SGF: An integrated model for multiple annotations and its application in a linguistic domain

Maik Stührenberg

Daniela Goecke

Introductory words

In the early days enriching textual information was quite different...

Introductory words

...and was made much easier when digital text was invented...

This is a sentence.

Introductory words

...especially when standardized markup language were established.

```
<s>This is a sentence.</s>
```

Introductory words

...especially when standardized markup language were established.

```
<s>
 <np>
  <pron>This</pron>
 </np>
 <vp>
  <v>is</v>
  <np>
   <det>a</det>
   <n>sentence</n>
  </np>
 </vp>
 .
</s>
```

Introductory words

However, when multiple markup entered the stage, things got difficult again...

```
<s>
<syllables>
 <np>
  <pron><syll>This</syll></pron>
 </np>
 <vp>
  <v><syll>is</syll></v>
  <np>
   <det><syll>a</syll></det>
   <n><syll>sen</syll><syll>tence</syll></n>
  </np>
 </vp>
 .
</syllables>
</s>
```

Introductory words

The problem is –
in the field we're working in we need multiple annotated documents...

Introductory words

...therefore, this talk will focus on different methods of dealing with multiple annotations in general –
and one format in detail.

Overview

- Architectures for multiple annotations
- SGF – The Sekimo Generic Format
- Real World Application Scenarios

Overview

- Architectures for multiple annotations
- SGF – The Sekimo Generic Format
- Real World Application Scenarios

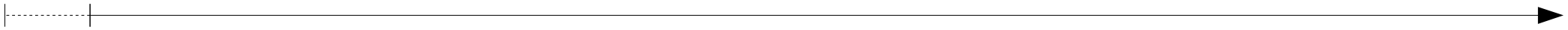
Architectures for multiple annotations

Architectures for storing multiple annotations developed in the last years:

1. Prolog-based
2. XML-related
3. XML-based Graph models

Architectures for multiple annotations – Prolog-based

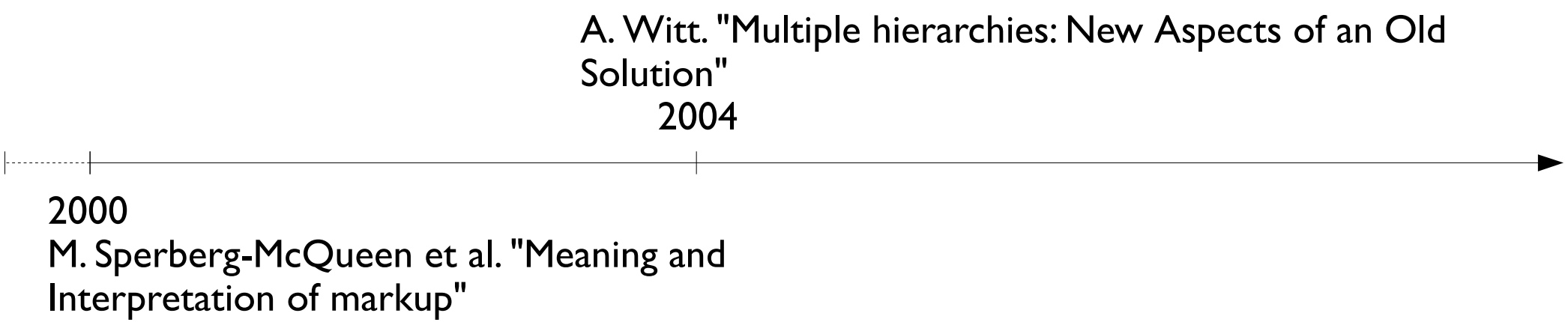
Evolution:



2000
M. Sperberg-McQueen et al. "Meaning and Interpretation of markup"

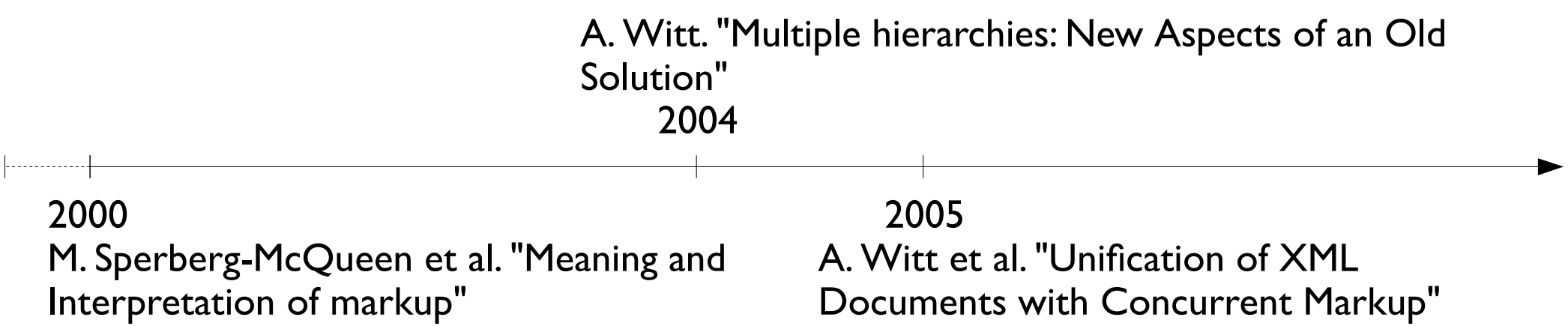
Architectures for multiple annotations – Prolog-based

Evolution:



Architectures for multiple annotations – Prolog-based

Evolution:



Architectures for multiple annotations – Prolog-based

The following information is saved as Prolog predicates:

- The type of node (element, attribute or text) as the name of the predicate
- The name of the annotation layer
- The absolute start and end positions of the annotated text sequence
- The position of the node in the document tree
- The name of the element or attribute
- The value of an attribute

Architectures for multiple annotations – Prolog-based

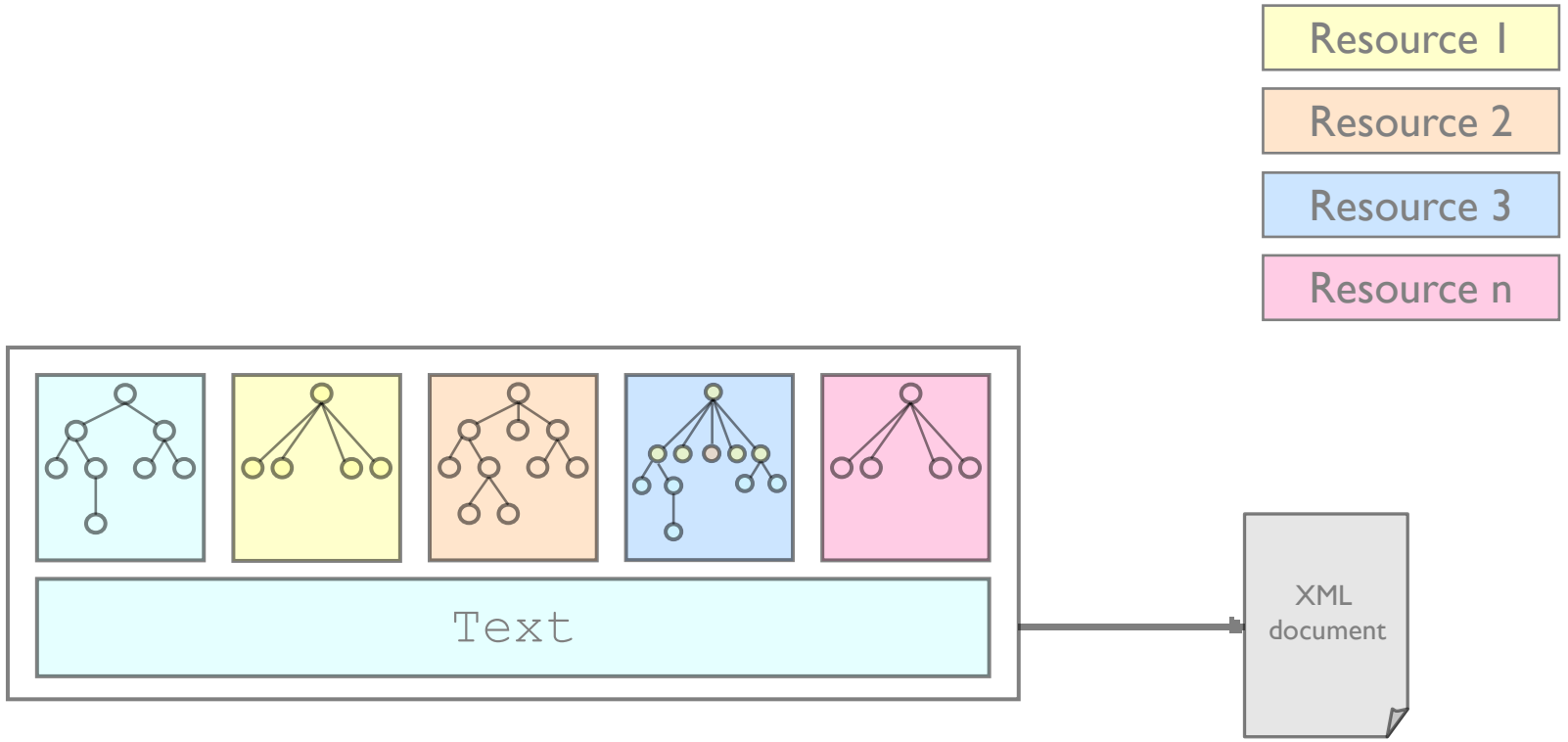
The following information is saved as Prolog predicates:

- The type of node (element, attribute or text) as the name of the predicate
- The name of the annotation layer
- The absolute start and end positions of the annotated text sequence
- The position of the node in the document tree
- The name of the element or attribute
- The value of an attribute

