

EXMARaLDA 1.0

Dokumentation

Thomas Schmidt, Juli 2001

Einleitung	3
A. EXMARaLDA 1.0	4
1. Allgemeine Architektur	4
2. Hauptkomponenten	7
2.1. Transkriptions-Kopf (<head>)	7
2.1.1. Meta-Information (<meta-information>)	7
2.1.2. Sprechertabelle (<speakertable>)	7
2.2. Transkriptions-Rümpfe	8
2.2.1. Segmentierte Transkription (<segmented-body>)	8
2.2.2. Basis-Transkription (<basic-body>)	8
2.2.3. Listen-Transkription (<list-body>)	9
2.2.4. Partitur-Tabellen-Transkription (<partitur-table-body>)	9
3. Hilfskomponenten	11
3.1. Formatierungs-Information (<tier-format-table>)	11
3.2. Konvertierungs-Informationen	11
3.2.1. Segmentierte Transkription nach Basis-Transkription (<segmented-to-basic-conversion-info>)	11
3.2.2. Segmentierte Transkription nach Listen-Transkription (<segmented-to-list-conversion-info>)	12
3.3. Segmentierungs-Information (<word-utterance-segmentation-info-table>)	12
4. Transkriptionskonventionen	13
4.1. Simple Exmaralda	13
4.2. CHAT Exmaralda	14
B. jexmaralda 1.0	15
1. Kommandozeilentools (jexmaralda/command)	15
1.1. Kommandozeilentools für die Konvertierung zwischen EXMARaLDA- und Ausgabe-Formaten	15
1.2. Kommandozeilentools für Hilfsformate	21
2. Basis-API (jexmaralda/jexmaralda)	23
3. GUI-Tools	24
Literatur	25
Anhang A: Beispiel-Daten	26
A0: Beispieldiskurs	26
a. als Partitur (Basis-Transkription → Partitur-Tabellen-Transkription → RTF-Partitur)	26
b. als Liste (Basis-Transkription → Partitur-Tabellen-Transkription → RTF-Liste)	26
A1: Kopf (<head>) einer Transkription:	27
A2: Rumpf (<basic-body>) einer Basis-Transkription	28
A3: Rumpf (<segmented-body>) einer segmentierten Transkription	29
A4: Rumpf (<list-body>) einer Listen-Transkription	30
A5: Rumpf (<partitur-table-body>) einer Partitur-Tabellen-Transkription (auf der Basis einer Basis-Transkription)	31
A6: Rumpf (<partitur-table-body>) einer Partitur-Tabellen-Transkription (auf der Basis einer Listen-Transkription)	32
A7: Formatierungs-Information (<tier-format-table>)	33
A8: Konvertierungs-Information segmentierte Transkription → Basis-Transkription (<segmented-to-basic-conversion-info>)	34
A9: Konvertierungs-Information segmentierte Transkription → Listen-Transkription (<segmented-to-list-conversion-info>)	34
A10: Segmentierungs-Information (<word-utterance-segmentation-info-table>)	35
Index	36

Einleitung

Die vorliegende Dokumentation beschreibt Aufbau und Funktionsweise der einzelnen Komponenten des EXMARaLDA-Systems. Sie ist vornehmlich an Adressaten gerichtet, die an der computerseitigen Umsetzung von EXMARaLDA interessiert sind. Für den theoretischen Hintergrund von EXMARaLDA sei auf Schmidt(2001) verwiesen.

Teil A beschreibt in Abschnitt 1 zunächst die allgemeine Architektur des Systems, d.h. die verschiedenen EXMARaLDA-Datenformate und wie sie zueinander in Beziehung stehen. Danach wird in den Abschnitten 2 und 3 die Struktur dieser Datenformate im Detail beschrieben. Abschnitt 4 schließlich beschreibt die Transkriptionskonventionen Simple Exmaralda und CHAT Exmaralda.

Teil B dokumentiert die unter dem Namen jexmaralda zusammengefassten JAVA-Tools zur Bearbeitung von EXMARaLDA-Daten. In der derzeitigen Version sind dies im Wesentlichen einfache Kommandozeilentools, mit deren Hilfe Konvertierungen zwischen EXMARaLDA-Datenformaten und Ausgabe-Formaten vorgenommen werden können. In künftigen Versionen werden jedoch weitere Tools mit graphischer Benutzer-Oberfläche hinzukommen, insbesondere ein Partitur-Transkriptions-Editor.

A. EXMARaLDA 1.0

1. Allgemeine Architektur

EXMARaLDA 1.0 besteht aus einer Reihe von XML-Document-Type-Definitions, die in ihrer Gesamtheit den Kern eines Systems zur Diskurstranskription und –annotation auf dem Computer bilden.

Die zentrale Komponente von EXMARaLDA 1.0 ist ein in der Document-Type-Definition „segmented-transcription.dtd“ definiertes XML-Format zur Kodierung von zeitlich und sprachlich segmentierten Diskurstranskriptionen – die **segmentierte Transkription**.

Weiterhin definiert EXMARaLDA 1.0 in „basic-transcription.dtd“ und „list-transcription.dtd“ zwei XML-Formate zur Kodierung bestimmter *Teilmengen einer segmentierten Transkription*, die nur die für die gängigen Eingabe- und Darstellungstypen von Diskurstranskriptionen (Partitur- und Spalteneingabe/-darstellung bzw. vertikale Eingabe/Darstellung, siehe Edwards (1995)) relevanten Informationen enthalten – die **Basis-Transkription** und die **Listen-Transkription**.

In „partitur-table-transcription.dtd“ wird die XML-Kodierung einer – gegebenenfalls auf eine gewisse Zeilenbreite umgebrochenen – *Partiturfächendarstellung einer Transkription* definiert, die als Grundlage zur Berechnung eines geeigneten Darstellungsformates (z.B. HTML oder RTF) dienen kann – eine **Partitur-Tabellen-Transkription**.

Die XML-Kodierung der zu einer *Konvertierung zwischen diesen vier Formaten* nötigen Information wird ebenfalls in Document-Type-Definitions definiert:

„segmented-to-list-conversion-info.dtd“ und „segmented-to-basic-conversion-info.dtd“ definieren die Kodierung derjenigen Teilmengen einer segmentierten Transkription, die bei einer Konvertierung zu einer Listen- bzw. Basis-Transkription beibehalten werden.

„tier-format-table.dtd“ definiert die Kodierung von Formatierungs-Informationen, die zur Berechnung einer Partitur-Tabellen-Transkription oder deren Umbruch auf eine bestimmte Zeilenbreite nötig sein können.

Schließlich wird in „word-utterance-segmentation-info.dtd“ definiert, wie Information zur automatischen Segmentierung einer aus einer Basis- oder Listen-Transkription gewonnenen segmentierten Transkription in Äußerungen und Wörter zu kodieren ist.

Die Informationen zur Konvertierung zwischen den Formaten und zur Segmentierung können in der Regel automatisch aus den betreffenden Daten generiert werden. Alle in Abbildung 1 durch Pfeile dargestellten Konvertierungen können daher in der Regel automatisch (d.h. ohne manuelle Spezifizierung der Konvertierungsinformation) vorgenommen werden.

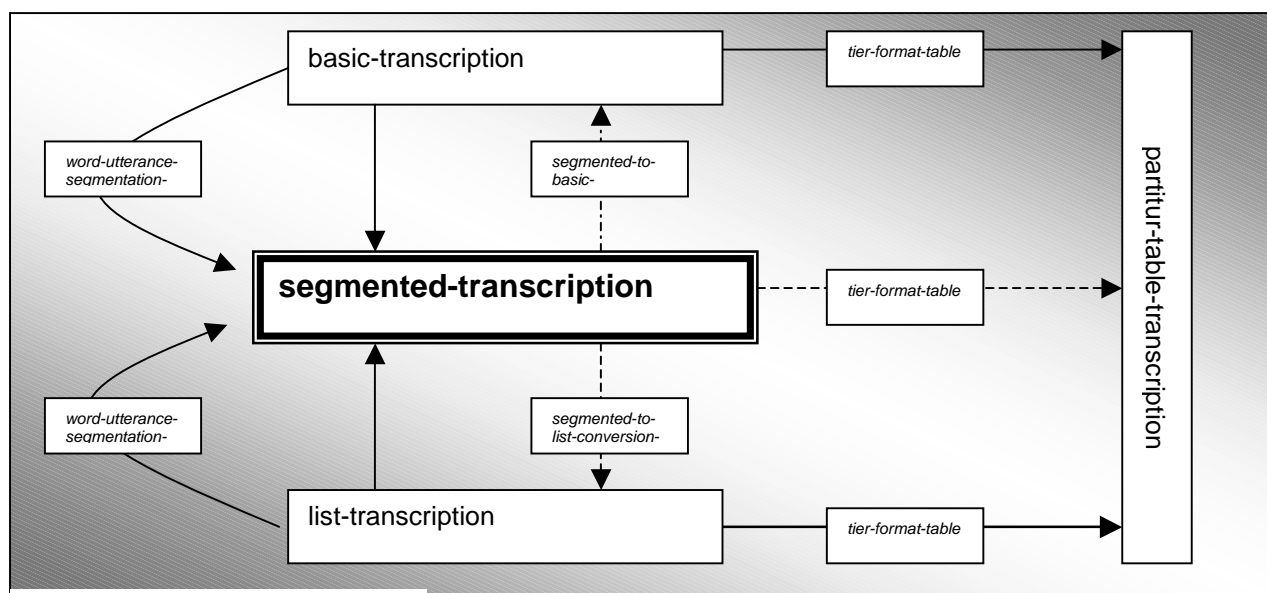


Abbildung 1: EXMARaLDA-Formate

Die segmentierte Transkription enthält die maximale Information zu einer Diskurstranskription. Liegt eine Diskurs-Transkription in diesem Format vor, können alle anderen Formate auf eine Rolle als temporäre Austauschformate reduziert werden, d.h. die segmentierte Transkription und nur diese ist als Basis einer Archivierung in Form von Korpora oder Datenbank vorgesehen.

Es ist weiterhin vorgesehen, daß nachträgliche Annotationen (d.h. Annotationen, die nicht während des ursprünglichen Transkriptionsprozesses vorgenommen werden, insbesondere sogenannte „Stand-Off-Annotations“) sich ebenfalls lediglich auf eine segmentierte Transkription beziehen. In der vorliegenden Version 1.0 spezifiziert EXMARaLDA allerdings weder, wie solche Annotationen in XML kodiert werden, noch wie sie zusammen mit der Transkription in ein darstellungsnäheres Format wie das der Basis-, Listen- oder Partitur-Tabellen-Transkription konvertiert werden können. Dies wird in einer späteren Version nachgeholt werden.

Alle gängigen Typen der Darstellung von Diskurstranskriptionen lassen sich aus den in einer Basis- bzw. Listentranskription kodierten Informationen herleiten. Die segmentierte Transkription enthält hingegen unter Umständen zu viele Informationen, um vollständig als Partitur-, Spalten- oder vertikale Transkription dargestellt werden zu können (siehe hierzu Schmidt(2001)).

