Beethoven abgespielt:
<audio src="Sinfonie_9.mp3">
  Hoppla! Sie sollten an dieser Stelle eigentlich die <emphasis> Neunte Sinfonie
</emphasis> von Beethoven hören...
</audio>
```

Element `mark`

Setzt Labels als Sprungmarken (ähnlich einem [GoTo?](#)). Somit lassen sich Einsprungadressen für z. B. [VoiceXML?](#) definieren. Wird hier nicht weiter behandelt.

Beispiele

...folgen.

Fazit

[SSML](#) bietet in einigen Punkten (z.B. im `prosody` Element) nur sehr rudimentäre Kontrolle der

Sprachausgabe an, weshalb einige Systeme die SSML proprietär erweitert haben. Beispiel dafür ist das Emo-System von MARY des DFKI. Alles in Allem kann und soll SSML nicht darüber hinweg täuschen, dass ein semantisches Textverständnis seitens der Sprachsynthese-Systeme nicht gegeben ist und von der SSML auch nur sehr rudimentär hergestellt werden kann. Sinnvoll ist die SSML besonders im Bezug auf *device independency*, da ein bereits in XML vorliegender Text leicht um SSML-Elemente erweitert werden kann. Erst unter Benutzung aller Teile des *Speech Interface Frameworks* (SSML + VoiceXML + SRGS) kann die SSML ihren vollen Nutzen entfalten. Weiteres folgt.

Quellen

- [Sprachsynthese-Artikel auf de.wikipedia.org](http://de.wikipedia.org)
- [kurzer SSML-Artikel auf en.wikipedia.org](http://en.wikipedia.org)
- [Geschichte der Sprachsynthese](#)
- [interaktive online-Sprachsynthese mit guter Qualität](#)
- [direktes Testen von Mary über den Browser mittels Java-Interface \(direkte Eingabe von SSML möglich\)](#)
- [viele Beispiele moderner Sprachsynthese](#)
- [Recommendation des W3C \(vom September 2004\)](#)
- übersetzte Pressemitteilung des W3C zur Recommendation
- [O'Reilly „Speech Synthesis Markup Language: An Introduction“](#)

Audiovisuelle Quellen:

- Abbildung 1: <http://www.ling.su.se/staff/hartmut/bilder/kempln24.gif>
- Abbildung 2: <http://mambo.ucsc.edu/psl/smus/faber.gif>
- Abbildung 3: <http://mambo.ucsc.edu/psl/smus/voder2.gif>
- Abbildung 4: <http://www.ling.su.se/staff/hartmut/bilder/patplay.gif>
- Audio 1: [Hörbeispiel "Voder"](#)
- Audio 2: [Hörbeispiel "Pattern Playback"](#)
- Audio 3: [Hörbeispiel Ove](#)