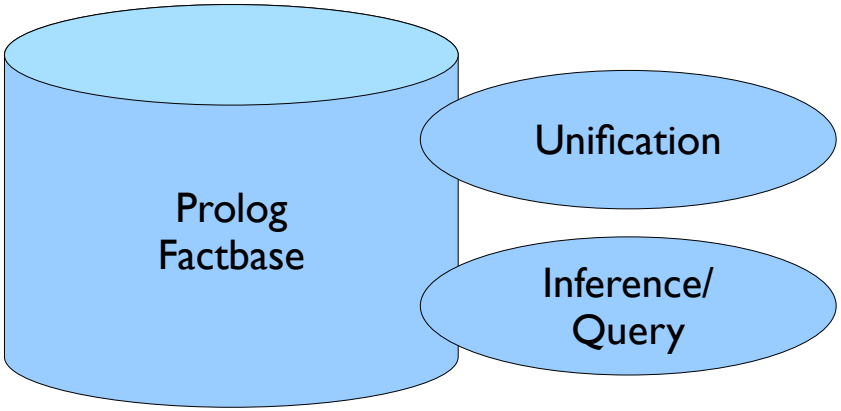
```
node('ling-deu-003-chs.xml', 0, 67831, [1], element('txt_text')).
node('ling-deu-003-chs.xml', 0, 67831, [1, 1], element('cnx-pi_analysis')).
node('ling-deu-003-chs.xml', 0, 22, [1, 1, 1], element('txt_para')).
attr('ling-deu-003-chs.xml', 0, 67831, [1], 'lang', 'de').
attr('ling-deu-003-chs.xml', 0, 22, [1, 1, 1], 'lang', 'en').
attr('ling-deu-003-chs.xml', 0, 22, [1, 1, 1], 'skip', 'yes').
pcdata_node(0, 1, 'L').
pcdata_node(1, 2, 'i').
pcdata_node(2, 3, 'n').
```

Architectures for multiple annotations – Prolog-based

Example:



Architecture of the first phase of the Sekimo project



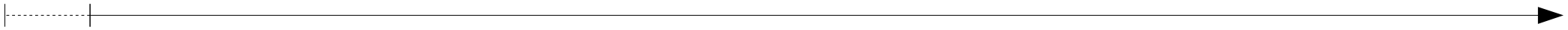
Architectures for multiple annotations – Prolog-based

Disadvantages:

- Conversion from XML to Prolog (and back) is necessary both for markup unification and for analyzing relations between different annotation levels
- The need for information about the position of each single character of the primary data – which is demanded for reconstructing the primary data – results in quite large files
- The whole process is quite complex

Architectures for multiple annotations – XML-related

Evolution/Examples:



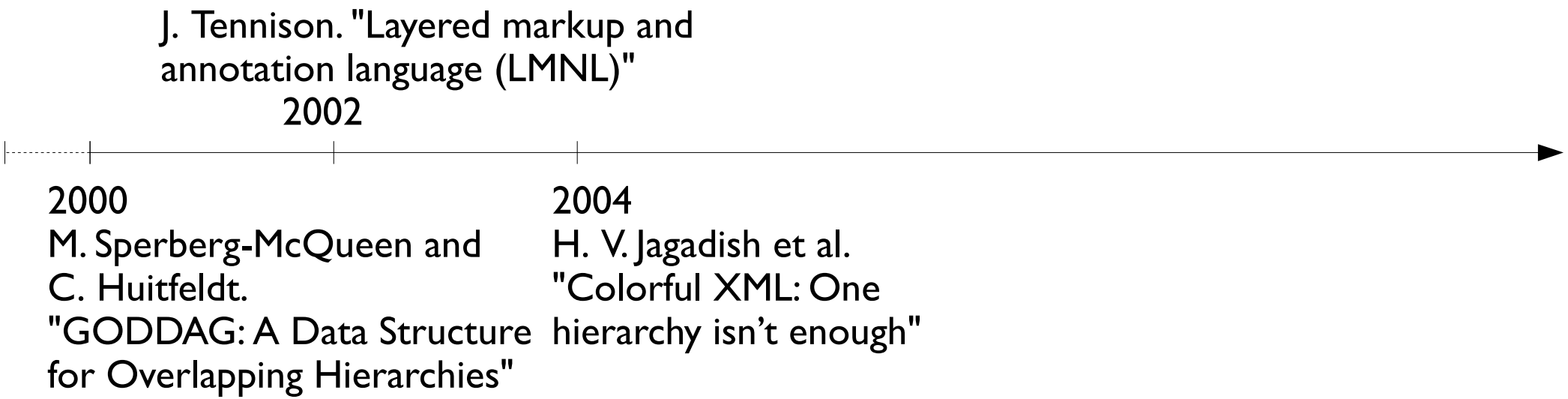
2000
M. Sperberg-McQueen and
C. Huitfeldt.
"GODDAG: A Data Structure
for Overlapping Hierarchies"

Architectures for multiple annotations – XML-related

Evolution/Examples:

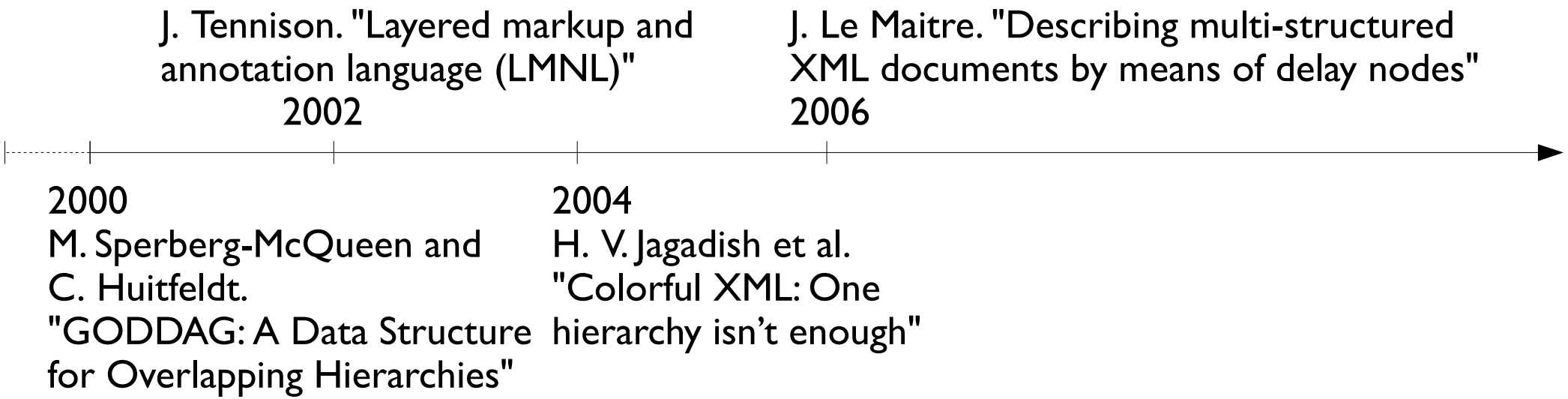
Architectures for multiple annotations – XML-related

Evolution/Examples:



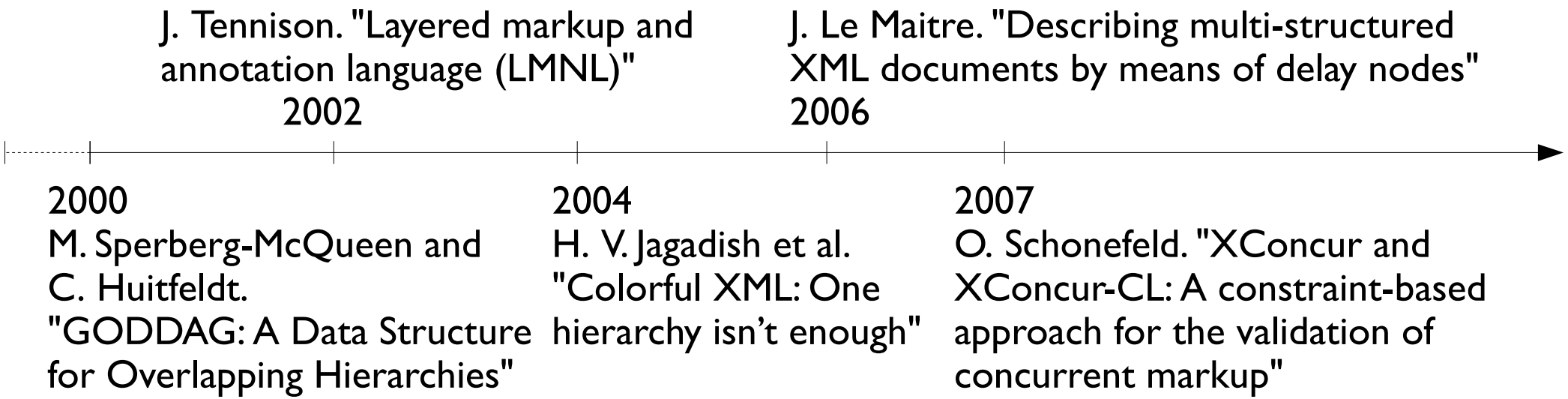
Architectures for multiple annotations – XML-related

Evolution/Examples:



Architectures for multiple annotations – XML-related

Evolution/Examples:



Architectures for multiple annotations – XML-related

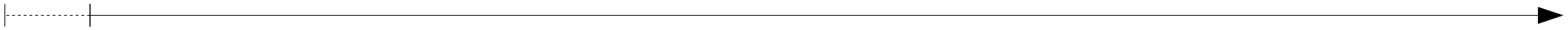
Disadvantages:

- Although inline annotation of multiple annotation layers is supported, these documents can get very complex
- Both, design and implementation rely on the work of only a few people – future development is often unknown
- Lack of support of the large number of tools that is available for XML-based solutions – no option if you want to stay in the XML context

```
<?xconcur version="1.1" encoding="utf-8"?>
<?xconcur-schema layer="l1" root="s"
 system="pos.xsd"?>
<?xconcur-schema layer="l2" root="syllables"
 system="syll.xsd"?>
<(l1)s>
<(l2)syllables>
 <(l1)np>
  <(l1)pron>
   <(l2)syll>This</(l2)syll>
  </(l1)pron>
 </(l1)np>
 <(l1)vp>
  <(l1)v>
   <(l2)syll>is</(l2)syll>
  </(l1)v>
  <(l1)np>
   <(l1)det>
    <(l2)syll>a</(l2)syll>
   </(l1)det>
   <(l1)n>
    <(l2)syll>sen</(l2)syll>
    <(l2)syll>tence</(l2)syll>
   </(l1)n>
  </(l1)np>
 </(l1)vp>.
</(l2)syllables>
</(l1)s>
```