Umgekehrt gilt auch, daß das Resultat des normalen Transkriptionsprozesses, der seinerseits fast immer an den gängigen Darstellungstypen orientiert ist (z.B. Partitur-Transkription mit syncWriter oder vertikale Transkription nach CHAT), nicht mehr Informationen enthält als sich in einer Basis- bzw. Listentranskription kodieren lassen.

Ein- und Ausgabe von Diskurstranskriptionen basieren daher auf den Teilmengen der Basistranskription (für Partitur und Spaltendarstellung) bzw. Listentranskription (für vertikale Darstellung).¹

Ein Import von oder Export in ein anderes Daten-Format kann sich ebenfalls auf die ihm ähnlichste Teilmenge beziehen – so können an der vertikalen Darstellung orientierte Formate wie CHAT, GAT, TEI oder die EXMARaLDA Formate Simple-Exmaralda und CHAT-Exmaralda in eine Listen-Transkription importiert, bzw. aus einer solchen exportiert werden, während eine Basis-Transkription der geeignete Ansatzpunkt für einen Import von an der Partiturdarstellung orientierten Daten wie HI-AT-DOS oder syncWriter ist.

¹ Es ist jedoch nicht ausgeschlossen, daß es praktikable Eingabe- und Darstellungsmethoden für segmentierte Transkriptionen gibt. Sollten solche gefunden werden, können sie in EXMARaLDA integriert werden – die beiden Teilmengenformate würden dadurch unter Umständen redundant.

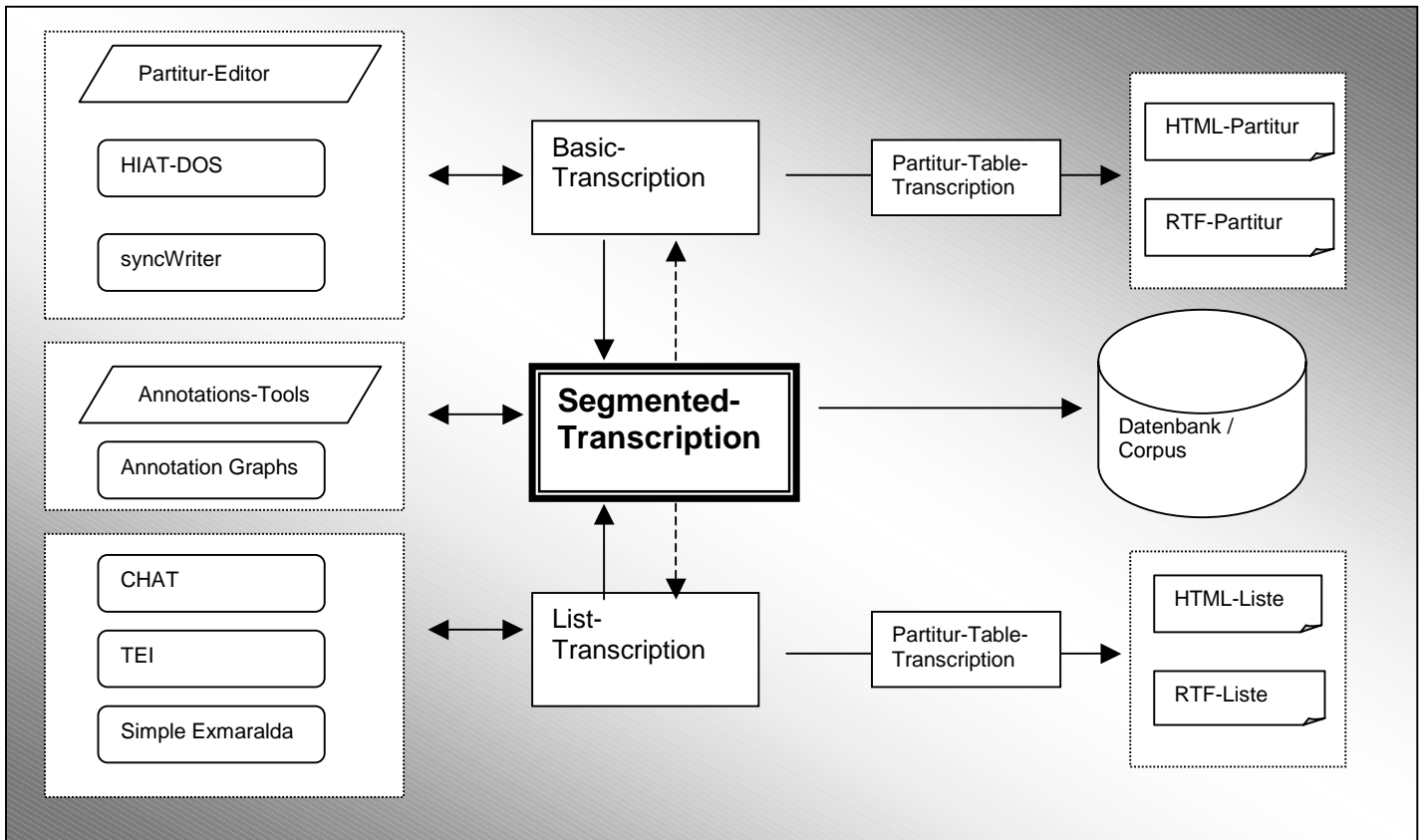


Abbildung 1: EXMARaLDA Ausgabe, Eingabe Import, Export

2. Hauptkomponenten

Die Hauptkomponenten von EXMARaLDA sind die vier Dokumenttypen „segmented-transcription“, „basic-transcription“, „list-transcription“ und „partitur-table-transcription“. Sie bestehen jeweils aus einem Kopf (head), der für alle vier Typen identisch ist, und einem Rumpf (body), der sich je nach Typ unterscheidet.

2.1. Transkriptions-Kopf (<head>)

Im Kopf einer Transkription werden einer zum einen Meta-Informationen zum Diskurs kodiert, zum anderen beinhaltet er die Sprechertabelle.

2.1.1. Meta-Information (<meta-information>)

Die DTDs schreiben lediglich fünf festgelegte Attribute für die Meta-Informationen vor. Im einzelnen sind dies:

<project-name> :	der Name des Projektes, aus dem die Transkription stammt
<transcription-name> :	der Name, unter dem die Transkription in ihrem Projekt funktioniert
<referenced-file>:	die (Video- oder Audio-)Datei, auf die sich die Transkription bezieht
<comment>:	ein freier (d.h. unstrukturierter) Text-Kommentar zur Transkription
<transcription-convention>:	die bei der Transkription benutzte Transkriptions-Konvention

Die Werte dieser Attribute können leergelassen werden, sie stellen aber unserer Einschätzung nach die minimalen Informationen dar, die zu einer sinnvollen Archivierung der Daten, z.B. in einer Datenbank, benötigt werden.

Erfahrungsgemäß werden je nach Projektzusammenhang und Verwendungsart der Daten weitere Meta-Informationen zur Diskurstranskription benötigt. Da diese aber schwer bis gar nicht vorhersehbar sind (siehe jedoch Wittenburg et al. (2000)), sehen die DTDs eine Möglichkeit zum Hinzufügen benutzer-definierter Informationen (<ud-meta-information>) vor, die in Form von Attribut-Wert-Paaren (<ud-information>) kodiert werden

2.1.2. Sprechertabelle (<speakertable>)

Die Sprechertabelle besteht aus einer Anzahl von Sprechern (<speaker>), die jeweils mit einer eindeutigen ID versehen werden. Zu jedem Sprecher sind folgende festgelegte Attribute vorgesehen:

<abbreviation> :	das Sprecherkürzel (, das i.d.R. bei der Darstellung verwendet wird)
<sex> :	das Geschlecht des Sprechers - vorgesehene Werte sind ‚m‘ (male), ‚f‘ (female) und ‚u‘ (unknown)
<comment> :	ein freier Text-Kommentar zum Sprecher

Sprachinformationen zum Sprecher werden unter

<languages-used>:	die im transkribierten Diskurs von diesem Sprecher benutzten Sprache(n)
<l1>:	die Erstsprache(n) des Sprechers
<l2>:	die Zweitsprache(n) des Sprechers

als eine Reihe von Attribut-Wert-Paaren (<language>) kodiert, wobei die Sprache in einem XML-Attribut des standardisierten Typs xml:lang vermerkt wird. Mindestens ein Eintrag unter <languages-used> ist erforderlich (dies scheint wiederum vom Standpunkt einer sinnvollen Archivierung aus notwendig), die anderen beiden Sprachinformationen (<l1>, <l2>) können ggf. auch leer bleiben.

Mehr noch als bei den Meta-Informationen sind auch bei den Sprecherinformationen je nach Projektzusammenhang und Verwendungsart der Daten eine Vielzahl weiterer Informationen (z.B. Alter, voller

Name des Sprechers etc.) notwendig, diese können wiederum als Attribut-Wert-Paare <ud-information> unter <ud-speaker-information> kodiert werden.

2.2. Transkriptions-Rümpfe

2.2.1. Segmentierte Transkription (<segmented-body>)

Der Rumpf einer segmentierten Transkription besteht aus

- der (ungeordneten) Menge <timepoints> aller Zeitpunkte (<timepoint>), die in der gemeinsamen Zeitachse und den individuellen Zeitachsen der einzelnen Spuren verwendet werden. Jeder Zeitpunkt wird mit einer eindeutigen ID versehen und kann optional Informationen zur absoluten Zeit unter dem Attribute ‚absolute-time‘ enthalten
- der Zeitachse <common-timeline>, die Verweise (<tpr>) auf all diejenigen Zeitpunkte in <timepoints> enthält, die in allen individuellen Zeitachsen der einzelnen Spuren verwendet werden. Der Verweis wird über IDREFs auf die IDs der Zeitpunkte realisiert, und die Ordnung der Verweise entspricht der relativen zeitlichen Ordnung der Punkte auf der gemeinsamen Zeitachse.
- einer Anzahl von segmentierten Spuren (<segmented-tier>)

Jede segmentierte Spur muß mit einer eindeutigen ID versehen werden, außerdem muß ihr Sprecher (als IDREF auf eine ID in der Sprechertabelle), ihre Kategorie (als frei wählbarer Wert) und ihr Typ (einer der Werte ‚t‘, ‚d‘, ‚a‘, ‚l‘, ‚ud‘ – siehe hierzu Schmidt(2001)) spezifiziert werden. Unter <timeline> wird die individuelle Zeitachse der jeweiligen Spur kodiert, wiederum als eine geordnete Menge von Verweisen (<tpr>) auf die Zeitpunkte.

Die Spur beinhaltet ihrerseits eine oder mehrere Segmentierungen (<segmentation>). Das identifizierende Element einer Segmentierung ist ihr Name, der unter einem entsprechenden Attribut ‚name‘ spezifiziert wird (z.B.: ‚event‘, ‚utterance‘ etc.).

Jede Segmentierung besteht ihrerseits wieder entweder aus einer Menge von Segmenten (<segment>) oder aus einer Menge von Dateiverweisen (<link> - bei Spuren des Typs ‚l‘). Diese Elemente besitzen jeweils Attribute ‚start‘ und ‚end‘, die auf Punkte der Zeitachse verweisen. Segmente werden zusätzlich mit einer ID versehen (insbesondere, um eine „Stand-off“-Annotation zu ermöglichen), Dateiverweise besitzen Attribute ‚medium‘ und ‚url‘, die die Art und den Ort der referenzierten Datei spezifizieren. Inhalt eines <Segment> oder <Link>-Elementes ist der Text, der das jeweilige Element beschreibt. Die Ordnung der Dateiverweise und Segmente muß sich nicht an der Zeitachse der jeweiligen Spur orientieren (diese Information ist implizit in den Start- und End-Attributen enthalten), sollte dies nach Möglichkeit aber tun, um die Datei besser lesbar zu machen.

2.2.2. Basis-Transkription (<basic-body>)

Der Rumpf einer Basis-Transkription besteht aus

- der gemeinsamen Zeitachse <common-timeline>, die ihrerseits alle Zeitpunkte <tli> enthält, auf die von den Ereignissen und Dateiverweisen in den einzelnen Spuren verwiesen wird. Jeder Zeitpunkt enthält daher eine obligatorische ID. Zusätzlich kann in einem optionalen Attribut ‚absolute-time‘ ein absoluter Zeitwert spezifiziert werden. Die Ordnung der <tli>-Elemente entspricht der realtiven Ordnung der Zeitpunkte.
- einer Anzahl einfacher Spuren (<tier>)

Jede einfache Spur muß mit einer eindeutigen ID versehen werden, außerdem muß ihr Sprecher (als IDREF auf eine ID in der Sprechertabelle), ihre Kategorie (als frei wählbarer Wert) und ihr Typ (einer der Werte ‚t‘, ‚d‘, ‚a‘, ‚l‘, ‚ud‘ – siehe hierzu Schmidt(2001)) spezifiziert werden.

Jede einfache Spur beinhaltet entweder eine Menge von Ereignissen (<event>) oder eine Menge von Dateiverweisen (<link> - bei Spuren des Typs ‚l‘). Diese Elemente besitzen jeweils Attribute ‚start‘ und ‚end‘, die auf Punkte der Zeitachse verweisen. Dateiverweise besitzen zusätzliche Attribute ‚medium‘ und ‚url‘, die die Art und den Ort der referenzierten Datei spezifizieren. Inhalt eines <Event> oder <Link>-Elementes ist der Text, der das jeweilige Element beschreibt. Die Ordnung der Dateiverweise und Ereignisse muß sich nicht an der gemeinsamen Zeitachse der Basis-Transkription orientieren (diese Information ist implizit in den Start- und End-Attributen enthalten), sollte dies nach Möglichkeit aber tun, um die Datei besser lesbar zu machen.

2.2.3. Listen-Transkription (<list-body>)

Der Rumpf einer Listen-Transkription besteht aus

- der gemeinsamen Zeitachse <common-timeline>, die ihrerseits alle Zeitpunkte <tli> enthält, auf die von den Ereignissen und Dateiverweisen in den einzelnen Listen-Spuren der Listen-Elemente verwiesen wird. Jeder Zeitpunkt enthält daher eine obligatorische ID. Zusätzlich kann in einem optionalen Attribut ‚absolute-time‘ ein absoluter Zeitwert spezifiziert werden. Die Ordnung der <tli>-Elemente entspricht der realtiven Ordnung der Zeitpunkte.
- eine Menge (<tier-references>) von Spur-Referenzen (<tier-reference>). Die einzelnen Spur-Referenzen enthalten Informationen, die zum Zusammenfassen einzelner Listen-Spuren zu segmentierten oder einfachen Spuren (d.h. zur Konvertierung der list-transcription in eine segmented-transcription oder basic-transcription) notwendig sind. Die Listen-Spuren beziehen sich auf diese Spur-Referenzen mittels einer IDREF auf das ID-Attribut ‚id‘, die übrigen Attribute ‚speaker‘, ‚category‘ und ‚type‘ beinhalten die benötigten Informationen
- einer Menge von Listen-Elementen (<list-item>)

Jedes Listen-Element enthält Attribute ‚start‘ und ‚end‘, die auf die gemeinsame Zeitachse verweisen und angeben, welche Zeitspanne die in seinen Listen-Spuren enthaltenen Ereignisse und Dateiverweise umfassen. Weiterhin wird über das Attribut ‚speaker‘ eine Referenz auf die Sprechertabelle vorgenommen; das Attribut ‚name‘ gibt an, welche Einheit der Listen-Transkription zugrunde gelegt wird (siehe Schmidt(2001)).

Ein Listen-Element besteht seinerseits aus mehreren Listen-Spuren (<list-tier>). Diese enthalten im Attribut ‚tierref‘ eine Referenz auf die oben beschriebenen Spur-Referenzen und im Attribut ‚level‘ die Information, ob es sich um die Haupt-Spur („main“) oder eine abhängige („dependent“), Annotations- („annotation“) oder Dateiverweisspur („links“). Davon abgesehen ist eine Listen-Spur aufgebaut wie die oben beschriebene einfache Spur, d.h. sie besteht aus einer (nicht zwingend geordneten) Menge von Ereignissen oder Dateiverweisen.

2.2.4. Partitur-Tabellen-Transkription (<partitur-table-body>)

Die Partitur-Tabellen-Transkription fungiert als Schnittstelle zwischen den inhaltsorientierten Formaten basic-transcription und list-transcription und darstellungsorientierten Formaten wie z.B. HTML und RTF. Bei einer Konvertierung in dieses Format werden die abstrakt zeitorientierten Informationen der Ausgangsformate in abstrakt raumorientierte Informationen, nämlich vertikale und horizontale Position in einer Tabelle, Zellenbreiten der Tabelle etc., umgewandelt. Aus diesen Informationen können dann einfache und auf flexible Art und Weise konkrete Darstellungen berechnet werden. Umgekehrt können aus einer solchen darstellungsorientierten Repräsentation allerdings nur schwer die inhaltsbezogenen Informationen rekonstruiert werden. Die Konvertierung zwischen Listen- bzw. Basis-Transkription und Partitur-Tabellen-Transkription wird daher in der Regel nur in einer Richtung erfolgen.

Gewisse eine solche Tabelle beschreibende Werte (wie. z.B. Zellenbreite) sind nur im Zusammenhang mit konkreten Informationen zu den Font-Typen, -Größen etc., auf deren Grundlage sie berechnet wurden, sinnvoll. Der Rumpf einer Partitur-Tabellen-Transkription besteht daher aus:

- der gemeinsamen Zeitachse (<common-timeline>). Diese wird aus den inhaltsorientierten Formaten einfach übernommen. Die meisten konkreten Darstellung werden ohne die in ihr kodierte Information auskommen, da diese bereits in der Anordnung der Tabellenzellen reflektiert ist.
- einer Formatierungstabelle (<tier-format-table>). Dieses Element ist optional. Es kann die oben angesprochenen Informationen zu den Formatierungen beinhalten, auf deren Grundlage die Partitur-Tabelle(n) berechnet wurden. Die interne Syntax einer Formatierungstabelle ist weiter unten beschrieben.
- einer oder mehreren Partitur-Tabellen (<partitur-table>)

Jede Partitur-Tabelle besteht aus einer Anzahl von Partitur-Zeilen (<partitur-row>), von denen die erste zusätzlich als Kopfzeile (<head-row>) ausgezeichnet ist. Jede Partitur-Zeile enthält ein Attribut ‚id‘, das z.B. zur Identifizierung der für sie vorgesehenen Formatierung dienen kann. Weiterhin besteht eine Partitur-Zeile aus einer Anzahl von Partitur-Zellen (<partitur-cell>), von denen wiederum die erste zusätzlich ausgezeichnet ist, und zwar als Zeilen-Beschriftung (<row-label>).

Die einzelnen Partitur-Zellen enthalten neben Text mehrere für eine konkrete Darstellung u.U. relevante Informationen in Form von Attributen. Im einzelnen sind dies:

id :	wie bei den Partitur-Zeilen kann dieses Attribut für die Identifizierung einer für diese Zelle vorgesehenen Formatierung dienen
anchor:	in diesem Attribut kann Information vermerkt werden, wenn die zugehörige Zelle als Verweisziel dienen soll. Beispielsweise können bei einer Darstellung in HTML die Punkte der Zeitachse (die sich in der Kopfzeile einer Partitur-Tabelle wiederfinden) als Hypertext-Zielpunkte zur Navigierung innerhalb der Transkription dienen.
hyperlink:	dieses Attribut ist vorgesehen, um die Information zu kodieren, die benötigt wird, wenn die zugehörige Zelle die Quelle eines Verweises sein soll. Dies ist vor allem bei der Darstellung von Dateiverweisen der Fall.
column:	dieses Attribut gibt an, zu welcher Spalten der Tabelle die zugehörige Zelle gehört (für Zeilen-Beschriftungen wird standardmäßig die Zeilennummer -1 vorgeschlagen)
width:	dieses Attribut gibt an, welche Breite für die zugehörige Zelle vorgesehen ist. Die Maßeinheit wird dabei nicht von der DTD festgelegt, es ist aber wahrscheinlich, dass die Werte in Pixeln angegeben werden.
span:	in diesem Attribut wird festgelegt, wie viele Spalten, die zugehörige Zelle umfasst.

Zu keinem dieser Attribute muß ein Wert angegeben werden, und die DTD überprüft auch nicht, ob die angegebenen Werte untereinander konsistent sind (z.B. ob alle Zeilen dieselbe Anzahl von Spalten umfassen, ob sich die Zellenbreiten in verschiedenen Zeilen zum selben Wert addieren oder ob Spaltenzahlen aufsteigend sind). Für eine einfache HTML-Darstellung (d.h. ohne explizite Formatierung) ist z.B. die in den Attributen kodierbare Information wahrscheinlich gar nicht nötig, wohingegen z.B. für eine angemessene RTF-Darstellung zumindest die Attribute ‚id‘ (zur Identifizierung einer Formatierung) und ‚width‘ (zur korrekten Positionierung des Zelleninhaltes) mit sinnvollen Werten belegt sein sollten.

3. Hilfskomponenten

3.1. Formatierungs-Information (<tier-format-table>)

Eine Formatierungs-Information besteht aus

- einer optionalen Liste von referenzierten Dateien (<referenced-file>), in deren Attribut ‚url‘ Verweise auf diejenigen Dateien vermerkt werden können, auf die sich die Formatierungen beziehen. Die Formatierungstabelle innerhalb einer Partitur-Tabellen-Transkription darf keine solchen Verweise enthalten da sie ja bereits in die referenzierte Datei eingebettet ist.
- einer Liste von Formatierungs-Informationen (<tier-format>)

Der Text-Inhalt eines <tier-format>-Elementes legt den Namen der Schriftart fest. Weitere Informationen werden in den Attributen kodiert:

tierref:	ist ein ID-Attribut, das auf das zu formatierende Element verweist. Dieses wird i.d.R. eine einfache Spur einer Basis-Transkription, eine Listen-Spur einer Listen-Transkription oder eine Partitur-Zeile oder –Zelle einer Partitur-Tabellen-Transkription sein – dies ist jedoch nicht zwingend (z.B. kann auch zusätzlich eine Formatierung für leere Partitur-Zellen oder Partitur-Zeilen-Beschriftungen festgelegt werden).
style-name:	dieses Attribut legt den Schriftschnitt fest. Mögliche Werte sind ‚plain‘ (nicht fett, nicht kursiv), ‚italic‘ (kursiv) und ‚bold‘ (fett)
size:	dieses Attribut legt die Schriftgröße fest. Um eine möglichst strenge Syntax-Kontrolle zu ermöglichen, lässt die DTD nur ganzzahlige Werte zwischen 1 und 72 zu – größere Schriftgrößen oder nicht-ganzzahlige Werte sollten aber in der Praxis auch nicht notwendig sein.
alignment-name:	dieses Attribut legt die Ausrichtung fest. Mögliche Werte sind ‚left‘ (linksbündig), ‚right‘ (rechtsbündig) und ‚center‘ (zentriert)
textcolor-name:	dieses Attribut legt die Schriftfarbe fest. Die möglichen Werte orientieren sich an den Werten für HTML-Color-Tags, d.h. es sind die folgenden sechzehn Werte vorgesehen: white, lightGray, darkGray, black, red, pink, orange, yellow, green, magenta, cyan, blue
bgcolor-name:	dieses Attribut legt die Hintergrundfarbe fest. Es sind dieselben möglichen Werte vorgesehen wie beim Attribut ‚textcolor-name‘.