Architectures for multiple annotations – Graph-based

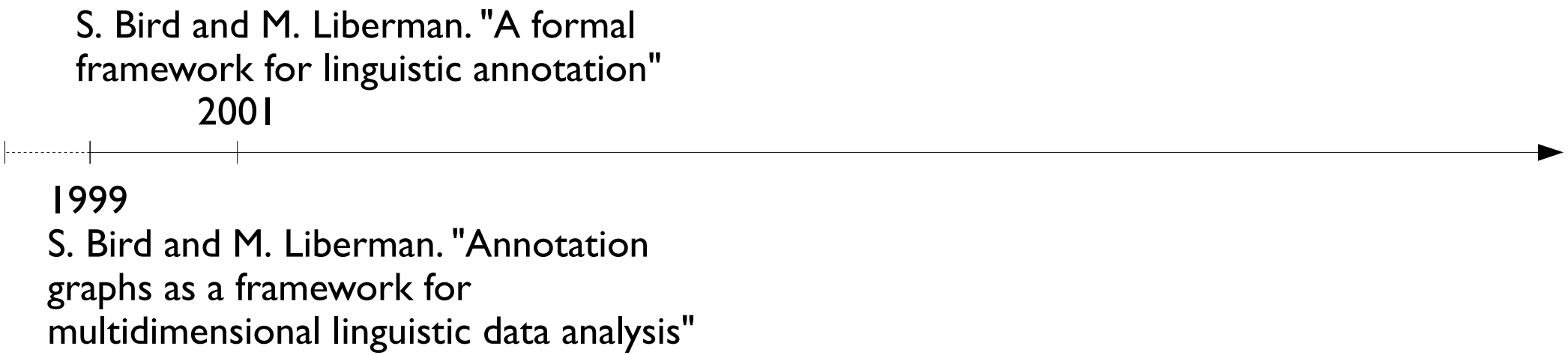
Evolution:



1999
S. Bird and M. Liberman. "Annotation graphs as a framework for multidimensional linguistic data analysis"

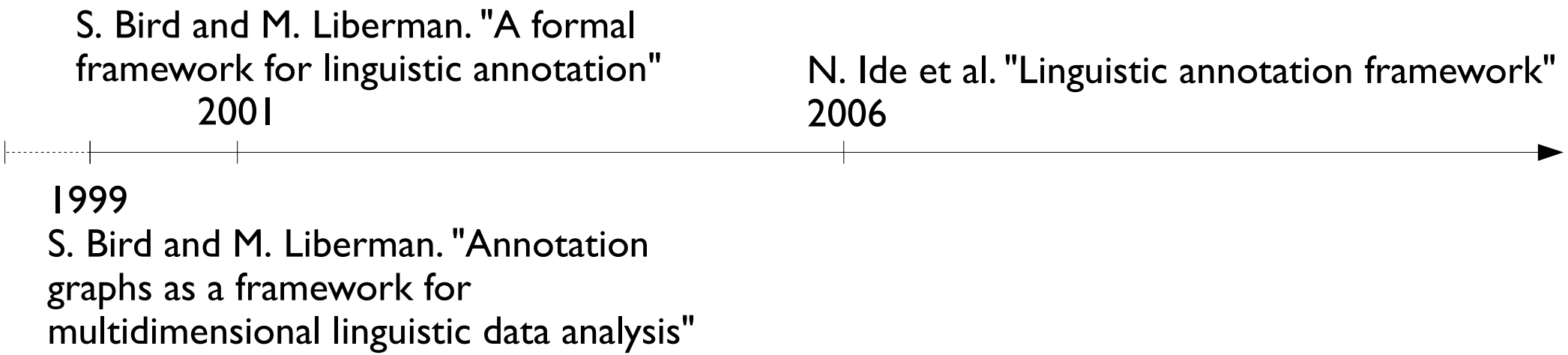
Architectures for multiple annotations – Graph-based

Evolution:



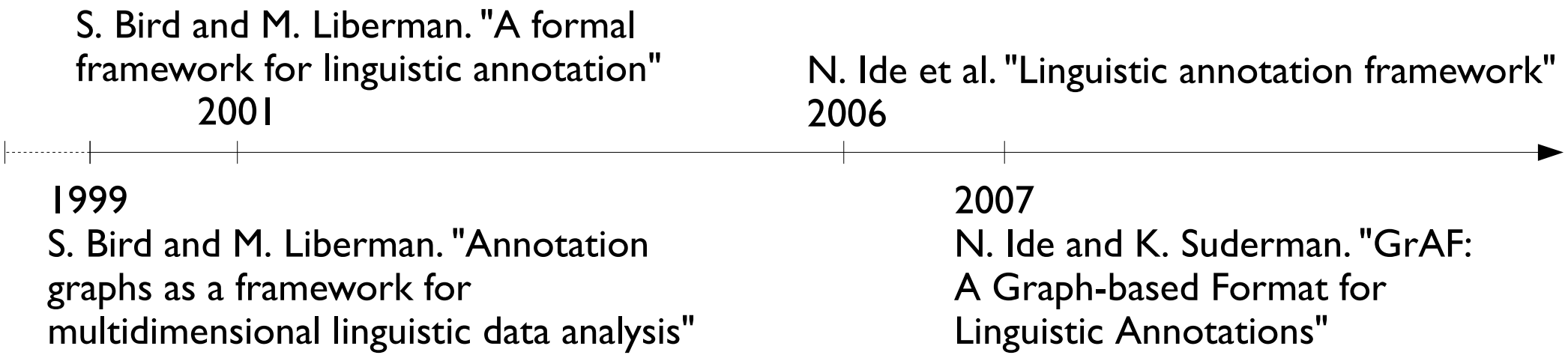
Architectures for multiple annotations – Graph-based

Evolution:



Architectures for multiple annotations – Graph-based

Evolution:



Architectures for multiple annotations – Graph-based

Additional Examples:

- NITE-XML
- PAULA (Potsdamer Austauschformat für linguistische Annotationen)
- GXL – Graph eXchange Language

...to name just a few

Architectures for multiple annotations – Graph-based

In principle, graph-based annotation formats are quite nice for the task of dealing with multiple annotated data – and allow for staying in the XML context...

Architectures for multiple annotations – Graph-based

...however, there are some disadvantages of the existing graph-based approaches:

- Often (even single) annotations are split into multiple files:
  - primary data file
  - markable/token file
    delimits text spans used in annotation
  - structure file
    stores relations between annotation elements
  - feature file
    stores the former annotation

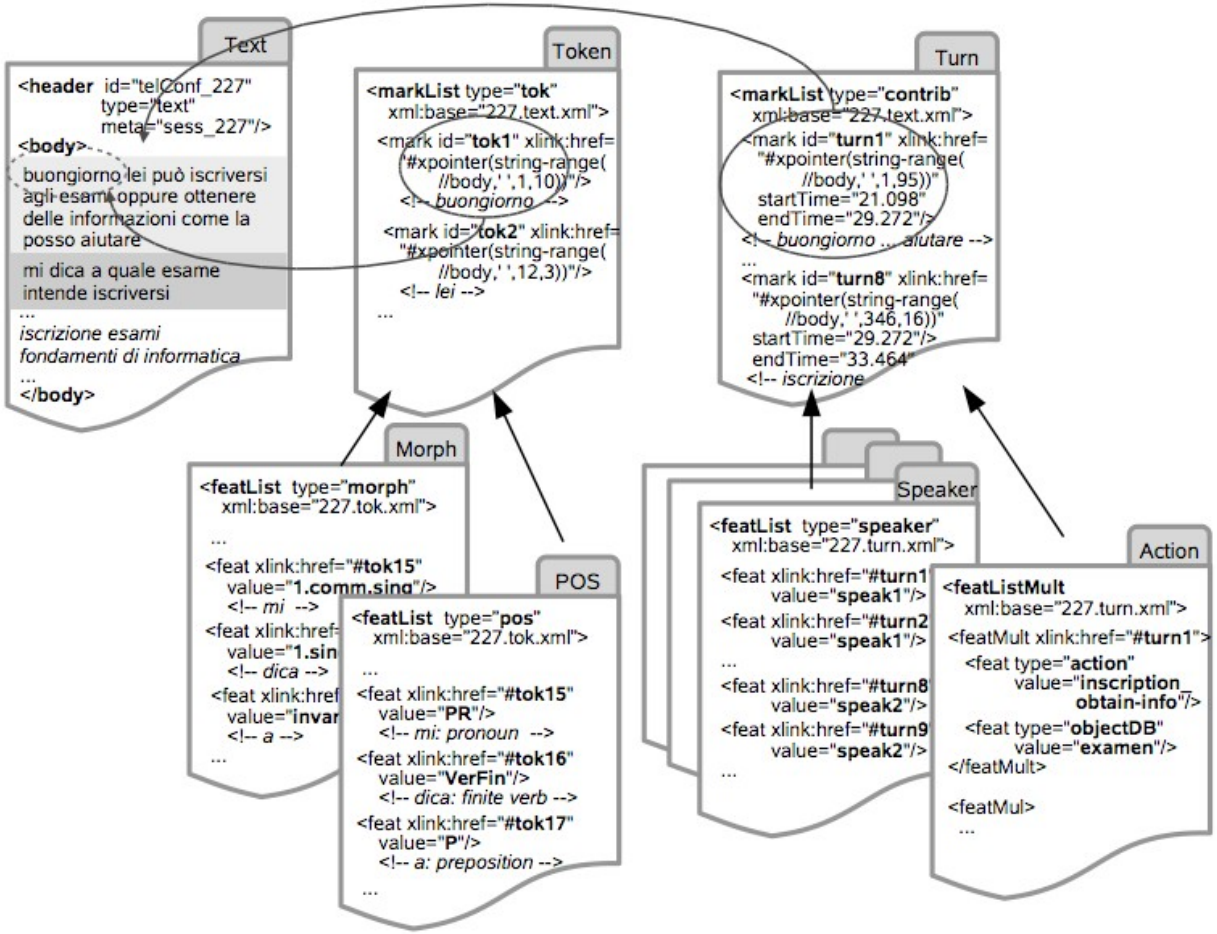


Figure taken from K. J. Rodríguez et al., 2007

Architectures for multiple annotations – Graph-based

Other disadvantages:

- Graph-based architectures tend to provide interchange formats, not annotation formats
- Although it is nice to have the expressiveness of a graph format, it is often breaking a butterfly on a wheel – because most single annotation layers can be structured in trees

We propose an architecture that is mostly tree based (including inline annotation) – but can use graphs as well if needed

Overview

- Architectures for multiple annotations
- SGF – The Sekimo Generic Format
- Real World Application Scenarios

SGF – The Sekimo Generic Format

Some considerations:

SGF – The Sekimo Generic Format

Some considerations:

- Multiple annotations (in separate files)

```
<s>
 <np>
  <pron>This</pron>
 </np>
 <vp>
  <v>is</v>
  <np>
   <det>a</det>
   <n>sentence</n>
  </np>
 </vp>
 .
</s>
```

```
<syllables>
 <syll>This</syll>
 <syll>is</syll>
 <syll>a</syll>
 <syll>sen</syll>
 <syll>tence</syll>
 .
</syllables>
```

SGF – The Sekimo Generic Format

Some considerations:

- Primary data is stored redundant

```
<s>
 <np>
  <pron>This</pron>
 </np>
 <vp>
  <v>is</v>
  <np>
   <det>a</det>
   <n>sentence</n>
  </np>
 </vp>
 .
</s>
```

```
<syllables>
 <syll>This</syll>
 <syll>is</syll>
 <syll>a</syll>
 <syll>sen</syll>
 <syll>tence</syll>
 .
</syllables>
```