3.2. Konvertierungs-Informationen

3.2.1. Segmentierte Transkription nach Basis-Transkription (<segmented-to-basic-conversion-info>)

Da eine segmentierte Spur i.d.R. mehrere Segmentierungen enthält, von denen meist nur eine bei der Konvertierung in eine Basis-Transkription berücksichtigt werden soll, und weil weiterhin nicht alle Segmentierungen für eine Konvertierung in eine Basis-Transkription in Frage kommen (z.B. wenn sich die in ihnen enthaltenen Segmente nicht sinnvoll in eine allen Spuren gemeinsame Zeitachse einordnen lassen), definiert EXMARaLDA ein Format, in dem die Informationen, die für diese Konvertierung benötigt werden, kodiert werden können: Ein Dokument des Typs <segmented-to-basic-conversion-info> besteht aus

- einem Verweis auf die zu konvertierende segmentierte Transkription (<referenced-file>). Das Attribut ‚url‘, das diese Datei spezifiziert, kann jedoch auch leer bleiben.
- einer Liste von Konvertierungsinformationen (<info>), die in ihren Attributen festlegen, welche Segmentation (Attribut ‚segmentation‘) welcher segmentierten Spur (Attribut ‚tierref‘) bei der Konvertierung berücksichtigt werden soll. Normalerweise wird pro Spur nur eine Segmentierung und damit nur eine Konvertierungsinformation vermerkt werden, die DTD schließt aber nicht aus, dass mehrere Segmentierungen ein und derselben Spur für die Übernahme in eine Basis-Transkription vorgesehen werden.

3.2.2. Segmentierte Transkription nach Listen-Transkription (<segmented-to-list-conversion-info>)

Bei der Konvertierung einer segmentierten Transkription in eine Listen-Transkription müssen zunächst dieselben Informationen wie bei der Konvertierung in eine Basis-Transkription zur Verfügung stehen, d.h. welche Segmentierung aus welcher segmentierten Spur berücksichtigt werden soll. Ein Dokument des Typs <segmented-to-list-conversion-info> enthält deshalb zunächst dieselben Informationen wie ein Dokument des Typs <segmented-to-basic-conversion-info>. Zusätzlich muß aber erstens auch spezifiziert werden, welche Segmentierung als Haupteinheit der Listen-Transkription vorgesehen ist (i.d.R. wird dies die Äußerung oder der Turn sein – siehe hierzu Schmidt (2001)). Dies geschieht über das Attribut ‚main-segmentation‘ im Top-Level-Element. Zweitens muß angegeben werden, in welcher segmentierten Spur sich diese Haupteinheit für jeden Sprecher befindet. Dies geschieht über eine Liste von Elementen <main-tier>, die die besagten Informationen in den Attributen ‚speaker‘ und ‚tierID‘ vermerken. Dabei muß nicht notwendigerweise für jeden Sprecher der segmentierten Transkription eine solche <main-tier>-Information angegeben werden (z.B. ist es denkbar, dass es für gewisse Sprecher keine Äußerungs- oder Turn-Segmentierung gibt).

3.3. Segmentierungs-Information (<word-utterance-segmentation-info-table>)

Die automatische Segmentierung in Turns, Äußerungen und Wörter geschieht auf der Basis der Ereignisse in Basis- und Listen-Transkription, bzw. auf der Basis von Segmenten einer Segmentierung mit dem Namen „event“ in einer segmentierten Transkription. Da jedoch nicht alle Ereignisse für eine solche Segmentierung geeignet sind (i.d.R. nur Ereignisse des Typs ‚t‘ – siehe Schmidt(2001)), und die Segmentierung u.U. für verschiedene Spuren auf unterschiedliche Art und Weise (d.h. nach verschiedenen Äußerungs- und Wort-Endsymbolen) erfolgen soll, muß festgelegt werden, welche Spuren wie segmentiert werden sollen. Diese Information kann in einem Dokument des Typs <word-utterance-segmentation-info-table> kodiert werden. Ein solches besteht aus:

- einem Verweis auf die zu segmentierende Transkription (<referenced-file>). Das Attribut ‚url‘, das diese Datei spezifiziert, kann jedoch auch leer bleiben.
- einer Liste von Segmentierungsinformationen (<word-utterance-segmentation-info>).

Letztere legen zunächst im Attribut ‚tierref‘ fest, auf welche (segmentierte, einfache oder Listen-)Spur sie sich beziehen, d.h. welche Spuren nach Turns, Äußerungen und/oder Wörtern segmentiert werden sollen. Weiterhin enthalten sie eine Liste von Äußerungs- (<utterance-end-symbol>) und Wort-Endsymbolen (<word-end-symbol>).

Wenn nur nach einer oder zwei der Kategorien ‚Turn‘, ‚Äußerung‘ und ‚Wort‘ segmentiert werden soll, kann die dann nicht benötigte Information (z.B. Wort-Endsymbole) einfach ausgelassen werden.

4. Transkriptionskonventionen

Das EXMARaLDA-System selbst versteht sich als theorieneutral und macht deshalb auch bezüglich Transkriptionskonventionen, d.h. der Art und Weise wie sprachliche und andere für einen Diskurs relevante Handlungen verschriftlicht werden, nur minimale Annahmen (siehe Schmidt(2001)), die sich in erster Linie aus der technischen Realisierung des Systems ergeben.

Bei den Konventionen für Simple Exmaralda und CHAT Exmaralda handelt es sich daher ebenfalls um einige rein technische Festlegungen, bei deren Einhaltung es möglich ist, EXMARaLDA-Daten aus herkömmlichen Text-Dateien, wie sie beispielsweise in einem Text-Editor oder Textverarbeitungsprogramm erstellt werden, zu importieren. Gemäß den in Abschnitt 1 dargestellten Überlegungen ist das Zielformat eines solchen Importes zunächst eine Listen-Transkription, denn diese ist der in einer Textdatei praktizierten vertikalen Darstellungsweise am ähnlichsten. Die Konventionen legen also in keiner Weise fest, was und wie Gesprochenes und Getanes verschriftlicht wird, sondern lediglich, wie das Verschriftlichte typographisch zu organisieren ist, damit es sinnvoll in eine EXMARaLDA-Transkription zu überführen ist.

4.1. Simple Exmaralda

Simple Exmaralda erlaubt es, sehr einfache Transkriptionen im vertikalen Format als Textdatei zu erstellen, die dann in eine EXMARaLDA-Listen-Trankription importiert (und dann möglicherweise weiterbearbeitet) werden kann. Folgende Konventionen müssen dabei eingehalten werden:

1. Jede Zeile beginnt mit der Sigle des Sprechers der Äußerung, gefolgt von einem Doppelpunkt. Zwei Sprecher dürfen sich nicht dieselbe Sigle teilen, und die Groß- und Kleinschreibung in den Sprechersiglen ist relevant (d.h. z.B. ‚Tom‘ und ‚TOM‘ werden als zwei verschiedene Sprechersiglen behandelt):

```
TOM: .....
TIM: .....
```

2. Pro Zeile wird eine Äußerung transkribiert. Jede Zeile wird mit einem Zeilenendezeichen (carriage return) beendet.

```
TOM: Hallo, Tim!
TIM: Hallo, Tom.
```

3. Eine etwaige Transkription von non-verbale Handlungen, die die Äußerung begleiten (d.h. parallel zu ihr stattfinden), kann vor der Äußerung in eckigen Klammern vorangestellt werden.

```
TOM: [winkt] Hallo, Tim!
TIM: [winkt] Hallo, Tom.
```

4. Eine etwaige Annotation der Äußerung (z.B. eine Übersetzung) kann der Äußerung in geschweiften Klammern nachgestellt werden. Dies geschieht auf der gleichen Zeile, auf der auch die zugehörige Äußerung steht.

```
TOM: [winkt] Hallo, Tim! {Salut, Tim!}
TIM: [winkt] Hallo, Tom. {Salut, Tom!}
```

5. Sich überlappende Äußerungsteile verschiedener Sprecher werden mit spitzen Klammern eingefasst. Der schließenden spitzen Klammer folgt eine beliebige Zeichenkette, die die Überlappung indiziert, und eine weitere schließende spitze Klammer. Die Indizierung sollte aus Gründen der Lesbarkeit durch Zahlen erfolgen, diese müssen jedoch nicht aufsteigend geordnet sein (sie müssen noch nicht einmal Zahlen sein, notwendig ist nur, dass sie eindeutig sind). Für eine bessere Lesbarkeit, können die sich überlappenden Äußerungsteile mittels Leerzeichen oder Tabulatoren eingerückt werden.

```
TOM: [winkt] Hallo, <Tim!>1> {Salut, Tim!}
TIM: [winkt] <Hallo>1>, Tom. {Salut, Tom!}
```

6. Eckige, geschweifte und spitze Klammern dürfen nur in der oben definierten Art und Weise verwendet werden und sollten sonst in der Transkription nicht vorkommen.

Simple Exmaralda sieht keine Möglichkeit vor, Meta-Informationen zum Diskurs oder den Sprechern zu kodieren, dies kann aber selbstverständlich nach dem Export in eine Listen-Transkription nachgeholt werden (da die DTDs vorsehen, dass für jeden Sprecher mindestens eine benutzte Sprache angegeben ist, wird diese default-mäßig auf ‚de‘ für Deutsch gesetzt).

4.2. CHAT Exmaralda

CHAT Exmaralda wird eine Einschränkung der CHAT-Konventionen (MacWhinney (1995)) sein. Nach diesen Konventionen erstellte Daten werden in eine EXMARaLDA-Listen-Transkription importiert werden können, d.h. CHAT Exmaralda-Transkriptionen werden sowohl mit den CLAN-Tools der CHILDES-Datenbank als auch mit den EXMARaLDA-Tools bearbeitbar sein.