SGF – The Sekimo Generic Format

Some considerations:

- Let's start with deleting this redundancy by using offsets...

```
  T  h  i  s     i  s     a     s  e  n  t  e  n  c  e  .
00|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15|16|17|18|19
```

SGF – The Sekimo Generic Format

Some considerations:

- Let's start with deleting this redundancy by using offsets...

```
  T  h  i  s     i  s     a     s  e  n  t  e  n  c  e  .
00|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15|16|17|18|19
```

```
<s start="0" end="19">
 <np start="0" end="4">
  <pron start="0" end="4"/>
 </np>
 <vp start="5" end="18">
  <v start="5" end="7"/>
  <np start="8" end="18">
   <det start="8" end="9"/>
   <n start="10" end="18"/>
  </np>
 </vp>
</s>
```

```
<syllables start="0" end="19">
 <syll start="0" end="4"/>
 <syll start="5" end="7"/>
 <syll start="8" end="9"/>
 <syll start="10" end="13"/>
 <syll start="13" end="18"/>
</syllables>
```

SGF – The Sekimo Generic Format

Some considerations:

- ...but there is still some redundancy left...

```
<s start="0" end="19">
 <np start="0" end="4">
  <pron start="0" end="4"/>
 </np>
 <vp start="5" end="18">
  <v start="5" end="7"/>
  <np start="8" end="18">
   <det start="8" end="9"/>
   <n start="10" end="18"/>
  </np>
 </vp>
</s>
```

```
<syllables start="0" end="19">
 <syll start="0" end="4"/>
 <syll start="5" end="7"/>
 <syll start="8" end="9"/>
 <syll start="10" end="13"/>
 <syll start="13" end="18"/>
</syllables>
```

SGF – The Sekimo Generic Format

Some considerations:

- getting closer...

```
<s segment="s0">
 <np segment="s0">
  <pron segment="s1"/>
 </np>
 <vp segment="s2">
  <v segment="s3"/>
  <np segment="s4">
   <det segment="s5"/>
   <n segment="s6"/>
  </np>
 </vp>
</s>
```

```
<segment id="s0" start="0"
end="19"/>
<segment id="s1" start="0"
end="4"/>
<segment id="s2" start="5"
end="18"/>
<segment id="s3" start="5"
end="7"/>
<segment id="s4" start="8"
end="18"/>
<segment id="s5" start="8"
end="9"/>
<segment id="s6" start="10"
end="18"/>
<segment id="s7" start="10"
end="13"/>
<segment id="s8" start="13"
end="18"/>
```

```
<syllables segment="s0">
 <syll segment="s1"/>
 <syll segment="s3"/>
 <syll segment="s5"/>
 <syll segment="s7"/>
 <syll segment="s8"/>
</syllables>
```

SGF – The Sekimo Generic Format

Final step:

- Putting it all together in a single file
  - Allows for XML ID/IDREF for connecting segments of the primary data with the respective annotation(s)
  - Eases import & export
- Reuse as much available standards and formats as possible – try not to reinvent the wheel
  - Existing metadata specifications
  - Global XML attributes such as `xml:lang` or `xml:id`

```
<segment id="s0" start="0" end="19"/>
<segment id="s1" start="0" end="4"/>
<segment id="s2" start="5" end="18"/>
<segment id="s3" start="5" end="7"/>
<segment id="s4" start="8" end="18"/>
<segment id="s5" start="8" end="9"/>
<segment id="s6" start="10" end="18"/>
<segment id="s7" start="10" end="13"/>
<segment id="s8" start="13" end="18"/>
<s segment="s0">
 <np segment="s0">
  <pron segment="s1"/>
 </np>
 <vp segment="s2">
  <v segment="s3"/>
  <np segment="s4">
   <det segment="s5"/>
   <n segment="s6"/>
  </np>
 </vp>
</s>
<syllables segment="s0">
 <syll segment="s1"/>
 <syll segment="s3"/>
 <syll segment="s5"/>
 <syll segment="s7"/>
 <syll segment="s8"/>
</syllables>
```

SGF – The Sekimo Generic Format

SGF is a logical descendant to the before mentioned Prolog fact base:

- It uses the Annotation Graph model, i.e. annotations are aligned via character positions (if textual primary data is used) or positions in time
- Multiple annotations can be saved in a single XML file, a relational database, a native XML database or a hybrid database
- Standard (i.e. unmodified) XPath and XQuery is used for inferencing and analyzing multiple annotations

SGF – The Sekimo Generic Format

SGF knows two types of layers:

- The base layer
- One or more annotation layer(s)

SGF – The base layer

The base layer contains:

- Primary data
  - Textual primary data can be referenced via the `uri` attribute of the `location` element or stored directly as element content of the `textualContent` child element of the `primaryData` element
  - Multimodal primary data is referenced via the `uri` attribute of the `location` element – multiple primary data is possible, however, there has to be one master primary data (depicted by the `role` attribute) providing the master time line

```
<base:corpus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.text-technology.de/sekimo root.xsd"
 xmlns="http://www.text-technology.de/sekimo"
 xmlns:base="http://www.text-technology.de/sekimo">
 <base:corpusData id="c1" type="text">
  <base:primaryData start="0" end="1036" xml:lang="de">
   <base:location uri="ex1.txt" encoding="utf-8"/>
  </base:primaryData>
  <base:checksum algorithm="md5">15a244f76923</base:checksum>
 </base:corpusData>
</base:corpus>
```

SGF – The base layer

The base layer contains:

- An optional checksum for verifying primary data
- Optional metadata which is specified by an external document grammar (OLAC in our case) – metadata can be applied to the whole corpus, as child of the `corpusData` element or underneath an annotation level

```
<base:corpus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.text-technology.de/sekimo root.xsd"
 xmlns:base="http://www.text-technology.de/sekimo">
 <base:corpusData id="c1" type="text">
  <base:primaryData start="0" end="1036" xml:lang="de">
   <base:location uri="ex1.txt" encoding="utf-8"/>
  </base:primaryData>
  <base:checksum algorithm="md5">15a244f76923</base:checksum>
  <base:meta>
   <olac:olac xmlns:olac="http://www.language-archives.org/OLAC/1.0/"
    xmlns="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xsi:schemaLocation="http://www.language-archives.org/OLAC/1.0/ olac.xsd">
   </olac:olac>
  </base:meta>
 </base:corpusData>
</base:corpus>
```

SGF – Adding annotation layers

```
<base:corpus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.text-technology.de/sekimo root.xsd"
 xmlns:base="http://www.text-technology.de/sekimo">
 <base:corpusData id="c1" type="text">
  <base:primaryData start="0" end="1036" xml:lang="de">
   <base:location uri="ex1.txt" encoding="utf-8"/>
  </base:primaryData>
  <base:meta>
   <olac:olac xmlns:olac="http://www.language-archives.org/OLAC/1.0/"
    xmlns="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/">
   </olac:olac>
  </base:meta>
 </base:corpusData>
</base:corpus>
```

We start with the primary data file

SGF – Adding annotation layers

```
<base:corpus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.text-technology.de/sekimo root.xsd"
 xmlns:base="http://www.text-technology.de/sekimo">
 <base:corpusData id="c1" type="text">
  <base:primaryData start="0" end="1036" xml:lang="de">
   <base:location uri="ex1.txt" encoding="utf-8"/>
  </base:primaryData>
  <base:meta>
   <olac:olac xmlns:olac="http://www.language-archives.org/OLAC/1.0/"
    xmlns="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/">
   </olac:olac>
  </base:meta>
  <base:segments>
   <base:segment id="seg1" start="0" end="100"/>
   <base:segment id="seg2" start="100" end="110"/>
  </base:segments>
 </base:corpusData>
</base:corpus>
```

At first, the segments which delimit the annotated data are defined. Segments can be established by start and end positions...

SGF – Adding annotation layers

```
<base:corpus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.text-technology.de/sekimo root.xsd"
 xmlns:base="http://www.text-technology.de/sekimo">
 <base:corpusData id="c1" type="text">
  <base:primaryData start="0" end="1036" xml:lang="de">
   <base:location uri="ex1.txt" encoding="utf-8"/>
  </base:primaryData>
  <base:meta>
   <olac:olac xmlns:olac="http://www.language-archives.org/OLAC/1.0/"
    xmlns="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/">
   </olac:olac>
  </base:meta>
  <base:segments>
   <base:segment id="seg1" start="0" end="100"/>
   <base:segment id="seg2" start="100" end="110"/>
   <base:segment id="seg3" type="seg" segments="seg1 seg2"/>
  </base:segments>
 </base:corpusData>
</base:corpus>
```

...or by referring to already defined segments

SGF – Adding annotation layers

```
<base:corpus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.text-technology.de/sekimo root.xsd"
 xmlns:base="http://www.text-technology.de/sekimo"
 xmlns:doc="http://www.text-technology.de/sekimo/doc">
 <base:corpusData id="c1" type="text">
  <base:primaryData start="0" end="1036" xml:lang="de">
   <base:location uri="ex1.txt" encoding="utf-8"/>
  </base:primaryData>
  <base:meta>
   <olac:olac xmlns:olac="http://www.language-archives.org/OLAC/1.0/"
    xmlns="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms/">
   </olac:olac>
  </base:meta>
  <base:segments>
   <base:segment id="seg1" start="0" en
  </base:segments>
  <base:annotation>
   <base:level id="doc" priority="0">
    <base:layer xmlns:doc="http://www.text-technology.de/sekimo/doc">
     <doc:text base:segment="seg1">
      <doc:para base:segment="seg1"/>
     </doc:text>
    </base:layer>
   </base:level>
  </base:annotation>
 </base:corpusData>
</base:corpus>
```

Afterwards, an XML namespace and the annotation layer is added

SGF – Adding annotation layers

```
<base:corpus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.text-technology.de/sekimo root.xsd"
 xmlns:base="http://www.text-technology.de/sekimo"
 xmlns:doc="http://www.text-technology.de/sekimo/doc">
 <base:corpusData id="c1" type="text">
  <base:primaryData start="0" end="1036" xml:lang="de">
   <base:location uri="ex1.txt" encoding="utf-8"/>
  </base:primaryData>
  <base:meta>
   <olac:olac xmlns:olac="http://ww
    xmlns="http://purl.org/dc/eleme
    xmlns:dcterms="http://purl.org/
   </olac:olac>
  </base:meta>
  <base:segments>
   <base:segment id="seg1" start="0" end="100"/>
  </base:segments>
  <base:annotation>
   <base:level id="doc" priority="0">
    <base:layer xmlns:doc="http://www.text-technology.de/sekimo/doc">
     <doc:text base:segment="seg1">
      <doc:para base:segment="seg1"/>
     </doc:text>
    </base:layer>
   </base:level>
  </base:annotation>
 </base:corpusData>
</base:corpus>
```

The added `base:segment` attribute connect segments and annotation elements.
If a segment takes part in different annotation elements it will be defined once only

SGF – Adding annotation layers

```
<base:corpus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.text-technology.de/sekimo root.xsd"
 xmlns:base="http://www.text-technology.de/sekimo"
 xmlns:doc="http://www.text-technology.de/sekimo/doc">
 <base:corpusData id="c1" type="text">
  <base:primaryData sta
   <base:location uri="
  </base:primaryData>
  <base:meta>
   <olac:olac xmlns:ola
    xmlns="http://purl
    xmlns:dcterms="http
   </olac:olac>
  </base:meta>
  <base:segments>
   <base:segment id="s
  </base:segments>
  <base:annotation>
   <base:level id="doc
    <base:layer xmlns:doc="http://www.text-technology.de/sekimo/doc">
     <doc:text base:segment="seg1">
      <doc:para base:segment="seg1"/>
     </doc:text>
    </base:layer>
   </base:level>
  </base:annotation>
 </base:corpusData>
</base:corpus>
```

For each annotation layer the following holds:

- Declaration is done external (an XSD must be present) – the original document grammar is converted according to the following rules:
  - Mixed content elements are converted to container elements
  - PCDATA elements are converted to empty elements
  - The `base:segment` attribute is added to former non-empty elements
- Hierarchy of elements remains intact, no need for additional files – as long as tree structures are adequate
- All attributes remain intact

SGF – Sekimo Generic Format

As already shown SGF allows for two ways of defining segments:

1. By providing start and end positions

```
<base:segment id="seg1" start="0" end="100"/>
```

2. By referring to already defined segments (in this case the otherwise optional `type` attribute has to be set to the value 'seg')

```
<base:segment id="seg1" start="0" end="100"/>
<base:segment id="seg2" start="200" end="250"/>
<base:segment id="seg3" type="seg" segments="seg1 seg2"/>
```

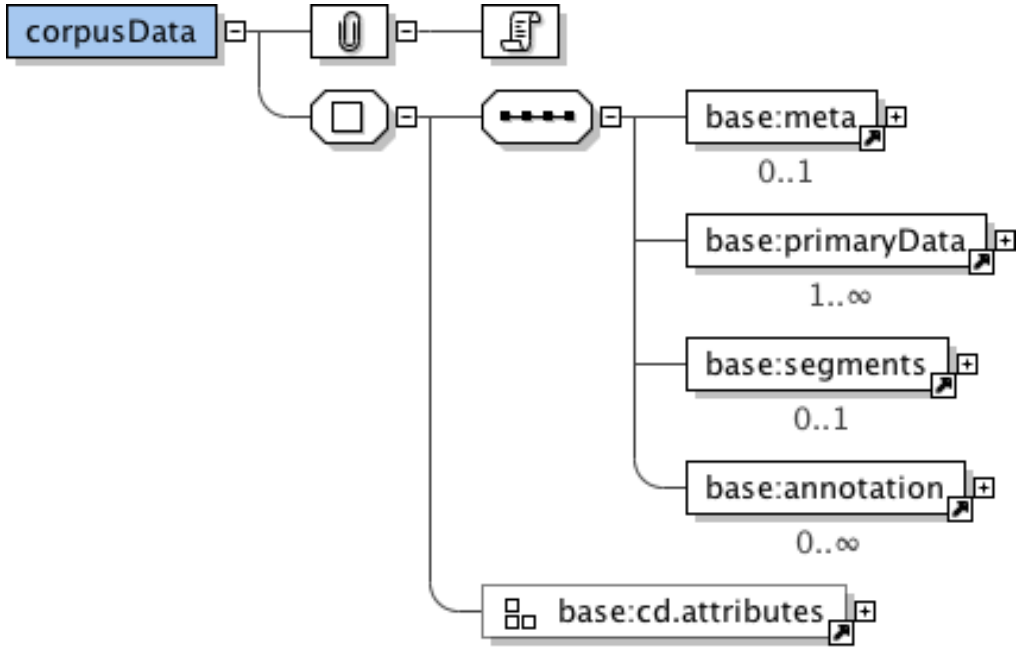
The `mode` attribute can be used to define continuous segments (default) or disjoint segments

```
<base:segment id="seg3" type="seg" segments="seg1 seg2" mode="continuous"/>
<base:segment id="seg4" type="seg" segments="seg1 seg2" mode="disjoint"/>
```

SGF – Sekimo Generic Format

Some further remarks:

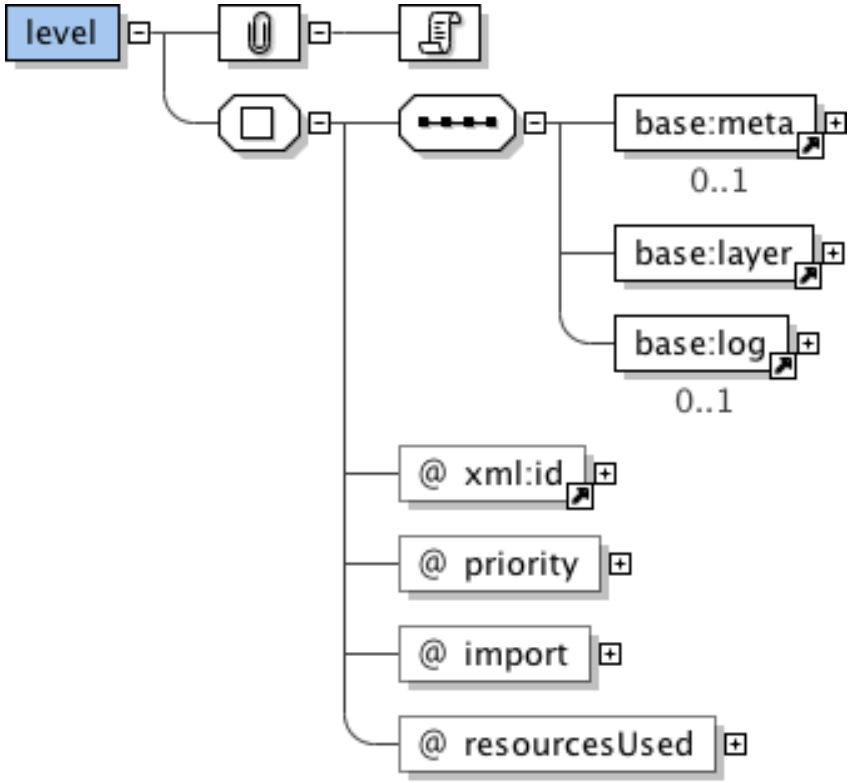
- A `base:corpusData` element can contain several `base:annotation` elements
- A `base:annotation` element can consist of several `base:level` elements.
  This mechanism can be used for subsuming annotation layers



SGF – Sekimo Generic Format

Some further remarks:

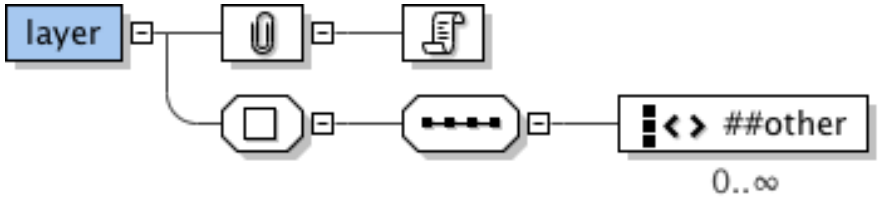
- A `base:level` element contains of optional metadata, a `base:layer` element and an optional log (used for the *Serengeti* web application)
- The optional `priority` attribute can be used in case of resulting overlapping annotation layers – the highest value determines the innermost annotation layer
- The optional `import` attribute can be used for layers the actual layer relies on



SGF – Sekimo Generic Format

Some further remarks:

- A `base:layer` element serves as a wrapper for annotations of a different namespace