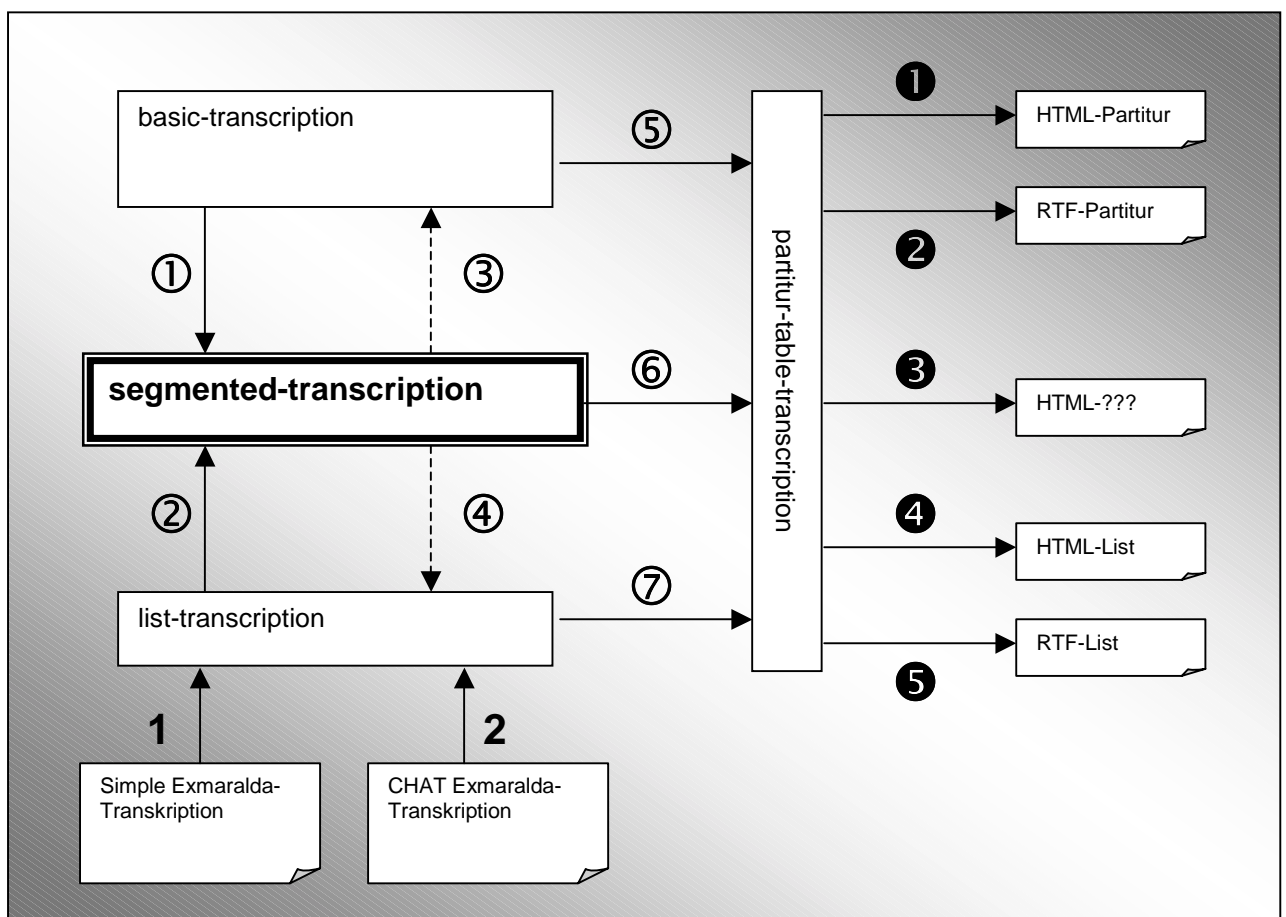
B. jexmaralda 1.0

jexmaralda 1.0. ist eine Sammlung von Java-Tools zur Eingabe, Bearbeitung und Ausgabe von EXMARaLDA-Daten. Alle Tools werden in der Java-Version 1.3. implementiert und sind damit unter den Betriebssystemen Windows 98 / Windows ME / Windows XP / Windows NT 4.x / Windows 2000, MAC OS X, Unix und Linux lauffähig, nicht jedoch unter MAC OS 9.x oder früheren Macintosh-Betriebssystemen. Außer den Standard-JRE-Packages (Version 1.3.) benutzt jexmaralda 1.0 das Sun-API für XML (JAXP 1.0). Um jexmaralda-Tools benutzen zu können, müssen diese beiden Komponenten also auf dem jeweiligen System installiert sein (laut SUN wird JAXP ab Version 1.4. in die Standard-Java-Umgebung integriert sein).

1. Kommandozeilentools (jexmaralda/command)

1.1. Kommandozeilentools für die Konvertierung zwischen EXMARaLDA- und Ausgabe-Formaten

Mit den jexmaralda-Kommandozeilentools kann auf Kommandozeilenebene des Betriebssystems zwischen verschiedenen EXMARaLDA- und Ausgabe-Formaten konvertiert werden. Die nachstehende Abbildung gibt einen Überblick über die möglichen Konvertierungen:



- ① Basic2Segmented
- ② List2Segmented
- ③ Segmented2Basic
- ④ Segmented2List
- ⑤ Basic2PartiturTable
- ⑥ Segmented2PartiturTable
- ⑦ List2PartiturTable

- ① Basic2HTML
- ② Basic2RTF
- ③ Segmented2HTML
- ④ List2HTML
- ⑤ List2RTF

- 1 SimpleExmaralda2List
- 2 CHATExmaralda2List

Im einzelnen arbeiten diese Kommandozeilentools wie folgt:

① Basic2Segmented

konvertiert eine Basis-Transkription in eine segmentierte Transkription und nimmt eine Segmentierung in Turns, Äußerungen und Wörter vor.

Verwendung: `java command.Basic2Segmented input.xml output.xml`
 oder: `java command.Basic2Segmented input.xml output.xml info.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende Basis-Transkription (<basic-transcription>) enthält
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die konvertierte segmentierte Transkription (<segmented-transcription>) geschrieben werden soll
<code>info.xml</code>	der Name (und Pfad) einer Datei, die Informationen zur vorzunehmenden Segmentierung enthält (<word-utterance-segmentation-info-table>). Wenn dieser Name nicht angegeben wird, wird versucht, eine Default-Segmentierungs-Information zu erstellen, und diese zur Segmentierung zu benutzen.

② List2Segmented

konvertiert eine Listen-Transkription in eine segmentierte Transkription und nimmt eine Segmentierung in Wörter vor.

Verwendung: `java command.List2Segmented input.xml output.xml`
 oder: `java command.List2Segmented input.xml output.xml info.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende Listen-Transkription (<list-transcription>) enthält
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die konvertierte segmentierte Transkription (<segmented-transcription>) geschrieben werden soll
<code>info.xml</code>	der Name (und Pfad) einer Datei, die Informationen zur vorzunehmenden Segmentierung enthält (<word-utterance-segmentation-info-table>). Wenn dieser Name nicht angegeben wird, wird versucht, eine Default-Segmentierungs-Information zu erstellen, und diese zur Segmentierung zu benutzen.

③ Segmented2Basic

konvertiert eine segmentierte Transkription in eine Basis-Transkription.

Verwendung: `java command.Segmented2Basic input.xml output.xml`
 oder: `java command.Segmented2Basic input.xml output.xml info.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende segmentierte Transkription (<segmented-transcription>) enthält
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die konvertierte Basis-Transkription (<basic-transcription>) geschrieben werden soll
<code>info.xml</code>	der Name (und Pfad) einer Datei, die Informationen zur vorzunehmenden Konvertierung enthält (<segmented-to-basic-conversion-info>). Wenn dieser Name nicht angegeben wird, wird versucht, eine Default-Konvertierungs-Information zu erstellen, und diese zur Konvertierung zu benutzen.

④ Segmented2List

konvertiert eine segmentierte Transkription in eine Listen-Transkription.

Verwendung: `java command.Segmented2List input.xml output.xml info.xml`
 oder: `java command.Segmented2List input.xml output.xml DEFAULT`
 main-segmentation

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende segmentierte Transkription (<segmented-transcription>) enthält
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die konvertierte Listen-Transkription (<list-transcription>) geschrieben werden soll
<code>info.xml</code>	der Name (und Pfad) einer Datei, die Informationen zur vorzunehmenden Konvertierung enthält (<segmented-to-list-conversion-info>). Wenn anstelle dieses Namens die Zeichenkette DEFAULT angegeben wird, wird versucht, eine Default-Konvertierungs-Information auf Basis der unter <i>main-segmentation</i> angegebenen Haupt-Segmentierung zu erstellen, und diese zur Konvertierung zu benutzen.
<i>main-segmentation</i>	der Name der Haupt-Segmentierung, die bei der Erstellung einer Default-Konvertierungs-Information verwendet werden soll

⑤ Basic2PartiturTable

konvertiert eine Basis-Transkription in eine Partitur-Tabellen-Transkription und nimmt ggf. einen Umbruch auf eine gewisse Pixel-Breite vor.

Verwendung: `java command.Basic2PartiturTable input.xml output.xml DEFAULT`
 oder: `java command.Basic2PartiturTable input.xml output.xml format.xml`
 oder: `java command.Basic2PartiturTable input.xml output.xml DEFAULT`
 pixelWidth respectWords
 oder: `java command.Basic2PartiturTable input.xml output.xml format.xml`
 pixelWidth respectWords

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende Basis-Transkription (<basic-transcription>) enthält
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die konvertierte Partitur-Tabellen-Transkription (<partitur-table-transcription>) geschrieben werden soll
<code>format.xml</code>	der Name (und Pfad) einer Datei, die Formatierungs-Informationen zur vorzunehmenden Konvertierung enthält (<tier-format-table>). Wenn anstelle dieses Namens die Zeichenkette DEFAULT angegeben wird, wird eine Default-Formatierungs-Information für die zu konvertierende Basis-Transkription verwendet
<i>pixelWidth</i>	ein ganzzahliger positiver Wert, der angibt, auf welche Breite (in Pixeln) die Transkription umgebrochen werden soll
<i>respectWords</i>	einer der Werte TRUE oder FALSE , der angibt, ob beim Umbruch Wortgrenzen berücksichtigt werden sollen

⑥ Segmented2PartiturTable

konvertiert eine segmentierte Transkription in eine Partitur-Tabellen-Transkription.

Verwendung: `java command.Segmented2PartiturTable input.xml output.xml`
 oder: `java command.Segmented2PartiturTable input.xml output.xml`
 `format.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende segmentierte Transkription (<segmented-transcription>) enthält
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die konvertierte Partitur-Tabellen-Transkription (<partitur-table-transcription>) geschrieben werden soll
<code>format.xml</code>	der Name (und Pfad) einer Datei, die Formatierungs-Informationen zur vorzunehmenden Konvertierung enthält (<tier-format-table>). Wenn kein solcher Name angegeben ist, wird eine Default-Formatierungs-Information für die zu konvertierende segmentierte Transkription verwendet

⑦ List2PartiturTable

konvertiert eine Listen-Transkription in eine Partitur-Tabellen-Transkription.