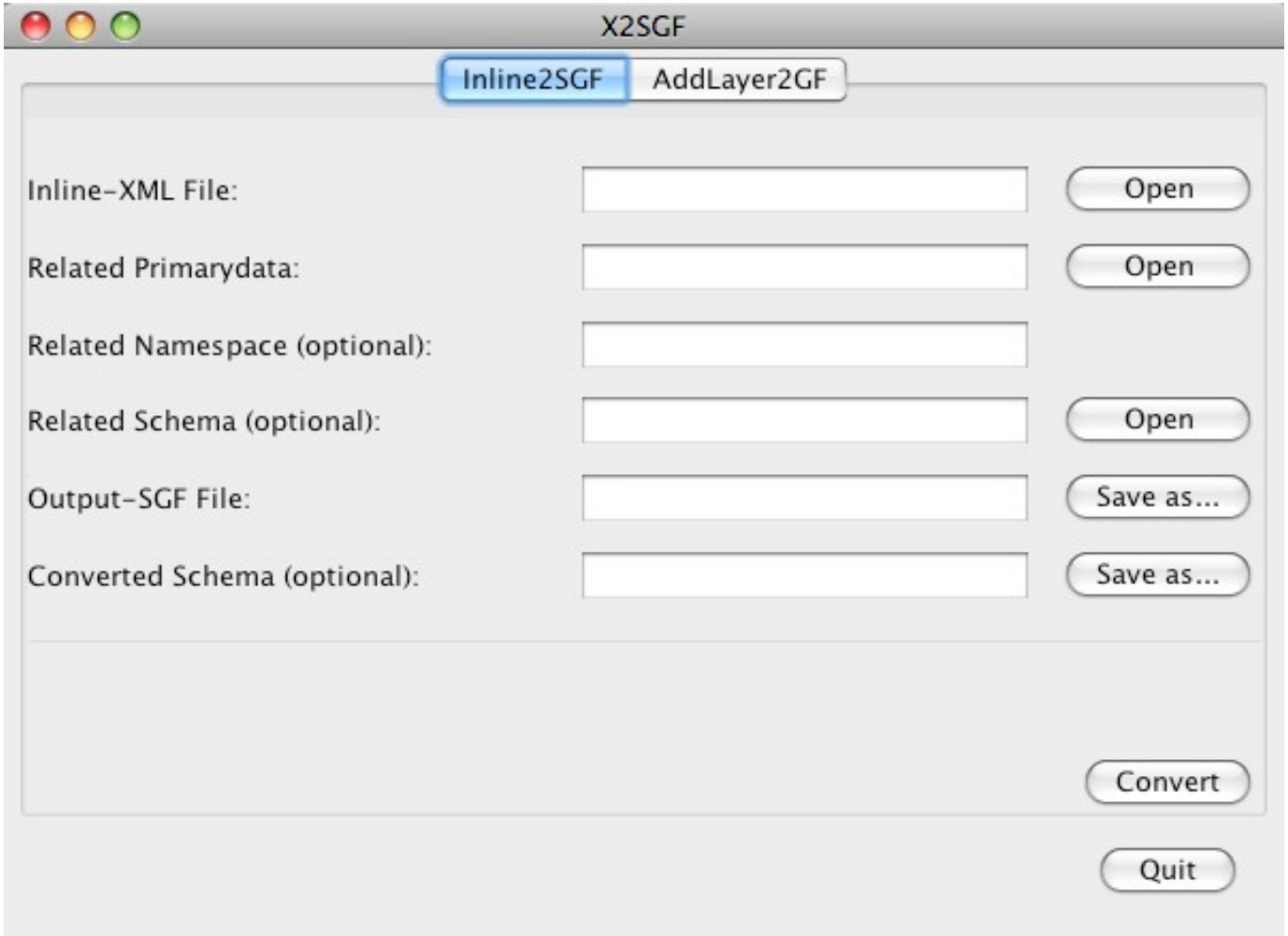
- Validation of annotation layers via XSD is mandatory
- Apart from XSD validation, embedded Schematron asserts are available as additional constraints
  - Desperately waiting for XML Schema 1.1

```
<xsd:element name="layer">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
    processContents="strict"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

SGF – Sekimo Generic Format

Some further remarks:

- There are two implementations for converting inline and standoff annotation into the new format (XSLT 2.0 and Java – shown below)
- Both implementations allow for converting a single inline annotation into an SGF file and merging multiple SGF files afterwards into a single SGF instance
- In addition the Java implementation allows for converting the XSD



SGF – Sekimo Generic Format

Some further remarks:

- The examples shown before use the XML inherent tree structure – however, as only the base layer is pre-declared, representing graph structures is possible as well
- The following DBMS were tested with small test sets:
  - eXist, Qizx/db 2.0, Oracle Berkeley DB XML (native)
  - IBM DB2 9.x/9.5 (hybrid)
  - MySQL (relational)
- For evaluation purposes Saxon-SA on a per-file-base was used

SGF – Sekimo Generic Format: Querying

Dipper et al. 2007 suggested the following XQuery queries for evaluating the PAULA architecture:

- Q1: "Find all sentences that include the word 'kam'"
- Q2: "Find all sentences that do not include the word 'kam'"
- Q3: "Find all NPs. Return the reference to that NP"
- Q7: "Find all pairs of anaphors and direct antecedents in which the anaphor is a personal pronoun"

- Q4 to Q6 were not used in our evaluation because the information queried is not available in the corpus under investigation.
  For testing a worst case scenario we have added the following query:
  Q8: "Find all pairs of anaphors and antecedents and their respective parent(s) on the logical document layer"

SGF – Sekimo Generic Format: Querying

Query test set:

- Largest single text, a German scientific article (157 paragraphs, 696 sentences, 12,345 token, 2,550 markables and 1,358 anaphoric relations, annotated on three annotation levels logical document structure, POS, anaphoric relations)
- Whole corpus (six German scientific articles and eight German newspaper articles, containing 3,084 sentences, 56,203 tokens, 11,740 markables, 4,323 anaphoric relations, three annotation levels)

Queries were tested on two machines:

- PC1: Sun Fire V20z (dual AMD Opteron 248@2,2 GHz; 6 GB RAM; Sun Solaris 10-64bit)
  XQuery: Saxon-SA 9.0.0.1J on Java 1.5.0_15 (2 GB RAM for Java VM)
  Prolog: SWI-Prolog 5.6.21 (128 MB as local stack limit)
- PC2: PC (Intel Core2Duo E6600@2,99 GHz; 3.12 GB RAM; MS Windows XP SP3-32bit)
  XQuery: Saxon-SA 9.0.0.1J on Java 1.6.0_06 (1 GB RAM for Java VM)
  Prolog: SWI-Prolog 5.6.57 (128 MB as local stack limit)

SGF – Sekimo Generic Format: Querying

Query results (after five executions):

- Q1: "Find all sentences that include the word 'kam'"
- Q2: "Find all sentences that do not include the word 'kam'"
- Q3: "Find all NPs. Return the reference to that NP"
- Q7: "Find all pairs of anaphors and direct antecedents in which the anaphor is a personal pronoun"
- Q8: "Find all pairs of anaphors and antecedents and their respective parent(s) on the logical document layer"

Query	Prolog query (text)		XQuery (text)		XQuery (corpus)	
	PC1	PC2	PC1	PC2	PC1	PC2
Q1	0.220	0.540	4.612	1.244	9.609	4.162
Q2	13.502	4.554	5.161	1.234	9.390	4.357
Q3	0.084	0.030	4.035	1.219	9.556	4.084
Q7	30.66	7.798	5.764	1.481	11.669	5.350
Q8	84.16	24.738	15.379	11.134	152.683	114.525

SGF – Sekimo Generic Format: Querying

Evaluation:

- Most of the example queries can be performed inline and are therefore rather quick (especially on a current standard PC)

Query	Prolog query (text)		XQuery (text)		XQuery (corpus)	
	PC1	PC2	PC1	PC2	PC1	PC2
Q1	0.220	0.540	4.612	1.244	9.609	4.162
Q2	13.502	4.554	5.161	1.234	9.390	4.357
Q3	0.084	0.030	4.035	1.219	9.556	4.084
Q7	30.66	7.798	5.764	1.481	11.669	5.350
Q8	84.16	24.738	15.379	11.134	152.683	114.525

SGF – Sekimo Generic Format: Querying

Evaluation:

- Most of the example queries can be performed inline and are therefore rather quick (especially on a current standard PC)
- If information derived from different levels is needed, usage of the id() function is still fast enough – and outperforms the Prolog fact base (Q7)

Query	Prolog query (text)		XQuery (text)		XQuery (corpus)	
	PC1	PC2	PC1	PC2	PC1	PC2
Q1	0.220	0.540	4.612	1.244	9.609	4.162
Q2	13.502	4.554	5.161	1.234	9.390	4.357
Q3	0.084	0.030	4.035	1.219	9.556	4.084
Q7	30.66	7.798	5.764	1.481	11.669	5.350
Q8	84.16	24.738	15.379	11.134	152.683	114.525

SGF – Sekimo Generic Format: Querying

Evaluation:

- Most of the example queries can be performed inline and are therefore rather quick (especially on a current standard PC)
- If information derived from different levels is needed, usage of the id() function is still fast enough – and outperforms the Prolog fact base (Q7)
- SGF performs worse if the query processor has to go down to the `base:segment` elements (and up afterwards using the idref() function) – but still outperforms the Prolog fact base (Q8)

Query	Prolog query (text)		XQuery (text)		XQuery (corpus)	
	PC1	PC2	PC1	PC2	PC1	PC2
Q1	0.220	0.540	4.612	1.244	9.609	4.162
Q2	13.502	4.554	5.161	1.234	9.390	4.357
Q3	0.084	0.030	4.035	1.219	9.556	4.084
Q7	30.66	7.798	5.764	1.481	11.669	5.350
Q8	84.16	24.738	15.379	11.134	152.683	114.525

Overview

- Architectures for multiple annotations
- SGF – The Sekimo Generic Format
- Real World Application Scenarios

Overview

- Architectures for multiple annotations
- SGF – The Sekimo Generic Format
- Real World Application Scenarios

Real World Application Scenarios

SGF is used in different application scenarios:

- For analyzing anaphoric relations
- As exchange format for the web-based annotation tool *Serengeti*
- As exchange format for the web-based lexical chaining tool Scientific Workplace (SGF-LC)

Application of SGF – Analyzing anaphoric relations

- Anaphora resolution is the process of identifying the antecedent(s) in the previous context
- Anaphora resolution requires a variety of knowledge:
  - POS, morphology, grammatical function, semantic relatedness, distance, discourse structure among others
- The anaphora resolution process consists of several steps:
  - Detection of anaphoric elements
  - Creation of a candidate list for each anaphor
  - Filtering of candidate list
  - Identification of correct antecedent

Application of SGF – Analyzing anaphoric relations: Example

Lurup ist ein sozialer Brennpunkt der Hansestadt, ein Vorort mit Einzelhäusern, aber auch vielen Wohnblocks im Westen der Stadt.

Application of SGF – Analyzing anaphoric relations: Example

Lurup is a social ghetto of the hanseatic city (Hansestadt), an outskirt with single unit houses but also many apartment blocks in the west of the city (Stadt).

Application of SGF – Analyzing anaphoric relations: Example

Lurup is a social ghetto of the hanseatic city (Hansestadt), an outskirt with single unit houses but also many apartment blocks in the west of the city (Stadt).

```
<base:corpus>
 <base:corpusData id="c1" type="text">
  [...]
  <base:annotation>
   <base:level xml:id="de" priority="0">
    <base:layer xmlns:chs="http://www.text-technology.de/sekimo/chs">
     <chs:de base:segment="to827" deID="de226" headRef="w827" deType="nom"/>
     [...]
     <chs:de base:segment="to848" deID="de231" headRef="w848" deType="nom"/>
    </base:layer>
   <base:level>
   <base:level xml:id="semRel" priority="0">
    <base:layer xmlns:chs="http://www.text-technology.de/sekimo/chs">
     <chs:semRel>
      <chs:cospecLink relType="hyperonym" phorIDRef="de231"
       antecedentIDRefs="de226"/>
     </chs:semRel>
    </base:layer>
   </base:level>
  </base:annotation>
 </base:corpusData>
</base:corpus>
```

Application of SGF – Analyzing anaphoric relations: Example

Lurup is a social ghetto of the hanseatic city (Hansestadt), an outskirt with single unit houses but also many apartment blocks in the west of the city (Stadt).

```
<base:corpus>
 <base:corpusData id="c1" type="text">
  [...]
  <base:annotation>
   <base:level xml:id="de" priority="0">
    <base:layer xmlns:chs="http://www.text-technology.de/sekimo/chs">
     <chs:de base:segment="to827" deID="de226" headRef="w827" deType="nom"/>
     [...]
     <chs:de base:segment="to848" deID="de231" headRef="w848" deType="nom"/>
    </base:layer>
   <base:level>
   <base:level xml:id="semRel" priority="0">
    <base:layer xmlns:chs="http://www.text-technology.de/sekimo/chs">
     <chs:semRel>
      <chs:cospecLink relType="hyperonym" phorIDRef="de231"
       antecedentIDRefs="de226"/>
     </chs:semRel>
    </base:layer>
   </base:level>
  </base:annotation>
 </base:corpusData>
</base:corpus>
```

Application of SGF – Analyzing anaphoric relations: Example

Lurup is a social ghetto of the hanseatic city (Hansestadt), an outskirt with single unit houses but also many apartment blocks in the west of the city (Stadt).