Verwendung: `java command.List2PartiturTable input.xml output.xml`
 oder: `java command.List2PartiturTable input.xml output.xml`
 `format.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende Listen-Transkription (<list-transcription>) enthält
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die konvertierte Partitur-Tabellen-Transkription (<partitur-table-transcription>) geschrieben werden soll
<code>format.xml</code>	der Name (und Pfad) einer Datei, die Formatierungs-Informationen zur vorzunehmenden Konvertierung enthält (<tier-format-table>). Wenn kein solcher Name angegeben ist, wird eine Default-Formatierungs-Information für die zu konvertierende Listen-Transkription verwendet

⑧ Basic2HTML

konvertiert eine Basis-Transkription in eine Partitur-Tabellen-Transkription, bricht diese ggf. auf eine angegebene Seitenbreite um und berechnet auf ihrer Grundlage eine Partitur-Darstellung in HTML.

Verwendung: `java command.Basic2HTML input.xml output.html`
 oder: `java command.Basic2HTML input.xml output.html format.xml`
 oder: `java command.Basic2HTML input.xml output.html DEFAULT`
 `paperFormat respectWords`
 oder: `java command.Basic2HTML input.xml output.html format.xml`
 `paperFormat respectWords`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende Basis-Transkription (<basic-transcription>) enthält
<code>output.html</code>	der Name (und Pfad) der Datei, in die die konvertierte HTML-Darstellung geschrieben werden soll
<code>format.xml</code>	der Name (und Pfad) einer Datei, die Formatierungs-Informationen zur vorzunehmenden Konvertierung enthält (<tier-format-table>). Wenn

	kein solcher Name angegeben ist bzw. die Zeichenkette DEFAULT an seiner Stelle steht, wird eine Default-Formatierungs-Information für die zu konvertierende Basis-Transkription verwendet
<i>paperFormat</i>	einer der vier Werte DINA4_VERTICAL , DINA4_HORIZONTAL , DINA3_VERTICAL oder DINA3_HORIZONTAL , der das Seitenformat und die Seitenausrichtung angibt. Wenn kein Wert angegeben ist, wird die Partitur nicht umgebrochen.
<i>respectWords</i>	einer der Werte TRUE oder FALSE , der angibt, ob beim Umbruch Wortgrenzen berücksichtigt werden sollen

② Basic2RTF

konvertiert eine Basis-Transkription in eine Partitur-Tabellen-Transkription, bricht diese auf die angegebene Seitenbreite um und berechnet auf ihrer Grundlage eine Partitur-Darstellung in RTF.

Verwendung: **java command.Basic2RTF** input.xml output.rtf **DEFAULT**
paperFormat respectWords
 oder: **java command.Basic2RTF** input.xml output.rtf format.xml
paperFormat respectWords

Dabei ist

<i>input.xml</i>	der Name (und Pfad) der Datei, die die zu konvertierende Basis-Transkription (<basic-transcription>) enthält
<i>output.rtf</i>	der Name (und Pfad) der Datei, in die die konvertierte RTF-Darstellung geschrieben werden soll
<i>format.xml</i>	der Name (und Pfad) einer Datei, die Formatierungs-Informationen zur vorzunehmenden Konvertierung enthält (<tier-format-table>). Wenn die Zeichenkette DEFAULT an seiner Stelle steht, wird eine Default-Formatierungs-Information für die zu konvertierende Basis-Transkription verwendet
<i>paperFormat</i>	einer der vier Werte DINA4_VERTICAL , DINA4_HORIZONTAL , DINA3_VERTICAL oder DINA3_HORIZONTAL , der das Seitenformat und die Seitenausrichtung angibt. Wenn kein Wert angegeben ist, wird die Partitur nicht umgebrochen.
<i>respectWords</i>	einer der Werte TRUE oder FALSE , der angibt, ob beim Umbruch Wortgrenzen berücksichtigt werden sollen

③ Segmented2HTML

konvertiert eine segmentierte Transkription in eine Partitur-Tabellen-Transkription, und berechnet auf ihrer Grundlage eine Darstellung in HTML.

Verwendung: **java command.Segmented2HTML** input.xml output.html
 oder: **java command.Segmented2HTML** input.xml output.html format.xml

Dabei ist

<i>input.xml</i>	der Name (und Pfad) der Datei, die die zu konvertierende segmentierte Transkription (<segmented-transcription>) enthält
<i>output.html</i>	der Name (und Pfad) der Datei, in die die konvertierte RTF-Darstellung geschrieben werden soll
<i>format.xml</i>	der Name (und Pfad) einer Datei, die Formatierungs-Informationen zur vorzunehmenden Konvertierung enthält (<tier-format-table>). Wenn kein solcher Name angegeben ist, wird eine Default-Formatierungs-Information für die zu konvertierende segmentierte Transkription verwendet

4 List2HTML

konvertiert eine Listen-Transkription in eine Partitur-Tabellen-Transkription, und berechnet auf ihrer Grundlage eine Listen-Darstellung in HTML.

Verwendung: `java command.List2HTML input.xml output.html`
 oder: `java command.Listed2HTML input.xml output.html format.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende Listen-Transkription (<list-transcription>) enthält
<code>output.html</code>	der Name (und Pfad) der Datei, in die die konvertierte RTF-Darstellung geschrieben werden soll
<code>format.xml</code>	der Name (und Pfad) einer Datei, die Formatierungs-Informationen zur vorzunehmenden Konvertierung enthält (<tier-format-table>). Wenn kein solcher Name angegeben ist, wird eine Default-Formatierungs-Information für die zu konvertierende Listen-Transkription verwendet

5 List2RTF

konvertiert eine Listen-Transkription in eine Partitur-Tabellen-Transkription, und berechnet auf ihrer Grundlage eine Listen-Darstellung in RTF.

Verwendung: `java command.List2RTF input.xml output.rtf DEFAULT paperFormat`
 oder: `java command.List2RTF input.xml output.rtf format.xml paperFormat`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die zu konvertierende Listen-Transkription (<list-transcription>) enthält
<code>output.html</code>	der Name (und Pfad) der Datei, in die die konvertierte RTF-Darstellung geschrieben werden soll
<code>format.xml</code>	der Name (und Pfad) einer Datei, die Formatierungs-Informationen zur vorzunehmenden Konvertierung enthält (<tier-format-table>). Wenn an seiner Stelle die Zeichenkette DEFAULT steht, wird eine Default-Formatierungs-Information für die zu konvertierende Listen-Transkription verwendet
<code>paperFormat</code>	einer der vier Werte DINA4_VERTICAL , DINA4_HORIZONTAL , DINA3_VERTICAL oder DINA3_HORIZONTAL , der das Seitenformat und die Seitenausrichtung angibt. Die einzelnen Partitur-Tabellen werden jedoch nicht auf diese Seitenbreite umgebrochen.

1 SimpleExmaralda2List

konvertiert eine Simple Exmaralda-Transkription in eine Listen-Transkription.

Verwendung: `java command.SimpleExmaralda2List input.txt output.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Textdatei, die die zu konvertierende Simple Exmaralda-Transkription enthält. Es wird in der derzeitigen Version davon ausgegangen, dass diese Textdatei nach der Standardkodierungsmethode des jeweiligen Systems kodiert ist. Dies soll allerdings in einer kommenden Version geändert werden, um an dieser Stelle problemlos z.B. auch UNICODE-Textdateien verwenden zu können.
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die konvertierte Listen-Transkription (<list-Transcription>) geschrieben werden soll

2 CHATExmaralda2List

konvertiert eine CHAT Exmaralda-Transkription in eine Listen-Transkription.

Derzeit noch nicht implementiert.

1.2. Kommandozeilentools für Hilfsformate

Die Konvertierungs- und Segmentierungsinformation, die bei der Konvertierung von bzw. in eine segmentierte Transkription benötigt werden und die Formatierungsinformation, die bei der Konvertierung in eine Partitur-Tabllen-Transkription benötigt werden, können i.d.R. automatisch aus den Daten generiert werden. Wenn eine andere als die Default-Information benutzt werden soll, kann diese in einer XML-Datei des entsprechenden EXMARaLDA-Formates kodiert und an die jexmaralda Kommandozeilentools übergeben werden. Um diesen Prozess praktisch einfacher zu gestalten, enthält jexmaralda einige Kommandozeilentools, die die verschiedenen Default-Informationen aus den Daten generieren und in eine Datei schreiben. Wenn dann andere als die Default-Informationen bei der Konvertierung genutzt werden sollen, können einfach diese Dateien geändert werden, was aufgrund ihrer geringen Größe und einfachen Struktur in jedem Texteditor möglich sein sollte. Die Kommandozeilentools arbeiten im einzelnen wie folgt:

1. Format4Basic

generiert eine Default-Formatierungs-Information für eine Basis-Transkription

Verwendung: `java command.Format4Basic input.xml output.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die Basis-Transkription (<basic-transcription>) enthält, für die eine Formatierungs-Information (<tier-format-table>) erstellt werden soll
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die Formatierungs-Information geschrieben werden soll

2. Format4Segmented

generiert eine Default-Formatierungs-Information für eine segmentierte Transkription

Verwendung: `java command.Format4Segmented input.xml output.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die segmentierte Transkription (<segmented-transcription>) enthält, für die eine Formatierungs-Information (<tier-format-table>) erstellt werden soll
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die Formatierungs-Information geschrieben werden soll

3. Format4List

generiert eine Default-Formatierungs-Information für eine Listen-Transkription

Verwendung: `java command.Format4List input.xml output.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die Listen-Transkription (<list-transcription>) enthält, für die eine Formatierungs-Information (<tier-format-table>) erstellt werden soll
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die Formatierungs-Information geschrieben werden soll

4. BasicConversionInfo4Segmented

generiert eine Default-Konvertierungs-Information für die Konvertierung einer segmentierten Transkription in eine Basis-Transkription

Verwendung: `java command.BasicConversionInfo4Segmented input.xml output.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die segmentierte Transkription (<segmented-transcription>) enthält, für die eine Konvertierungs-Information (<segmented-to-basic-conversion-info>) erstellt werden soll
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die Konvertierungs-Information geschrieben werden soll

5. ListConversionInfo4Segmented

generiert eine Default-Konvertierungs-Information für die Konvertierung einer segmentierten Transkription in eine Listen-Transkription

Verwendung: `java command.ListConversionInfo4Segmented input.xml output.xml main-segmentation`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die segmentierte Transkription (<segmented-transcription>) enthält, für die eine Konvertierungs-Information (<segmented-to-list-conversion-info>) erstellt werden soll
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die Konvertierungs-Information geschrieben werden soll
<code>main-segmentation</code>	der Name der Haupt-Segmentierung, die bei der Erstellung der Default-Konvertierungs-Information verwendet werden soll

6. SegmentationInfo4Segmented

generiert eine Default-Segmentierungs-Information für die Segmentierung einer segmentierten Transkription in Turns, Äußerungen und Wörter

Verwendung: `java command.SegmentationInfo4Segmented input.xml output.xml`

Dabei ist

<code>input.xml</code>	der Name (und Pfad) der Datei, die die segmentierte Transkription (<segmented-transcription>) enthält, für die eine Segmentierungs-Information (<word-utterance-segmentation-info-table>) erstellt werden soll
<code>output.xml</code>	der Name (und Pfad) der Datei, in die die Segmentierungs-Information geschrieben werden soll

2. Basis-API (jexmaralda/jexmaralda)

Die jexmaralda-Kommandozeilentools wurden mit Hilfe einer Bibliothek von Basis-Klassen erstellt, die alle grundlegenden Methoden zum Einlesen, Manipulieren und Ausgeben von EXMARaLDA-Transkriptionen enthält. Sollte sich herausstellen, dass auch andere Entwickler auf dieses Basis-API zurückgreifen möchten, wird in einer kommenden Version an dieser Stelle eine ausführliche Dokumentation des Basis-API stehen.

3. GUI-Tools

jexmaralda wird in einer kommenden Version zunächst vor allem um GUI-Tools erweitert werden. Insbesondere wird auf der Basis eines bereits vorhandenen Prototyps ein Partitur-Editor entwickelt werden.

Literatur

MacWhinney, Brian (1995): *CHAT Manual*. <http://childes.psy.cmu.edu/pdf/chat.pdf>

Schmidt, Thomas (2001): *EXMARaLDA – ein System zur Diskurstranskription und –annotation auf dem Computer*. erscheint in: *Arbeiten zur Mehrsprachigkeit*, Hamburg.

Wittenburg, P. / Broeder, D. / Sloman, B. (2000): *EAGLES/ISLE: A Proposal for a Meta Description Standard for Language Resources*. <http://www.talkbank.org/resources/pewi.html>

Anhang A: Beispiel-Daten

A0: Beispieldiskurs

a. als Partitur (Basis-Transkription → Partitur-Tabellen-Transkription → RTF-Partitur)

[1]

	0	1	2	3	4
MAX [v]	Du fällst mir immer		ins Wort.	Siehst Du, Du hast es schon wieder	
MAX [nv]	<i>gestikuliert</i>			<i>schlägt die Hände vors Gesicht</i>	
TOM [v]		Stimmt ja wohl gar nicht.			
TOM [nv]		<i>grinst</i>			
MIA [v]					Er hat
NN [nv]		<i>Telefon klingelt</i>			

[2]

	4	5	6
MAX [v]	getan.		
MIA [v]	schon	recht,	Tom.

b. als Liste (Basis-Transkription → Partitur-Tabellen-Transkription → RTF-Liste)

MAX [v] : Du fällst mir immer 1 2
 [nv] : *gestikuliert*

NN [nv] : 1 2 3 4 5 6
Telefon klingelt

TOM [v] : 1 2 3
 Stimmt ja wohl gar nicht.
 [nv] : *grinst*

MAX [v] : 3 4 5
 Siehst Du, Du hast es schon wieder getan.
 [nv] : *schlägt die Hände vors Gesicht*

MIA [v] : 4 5 6
 Er hat schon recht, Tom.

A1: Kopf (<head>) einer Transkription:

```
<head>
  <meta-information>
    <project-name>Kommunikation in Extremsituationen</project-name>
    <transcription-name>Tropfsteinhoehle</transcription-name>
    <referenced-file url="tropfsteinhoehle.wav"/>
    <ud-meta-information>
      <ud-information attribute-name="recording-quality">reasonable</ud-information>
    </ud-meta-information>
    <comment>Aufnahme eines Gespraches in einer Tropfsteinhohle</comment>
    <transcription-convention>HIAT 2</transcription-convention>
  </meta-information>
  <speakertable>
    <speaker id="SPK2">
      [...]
    </speaker>
    <speaker id="SPK1">
      <abbreviation>KLA</abbreviation>
      <sex value="m"/>
      <languages-used>
        <language xml:lang="de"/>
      </languages-used>
      <l1>
        <language xml:lang="de"/>
        <language xml:lang="fr"/>
      </l1>
      <l2/>
      <ud-speaker-information>
        <ud-information attribute-name="age(years)">23;08;20</ud-information>
      </ud-speaker-information>
      <comment>spricht mit affektiertem Ruhrpott-Akzent</comment>
    </speaker>
    <speaker id="SPK0">
      [...]
    </speaker>
  </speakertable>
</head>
```

A2: Rumpf (<basic-body>) einer Basis-Transkription

```
<basic-body>
  <common-timeline>
    <tli id="T0" absolute-time="0.0"/>
    <tli id="T1"/>
    <tli id="T2"/>
    <tli id="T3" absolute-time="12.5"/>
    <tli id="T4"/>
    <tli id="T5"/>
    <tli id="T6"/>
  </common-timeline>
  <tier id="TIE0" speaker="SPK0" category="v" type="t">
    <event start="T0" end="T1">Du fällst mir immer </event>
    <event start="T1" end="T2">ins Wort. </event>
    <event start="T3" end="T4">Siehst Du, Du hast es schon </event>
    <event start="T4" end="T5">wieder getan. </event>
  </tier>
  <tier id="TIE1" speaker="SPK0" category="nv" type="d">
    <event start="T0" end="T2">gestikuliert</event>
    <event start="T3" end="T5">schlägt die Hände vors Gesicht</event>
  </tier>
  <tier id="TIE2" speaker="SPK1" category="v" type="t">
    <event start="T1" end="T2">Stimmt ja </event>
    <event start="T2" end="T3">wohl gar nicht. </event>
  </tier>
  <tier id="TIE3" speaker="SPK1" category="nv" type="d">
    <event start="T1" end="T3">grinst</event>
  </tier>
  <tier id="TIE4" speaker="SPK2" category="v" type="t">
    <event start="T4" end="T5">Er hat schon </event>
    <event start="T5" end="T6">recht, Tom. </event>
  </tier>
  <tier id="TIE5" speaker="SPK3" category="nv" type="d">
    <event start="T1" end="T6">Telefon klingelt</event>
  </tier>
</basic-body>
```

A3: Rumpf (<segmented-body>) einer segmentierten Transkription

```

<segmented-body>
  <timepoints>
    <timepoint id="T0"/>
    <timepoint id="T1"/>
    <timepoint id="T2"/>
    <timepoint id="T3"/>
    <timepoint id="T4"/>
    [...]
  </timepoints>
  <common-timeline>
    <tpr id="T0"/>
    <tpr id="T1"/>
    <tpr id="T2"/>
    <tpr id="T3"/>
    <tpr id="T4"/>
    <tpr id="T5"/>
    <tpr id="T6"/>
  </common-timeline>
  <segmented-tier id="TIE0" speaker="SPK0" category="v" type="t">
    <timeline>
      <tpr id="T0"/>
      <tpr id="T13"/>
      <tpr id="T14"/>
      <tpr id="T15"/>
      <tpr id="T1"/>
      [...]
      <tpr id="T6"/>
    </timeline>
    <segmentation name="event">
      <segment id="TIE0.EVT0" start="T0" end="T1">Du fällst mir immer </segment>
      <segment id="TIE0.EVT1" start="T1" end="T2">ins Wort. </segment>
      <segment id="TIE0.EVT2" start="T3" end="T4">Siehst Du, Du hast es schon </segment>
      <segment id="TIE0.EVT3" start="T4" end="T5">wieder getan. </segment>
    </segmentation>
    <segmentation name="utterance">
      <segment id="TIE0.UTT0" start="T0" end="T2">Du fällst mir immer ins Wort. </segment>
      <segment id="TIE0.UTT1" start="T3" end="T5">Siehst Du, Du hast es schon wieder getan.
    </segment>
    </segmentation>
    <segmentation name="word">
      <segment id="TIE0.WRD0" start="T0" end="T13">Du</segment>
      <segment id="TIE0.WRD1" start="T13" end="T14">fällst</segment>
      <segment id="TIE0.WRD2" start="T14" end="T15">mir</segment>
      <segment id="TIE0.WRD3" start="T15" end="T1">immer</segment>
      <segment id="TIE0.WRD4" start="T1" end="T6">ins</segment>
      <segment id="TIE0.WRD5" start="T6" end="T2">Wort</segment>
      [...]
    </segmentation>
  </segmented-tier>
  <segmented-tier id="TIE1" speaker="SPK0" category="nv" type="d">
    <timeline>
      <tpr id="T0"/>
      <tpr id="T1"/>
      <tpr id="T2"/>
      <tpr id="T3"/>
      <tpr id="T4"/>
      <tpr id="T5"/>
      <tpr id="T6"/>
    </timeline>
    <segmentation name="event">
      <segment id="TIE1.EVT0" start="T0" end="T2">gestikuliert</segment>
      <segment id="TIE1.EVT1" start="T3" end="T5">schlägt die Hände vors Gesicht</segment>
    </segmentation>
  </segmented-tier>
</segmented-body>

```

A4: Rumpf (<list-body>) einer Listen-Transkription

```

<list-body>
  <common-timeline>
    <tli id="T0"/>
    <tli id="T1"/>
    <tli id="T2"/>
    <tli id="T3"/>
    <tli id="T4"/>
    <tli id="T5"/>
    <tli id="T6"/>
  </common-timeline>
  <tier-references>
    <tier-reference id="TIE0" speaker="SPK0" category="v" type="t"/>
    <tier-reference id="TIE1" speaker="SPK0" category="nv" type="d"/>
    <tier-reference id="TIE2" speaker="SPK1" category="v" type="t"/>
    <tier-reference id="TIE3" speaker="SPK1" category="nv" type="d"/>
    <tier-reference id="TIE4" speaker="SPK2" category="v" type="t"/>
    <tier-reference id="TIE5" speaker="SPK3" category="nv" type="d"/>
  </tier-references>
  <list-item name="utterance" start="T0" end="T2" speaker="SPK0">
    <list-tier level="main" tierref="TIE0">
      <event start="T0" end="T1">Du fällst mir immer </event>
      <event start="T1" end="T2">ins Wort. </event>
    </list-tier>
    <list-tier level="dependent" tierref="TIE1">
      <event start="T0" end="T2">gestikuliert</event>
    </list-tier>
  </list-item>
  <list-item name="utterance" start="T1" end="T6" speaker="SPK3">
    <list-tier level="dependent" tierref="TIE5">
      <event start="T1" end="T6">Telefon klingelt</event>
    </list-tier>
  </list-item>
  <list-item name="utterance" start="T1" end="T3" speaker="SPK1">
    <list-tier level="main" tierref="TIE2">
      <event start="T1" end="T2">Stimmt ja </event>
      <event start="T2" end="T3">wohl gar nicht. </event>
    </list-tier>
    <list-tier level="dependent" tierref="TIE3">
      <event start="T1" end="T3">grinst</event>
    </list-tier>
  </list-item>
  <list-item name="utterance" start="T3" end="T5" speaker="SPK0">
    <list-tier level="main" tierref="TIE0">
      <event start="T3" end="T4">Siehst Du, Du hast es schon </event>
      <event start="T4" end="T5">wieder getan. </event>
    </list-tier>
    <list-tier level="dependent" tierref="TIE1">
      <event start="T3" end="T5">schlägt die Hände vors Gesicht</event>
    </list-tier>
  </list-item>
  <list-item name="utterance" start="T4" end="T6" speaker="SPK2">
    <list-tier level="main" tierref="TIE4">
      <event start="T4" end="T5">Er hat schon </event>
      <event start="T5" end="T6">recht, Tom. </event>
    </list-tier>
  </list-item>
</list-body>