```
<base:corpus>
 <base:corpusData id="c1" type="text">
  <candidateList maxDeDistance="15" filename="zeit-15-2005-sgf.xml">
   <semRel relationID="sr86" type="cospecLink" subtype="hyperonym"
     phorIDRef="de231" antecedentIDRefs="de226">
    <anaphor base:segment="seg1620" deID="de231" headRef="w848" deType="nom"
     text="Stadt" pos="N" syntax="@NH" lemma="stadt" dependValue="mod"
     start="4557" end="4566" npType="defNP" position="195" num="SG" gen="FEM"
     cas="GEN" sentencePos="8/8" paraPos="10/26" sentenceParaPos="2/7"
     docPath="doc:text[1]/doc:para[13]">der Stadt</anaphor>
    <!-- [...] -->
    <antecedentCandidate correctAntecedent="yes" base:segment="seg1589"
     deID="de226" headRef="w833" deType="nom" text="Hansestadt" pos="N"
     syntax="@NH" lemma="hanse#stadt" dependValue="mod" start="4473" end="4487"
     npType="defNP" position="190" num="SG" gen="FEM" cas="GEN"
     sentencePos="3/8" paraPos="5/26" sentenceParaPos="2/7"
     docPath="doc:text[1]/doc:para[13]" deDistance="5"
     anteHierarchy="siblings">der Hansestadt</antecedentCandidate>
    <antecedentCandidate base:segment="seg1615" deID="de230" headRef="w846"
     deType="nom" text="Westen" pos="N" syntax="@NH" lemma="westen" start="4548"
     end="4556" npType="defNP" position="194" num="SG" gen="MSC" cas="DAT"
     sentencePos="7/8" paraPos="9/26" sentenceParaPos="2/7"
     docPath="doc:text[1]/doc:para[13]" deDistance="1"
     anteHierarchy="siblings">m Westen</antecedentCandidate>
   </semRel>
  </candidateList>
```

Application of SGF – Analyzing anaphoric relations: Example

Lurup is a social ghetto of the hanseatic city (Hansestadt), an outskirt with single unit houses but also many apartment blocks in the west of the city (Stadt).

```
<base:corpus>
 <base:corpusData id="c1" type="text">
  <candidateList maxDeDistance="15" filename="zeit-15-2005-sgf.xml">
   <semRel relationID="sr86" type="cospecLink" subtype="hyperonym"
     phorIDRef="de231" antecedentIDRefs="de226">
    <anaphor base:segment="seg1620" deID="de231" headRef="w848" deType="nom"
     text="Stadt" pos="N" syntax="@NH" lemma="stadt" dependValue="mod"
     start="4557" end="4566" npType="defNP" position="195" num="SG" gen="FEM"
     cas="GEN" sentencePos="8/8" paraPos="10/26" sentenceParaPos="2/7"
     docPath="doc:text[1]/doc:para[13]">der Stadt</anaphor>
    <!-- [...] -->
    <antecedentCandidate correctAntecedent="yes" base:s
     deID="de226" headRef="w833" deType="nom" text="Han
     syntax="@NH" lemma="hanse#stadt" dependValue="mod"
     npType="defNP" position="190" num="SG" gen="FEM" c
     sentencePos="3/8" paraPos="5/26" sentenceParaPos="
     docPath="doc:text[1]/doc:para[13]" deDistance="5"
     anteHierarchy="siblings">der Hansestadt</anteceden
    <antecedentCandidate base:segment="seg1615" deID="d
     deType="nom" text="Westen" pos="N" syntax="@NH" le
     end="4556" npType="defNP" position="194" num="SG"
     sentencePos="7/8" paraPos="9/26" sentenceParaPos="
     docPath="doc:text[1]/doc:para[13]" deDistance="1"
     anteHierarchy="siblings">m Westen</antecedentCandi
   </semRel>
  </candidateList>
```

```
@data
no,10,5,1,yes,yes,yes,no
no,9,5,2,no,no,yes,no,no
no,8,5,3,no,yes,yes,no,no
no,7,5,4,no,no,yes,no,yes
no,5,5,2,no,yes,yes,no,no
no,4,5,1,yes,yes,yes,no,no
no,3,5,2,no,yes,yes,no,no
no,2,5,3,no,no,no,no,no

no,51,7,5,no,no,yes,yes,yes
no,10,7,3,no,no,no,no,no
no,8,7,5,no,yes,no,yes,no
...
```

Application of SGF – Analyzing anaphoric relations: Example

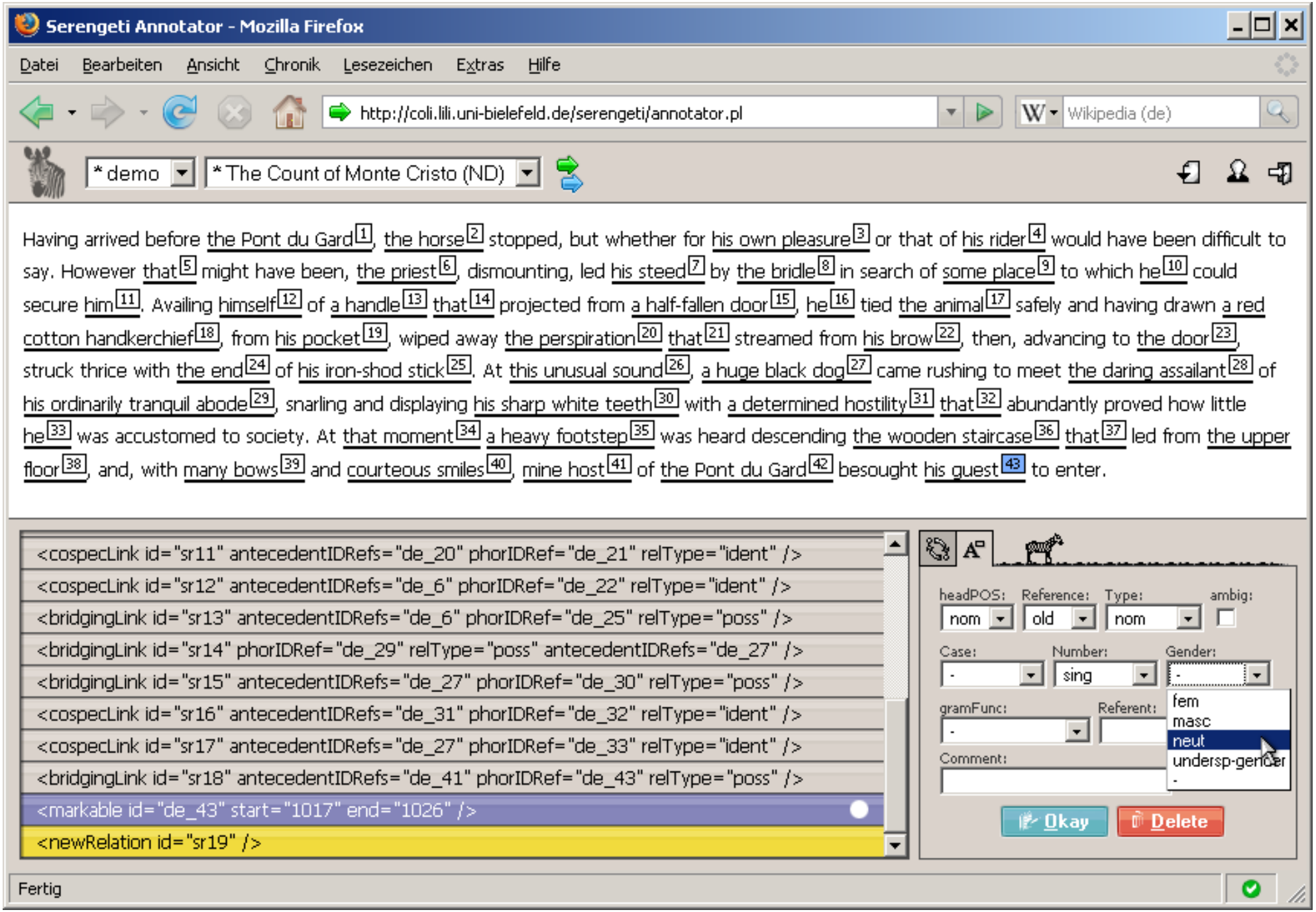
Lurup is a social ghetto of the hanseatic city (Hansestadt), an outskirt with single unit houses but also many apartment blocks in the west of the city (Stadt).

```
<base:corpus>
 <base:corpusData id="c1" type="text">
  <candidateList maxDeDistance="15" filename="zeit-15-2005-sgf.xml">
   <semRel relationID="sr86" type="cospecLink" subtype="hyperonym"
     phorIDRef="de231" antecedentIDRefs="de226">
    <anaphor base:segment="seg1620" deID="de231" headRef="w848" deType="nom"
     text="Stadt" pos="N" syntax="@NH" lemma="stadt" dependValue="mod"
     start="4557" end="4566" npType="defNP" position="195" num="SG" gen="FEM"
     cas="GEN" sentencePos="8/8" paraPos="10/26" sentenceParaPos="2/7"
     docPath="doc:text[1]/doc:para[13]">der Stadt</anaphor>
    <!-- [...] -->
    <antecedentCandidate correctAntecedent="yes" base:s
     deID="de226" headRef="w833" deType="nom" text="Han
     syntax="@NH" lemma="hanse#stadt" dependValue="mod"
     npType="defNP" position="190" num="SG" gen="FEM" c
     sentencePos="3/8" paraPos="5/26" sentenceParaPos="
     docPath="doc:text[1]/doc:para[13]" deDistance="5"
     anteHierarchy="siblings">der Hansestadt</anteceden
    <antecedentCandidate base:segment="seg1615" deID="d
     deType="nom" text="Westen" pos="N" syntax="@NH" le
     end="4556" npType="defNP" position="194" num="SG"
     sentencePos="7/8" paraPos="9/26" sentenceParaPos="
     docPath="doc:text[1]/doc:para[13]" deDistance="1"
     anteHierarchy="siblings">m Westen</antecedentCandi
   </semRel>
  </candidateList>
```

```
@data
no,10,5,1,yes,yes,yes,no
no,9,5,2,no,no,yes,no,no
no,8,5,3,no,yes,yes,no,no
no,7,5,4,no,no,yes,no,yes
no,5,5,2,no,yes,yes,no,no
no,4,5,1,yes,yes,yes,no,no
no,3,5,2,no,yes,yes,no,no
no,2,5,3,no,no,no,no,no

no,51,7,5,no,no,yes,yes,yes
no,10,7,3,no,no,no,no,no
no,8,7,5,no,yes,no,yes,no
...
```

Application of SGF – Exchange format for *Serengeti*



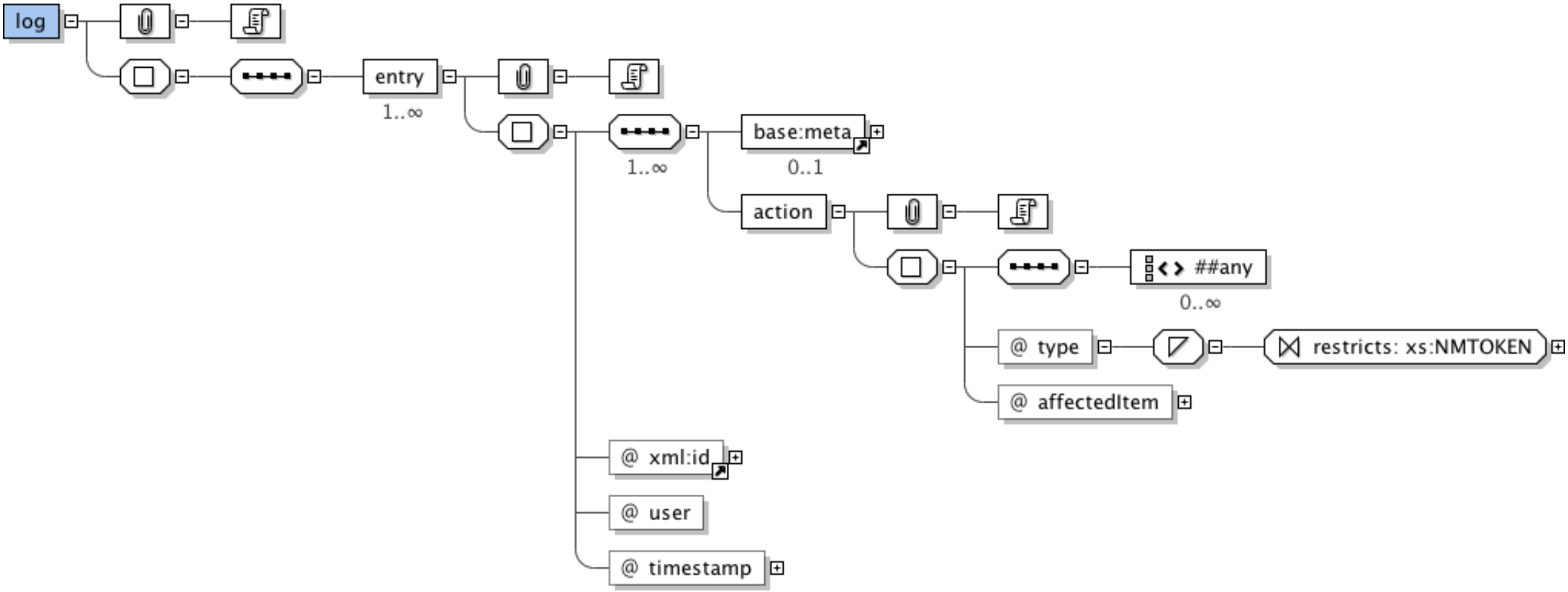
Application of SGF – Exchange format for *Serengeti*

- *Serengeti* is a web-based annotation tool developed in the Sekimo project using a MySQL database and AJAX
- It supports multi-user annotation, central file and user management and import & export capabilities
- While the first version was used exclusively in our project (cf. Stührenberg et al. 2007) the new version will be employed as expert annotation tool in the AnaWiki/AnaphoricBank by Massimo Poesio
- The upcoming new version will support different annotation schemas, and manipulation of markables and uses SGF for import and export – in return SGF's log functionality was inspired by *Serengeti*

Application of SGF – Exchange format for *Serengeti*

SGF's log functionality

- The optional log is used in *Serengeti* for saving the history of actions regarding a specific SGF file
- It consists of at least one `entry` element containing `action` wrapper elements



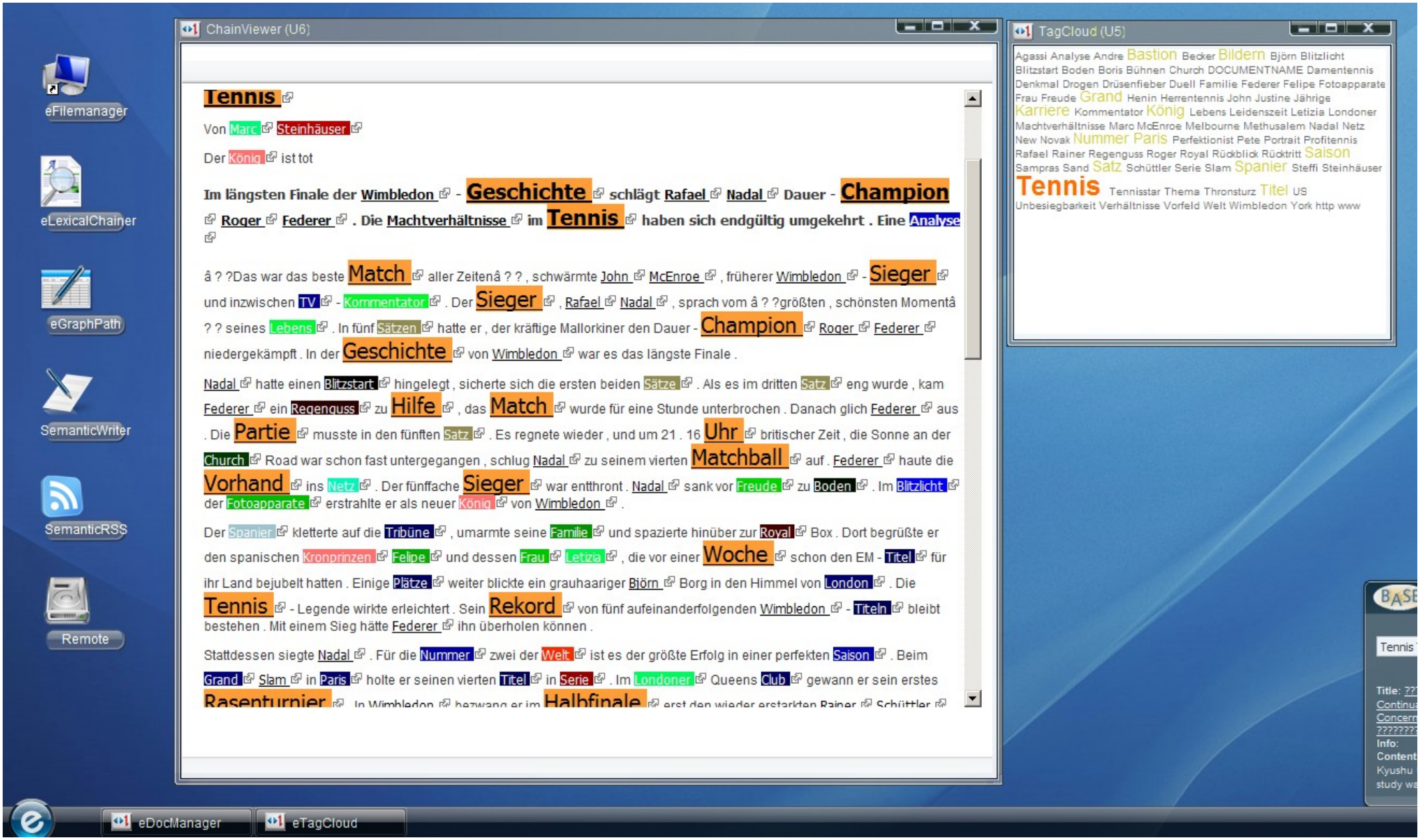
Application of SGF – Exchange format for *Serengeti*

SGF's log functionality

- The `action` element makes use of XML Schemas possibility to skip the content of the wrapper element

```
<xs:element name="action">
 <xs:complexType>
  <xs:sequence>
   <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
     processContents="skip"/>
  </xs:sequence>
  <xs:attribute name="type" use="required">
   <xs:simpleType>
    <xs:restriction base="xs:NMTOKEN">
     <xs:enumeration value="add"/>
     <xs:enumeration value="delete"/>
     <xs:enumeration value="modify"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="affectedItem" type="xs:IDREF" use="optional"/>
 </xs:complexType>
</xs:element>
```

Application of SGF – Exchange format for lexical chaining (SGF-LC)



Application of SGF – Exchange format for lexical chaining (SGF-LC)

- The *Scientific Workplace*/*Scientific Desktop* tool has been developed in the Indogram project of our research group
- Together with other members of the research group an annotation format was established that is imported into the SGF base layer
- In addition SGF-LC supports the storage of linguistic resources (either together in a single file with the other SGF data or as a reference to external resources)

Application of SGF – Exchange format for lexical chaining (SGF-LC)

```
<base:corpus>
 <base:corpusData id="c1" type="text">
  [...]
   <base:segments>
    <base:segment id="i1" type="char" start="1" end="5" />
    <base:segment id="i2" type="char" start="6" end="10" />
   </base:segments>
   <base:annotation>
    <base:level xml:id="lc" resourcesUsed="gn wiki">
     <base:layer xmlns:lc="http://www.text-technology.de/lc">
      <lc:link from="i5" to="i1" relation="hyperonymy" relationDirection="l"
       distance="0.848754899874578" distanceType="WordnetVektor-UMN"
       resourceUsed="gn"/>
      <lc:link from="i5" to="i3" relation="transitive" relationDirection="r"
       distance="-0.434" distanceType="NSS-Gwikipedia" resourceUsed="wiki"/>
      <lc:chains>
       <lc:chain id="lc_ch1" label="pflanzen">
        <lc:ref base:segment="i1"/>
        <lc:ref base:segment="i3"/>
        <lc:ref base:segment="i5"/>
       </lc:chain>
      </lc:chains>
     </base:layer>
    </base:level>
   </base:annotation>
  </base:corpusData>
</base:corpus>
```

Conclusion

To sum up:

- We've presented SGF – The Sekimo Generic Format – capable of storing multiple annotated primary data
- SGF is XML-based and uses the Annotation Graph model
- SGF combines inline and standoff annotation and allows for validating multiple annotated data
- SGF can be queried and processed by using standard XPath or XQuery – performance depends on the imported annotation layers
- Apart from our project SGF is used in other projects as well – and will be made freely available to other interested parties

Last but not least...

Thank your for your attention!

Visit us at `http://www.text-technology.de/Sekimo`

`maik.stuehrenberg@uni-bielefeld.de`