```

A5: Rumpf (<partitur-table-body>) einer Partitur-Tabellen-Transkription (auf der Basis einer Basis-Transkription)

```

<partitur-table-body>
  <common-timeline>
    <tli id="T0"/>
    <tli id="T1"/>
    <tli id="T2"/>
    <tli id="T3"/>
    <tli id="T4"/>
    <tli id="T5"/>
    <tli id="T6"/>
  </common-timeline>
  <tierformat-table>
    <tier-format tierref="TIE5" style-name="Italic" size="8" alignment-name="Center" textcolor-name="black" bgcolor-name="white">
      Arial Unicode MS</tier-format>
    <tier-format tierref="TIE4" style-name="Plain" size="10" alignment-name="Left" textcolor-name="black" bgcolor-name="white">
      Arial Unicode MS</tier-format>
    <tier-format tierref="TIE3" style-name="Italic" size="8" alignment-name="Center" textcolor-name="black" bgcolor-name="white">
      Arial Unicode MS</tier-format>
    [...]
  </tierformat-table>
  <partitur-table>
    <head-row>
      <partitur-row id="COLUMN-LABELS">
        <label>
          <partitur-cell id="EMPTY" anchor="" hyperlink="" column="-1" width="49" span="1"/>
        </label>
        <partitur-cell id="COLUMN-LABEL" anchor="T0" hyperlink="" column="0" width="86" span="1">0</partitur-cell>
        <partitur-cell id="COLUMN-LABEL" anchor="T1" hyperlink="" column="1" width="44" span="1">1</partitur-cell>
        <partitur-cell id="COLUMN-LABEL" anchor="T2" hyperlink="" column="2" width="67" span="1">2</partitur-cell>
        [...]
      </partitur-row>
    </head-row>
    <partitur-row id="TIE0">
      <label>
        <partitur-cell id="ROW-LABEL" anchor="" hyperlink="#SPK0" column="-1" width="49" span="1">MAX [v]</partitur-cell>
      </label>
      <partitur-cell id="TIE0" anchor="" hyperlink="" column="0" width="86" span="1">Du fällst mir immer </partitur-cell>
      <partitur-cell id="TIE0" anchor="" hyperlink="" column="1" width="44" span="1">ins Wort. </partitur-cell>
      <partitur-cell id="EMPTY" anchor="" hyperlink="" column="2" width="67" span="1"/>
      <partitur-cell id="TIE0" anchor="" hyperlink="" column="3" width="129" span="1">Siehst Du, Du hast es schon </partitur-cell>
      <partitur-cell id="TIE0" anchor="" hyperlink="" column="4" width="63" span="1">wieder getan. </partitur-cell>
      [...]
    </partitur-row>
    [...]
  </partitur-table>
</partitur-table-body>

```

A6: Rumpf (<partitur-table-body>) einer Partitur-Tabellen-Transkription (auf der Basis einer Listen-Transkription)

```

<partitur-table-body>
  <common-timeline>
  [...]
  </common-timeline>
  <tierformat-table>
    <tier-format tierref="TIE5" style-name="Italic" size="8" alignment-name="Center" textcolor-name="black" bgcolor-name="white">
      Arial Unicode MS</tier-format>
    [...]
  </tierformat-table>
  <partitur-table>
    <head-row>
      <partitur-row id="COLUMN-LABELS">
        <label>
          <partitur-cell id="EMPTY" anchor="" hyperlink="" column="-1" width="1" span="1"/>
        </label>
        <partitur-cell id="COLUMN-LABEL" anchor="T0" hyperlink="" column="0" width="0" span="1">0</partitur-cell>
        [...]
      </partitur-row>
    </head-row>
    <partitur-row id="TIE0">
      <label>
        <partitur-cell id="ROW-LABEL" anchor="" hyperlink="" column="-1" width="0" span="1">MAX [v] :</partitur-cell>
      </label>
      <partitur-cell id="TIE0" anchor="" hyperlink="" column="0" width="0" span="1">Du fällst mir immer </partitur-cell>
      <partitur-cell id="TIE0" anchor="" hyperlink="" column="1" width="0" span="1">ins Wort. </partitur-cell>
    </partitur-row>
    <partitur-row id="TIE1">
      <label>
        <partitur-cell id="SUB-ROW-LABEL" anchor="" hyperlink="" column="-1" width="0" span="1"> [nv] :</partitur-cell>
      </label>
      <partitur-cell id="TIE1" anchor="" hyperlink="" column="0" width="0" span="2">gestikuliert</partitur-cell>
    </partitur-row>
  </partitur-table>
  <partitur-table>
    <head-row>
    [...]
  </head-row>
  <partitur-row id="TIE5">
    <label>
      <partitur-cell id="ROW-LABEL" anchor="" hyperlink="" column="-1" width="0" span="1">NN [nv] :</partitur-cell>
    </label>
    <partitur-cell id="TIE5" anchor="" hyperlink="" column="0" width="0" span="5">Telefon klingelt</partitur-cell>
  </partitur-row>
</partitur-table>
</partitur-table-body>

```

A7: Formatierungs-Information (<tier-format-table>)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tierformat-table SYSTEM "tierformat-table.dtd">
<!-- (c) 2001 by Thomas Schmidt (thomas.schmidt@uni-hamburg.de) -->
<tierformat-table>
  <referenced-file url="E:\XML\Beispiele\Basic\Kolloquium_Basic2.xml"/>
  <tier-format tierref="TIE5" style-name="Italic" size="8" alignment-name="Center" textcolor-name="black" bgcolor-name="white">Arial Unicode MS</tier-format>
  <tier-format tierref="TIE4" style-name="Plain" size="10" alignment-name="Left" textcolor-name="black" bgcolor-name="white">Arial Unicode MS</tier-format>
  <tier-format tierref="TIE3" style-name="Italic" size="8" alignment-name="Center" textcolor-name="black" bgcolor-name="white">Arial Unicode MS</tier-format>
  <tier-format tierref="TIE2" style-name="Plain" size="10" alignment-name="Left" textcolor-name="black" bgcolor-name="white">Arial Unicode MS</tier-format>
  <tier-format tierref="TIE1" style-name="Italic" size="8" alignment-name="Center" textcolor-name="black" bgcolor-name="white">Arial Unicode MS</tier-format>
  <tier-format tierref="TIE0" style-name="Plain" size="10" alignment-name="Left" textcolor-name="black" bgcolor-name="white">Arial Unicode MS</tier-format>
  <tier-format tierref="COLUMN-LABEL" style-name="Plain" size="5" alignment-name="Left" textcolor-name="black" bgcolor-name="white">Arial Unicode MS</tier-format>
  <tier-format tierref="ROW-LABEL" style-name="Bold" size="10" alignment-name="Left" textcolor-name="black" bgcolor-name="white">Arial Unicode MS</tier-format>
  <tier-format tierref="SUB-ROW-LABEL" style-name="Plain" size="8" alignment-name="Right" textcolor-name="black" bgcolor-name="white">Arial Unicode MS</tier-format>
  <tier-format tierref="EMPTY" style-name="Plain" size="2" alignment-name="Left" textcolor-name="white" bgcolor-name="white">Arial Unicode MS</tier-format>
</tierformat-table>
```

A8: Konvertierungs-Information segmentierte Transkription → Basis-Transkription (<segmented-to-basic-conversion-info>)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE segmented-to-basic-conversion-info SYSTEM "segmented-to-basic-conversion-info.dtd">
<!-- (c) 2001 by Thomas Schmidt (thomas.schmidt@uni-hamburg.de) -->
<segmented-to-basic-conversion-info>
  <referenced-file url="E:\XML\Beispiele\Segmented\Kolloquium_Segmented.xml"/>
  <info tierref="TIE5" segmentation="event"/>
  <info tierref="TIE4" segmentation="event"/>
  <info tierref="TIE3" segmentation="event"/>
  <info tierref="TIE2" segmentation="event"/>
  <info tierref="TIE1" segmentation="event"/>
  <info tierref="TIE0" segmentation="event"/>
</segmented-to-basic-conversion-info>
```

A9: Konvertierungs-Information segmentierte Transkription → Listen-Transkription (<segmented-to-list-conversion-info>)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE segmented-to-list-conversion-info SYSTEM "segmented-to-list-conversion-info.dtd">
<!-- (c) 2001 by Thomas Schmidt (thomas.schmidt@uni-hamburg.de) -->
<segmented-to-list-conversion-info main-segmentation="utterance">
  <referenced-file url="E:\XML\Beispiele\Segmented\Kolloquium_Segmented.xml"/>
  <info tierref="TIE5" segmentation="event"/>
  <info tierref="TIE4" segmentation="event"/>
  <info tierref="TIE3" segmentation="event"/>
  <info tierref="TIE2" segmentation="event"/>
  <info tierref="TIE1" segmentation="event"/>
  <info tierref="TIE0" segmentation="event"/>
  <main-tier speaker="SPK2" tierID="TIE4"/>
  <main-tier speaker="SPK1" tierID="TIE2"/>
  <main-tier speaker="SPK0" tierID="TIE0"/>
</segmented-to-list-conversion-info>
```

A10: Segmentierungs-Information(<word-utterance-segmentation-info-table>)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE word-utterance-segmentation-info-table SYSTEM "word-utterance-segmentation-info-table.dtd">
<!-- (c) 2001 by Thomas Schmidt (thomas.schmidt@uni-hamburg.de) -->
<word-utterance-segmentation-info-table>
  <referenced-file url="E:\XML\Beispiele\Segmented\Kolloquium_Segmented.xml"/>
  <word-utterance-segmentation-info tierref="TIE4">
    <utterance-end-symbol symbol="." "/>
    <utterance-end-symbol symbol="!" "/>
    <utterance-end-symbol symbol="?" "/>
    <utterance-end-symbol symbol="\ " "/>
    <word-end-symbol symbol=" " "/>
    <word-end-symbol symbol="," "/>
    <word-end-symbol symbol=";" "/>
  </word-utterance-segmentation-info>
  <word-utterance-segmentation-info tierref="TIE2">
    <utterance-end-symbol symbol="." "/>
    <utterance-end-symbol symbol="!" "/>
    <utterance-end-symbol symbol="?" "/>
    <utterance-end-symbol symbol="\ " "/>
    <word-end-symbol symbol=" " "/>
    <word-end-symbol symbol="," "/>
    <word-end-symbol symbol=";" "/>
  </word-utterance-segmentation-info>
  <word-utterance-segmentation-info tierref="TIE0">
    [...]
  </word-utterance-segmentation-info>
</word-utterance-segmentation-info-table>
```

Index

Annotation 5
Basis-Transkription 4, 8, 16, 28
CHAT Exmaralda 14, 21
Formatierungs-Information 4, 11, **21**, 33
HTML-Ausgabe 18
HTML-Darstellung **6**
Konvertierungs-Information 4, 11, **22**, 34
Listen-Transkription 4, 9, 16, 30
Partitur-Tabellen-Transkription 4, 9, 17, 31
RTF-Ausgabe 19
RTF-Darstellung **6**, 26
segmentierte Transkription 4, 8, 16, 29
Segmentierungs-Information 4, 12, **22**, 35
Simple Exmaralda 13, 20
Sprechertabelle 7