Using XSLT for the Integration of Deep and Shallow Natural Language Processing Components

Ulrich Schäfer
Language Technology Lab,
German Research Center for Artificial Intelligence (DFKI),
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
email: ulrich.schaefer@dfki.de

Abstract

WHITEBOARD is a hybrid XML-based architecture that integrates deep and shallow natural language processing components. The online system consists of a fast HPSG parser that utilizes tokenization, PoS, morphology, lexical, named entity, phrase chunk and (for German) topological sentence field analyses from shallow components. This integration increases robustness, directs the search space and hence reduces processing time of the deep parser. In this paper, we focus on one of the central integration facilities, the XSLT-based WHITEBOARD Annotation Transformer (WHAT), report on the benefits of XSLT-based NLP component integration, and present examples of XSL transformation of shallow and deep annotations used in the integrated architecture. Furthermore, we report on a recent application of XSL transformation for the conversion of XML-encoded typed feature structures representation in the context of the DEEPTHOUGHT project where deep-shallow integration is performed on the basis of Robust Minimal Recursion Semantics (RMRS).

1 Introduction

Deep processing (DNLP) systems try to apply as much linguistic knowledge as possible during the analysis of sentences and result in a uniformly represented collection of the knowledge that contributed to the analysis. The result often consists of many possible analyses per sentence reflecting the uncertainty which of the possible readings was intended - or no answer at all if the linguistic knowledge was contradictory or insufficient with respect to the input sentence.

Shallow processing (SNLP) systems do not attempt to achieve such an exhaustive linguistic analysis. They are desigend for specific tasks ignoring many details in input and linguistic framework. Utilizing rule-based (*e.g.* finite-state) or statistics-based approaches, they are in general much faster than DNLP. Due to the lack of efficiency and robustness of DNLP systems, the trend in application-oriented language processing system development in the last years was to improve SNLP systems. They are now capable of analyzing Megabytes of texts within seconds, but precision and quality barriers are so obvious (especially on domains the systems were not designed for or trained on) that a need for 'deeper' systems re-emerged. Moreover, semantics construction from an input sentence is quite poor and erroneous in typical shallow systems.

However, development of DNLP made advances during the last few years, especially in the field of efficiency [Callmeier, 2000; Uszkoreit, 2002]. A promising solution to improve quality of natural language processing is the combination of deep and shallow technologies. Deep processing benefits

from specialized and fast shallow analysis results, shallow processing becomes 'deeper' using at least partial results from DNLP.

Many natural language processing applications could benefit from the synergy of the combination of deep and shallow, *e.g.* advanced information extraction, question answering, or grammar checking systems.

This paper is structured as follows. In section 2, we motivate and introduce the XSLT-based WHITEBOARD annotation transformer. In section 3, we describe the deep-shallow integration. In section 4, we focus on transformation of feature structure XML transformation. Section 5 describes related work, we conclude and give an outlook to future work in section 6.

2 Annotation Access and Transformation through XSLT

2.1 Motivation

The WHITEBOARD architecture integrates shallow and deep natural language processing components. Both online and offline coupling of existing software modules is supported, *i.e.*, the architecture provides direct access to standoff XML annotation as well as programming interfaces. Applications communicate with the components through programming interfaces. A multi-layer chart holds the linguistic processing results in the online system memory while XML annotations can be accessed online as well as offline. The different paradigms of DNLP and SNLP are preserved throughout the architecture, *e.g.* there is a shallow and a deep programming interface.

WHAM (WHITEBOARD Annotation Machine) offers programming interfaces which are not simply DOM interfaces isomorphic to the XML markup they are based on, but hierarchically defined classes. E.g., a fast index-sequential storage and retrieval mechanism based on XML is encapsulated through the shallow programming interface. However, while the typed feature structure-based programming interface to deep components became stable early, it turned out that the initial, XML interface was too inflexible when new, mainly shallow, components with new DTDs had to be integrated. Therefore, a more flexible approach had to be devised.

The solution is an XSLT-based infrastructure for NLP components that provides flexible access to standoff XML annotations produced by the components. XSLT [Clark, 1999] is a W3C standard language for the transformation of XML documents. Input of an XSL transformation must be XML, while output can be any syntax (*e.g.* XML, text, HTML, RTF, or even programming language source code, *etc*). The power of XSLT mainly comes from its sublanguage XPath [Clark and DeRose, 1999], which supports access to XML structure, elements, attributes and text through concise path expressions. An XSL stylesheet consists of templates with XPath expressions that must match the input document in order to be executed. The order in which templates are called is by default top-down, left to right, but can be modified, augmented, or suppressed through loops, conditionals, and recursive call of (named) templates.

2.2 WHAT, the WHITEBOARD Annotation Transformer

WHAT is built on top of a standard XSL transformation engine. It provides uniform access to standoff annotation through queries that can either be used from non-XML aware components to get access to information stored in the annotation (V and N-queries), or to transform (modify, enrich, merge) XML annotation documents (D-queries). While WHAT is written in a programming language such as Java

or C, the XSL query templates that are specific for a standoff DTD of a component's XML output are independent of that programming language, *i.e.*, they must only be written once for a new component and are collected in a so-called template library.

Based on an input XML document (or DOM object), a WHAT query that consists of component name, query name, and query-specific parameters such as an index or identifier is looked up in the XSLT template library for the specified component, an XSLT stylesheet is returned and applied to the XML document by the XSLT processor. The result of stylesheet application is then returned as the answer to the WHAT query. There are basically three kinds of results: (1) strings (including non-XML output), (2) references to nodes in the XML input document (IDs), (3) XML documents.

In other words, if we formulate queries as functions, we get the following query signatures, with C being the component, D denoting an XML document, P^* a (possibly empty) sequence of parameters, S^* a sequence of strings, and N^* a sequence of nodes.

- **V-queries**. getValue: $C \times D \times P^* \longmapsto S^*$
  V-queries return string values from XML attribute values or text. The simplest case is a single XPath lookup, *e.g.* of the gender of a word encoded in a shallow XML annotation.

- **N-queries**. getNodes: $C \times D \times P^* \longmapsto N^*$
  N-queries compute and return lists of node identifiers that can again be used as parameters for subsequent queries, *e.g.* all named entity nodes within a token or character range specified as query parameters.

- **D-queries**. getDocument: $C \times D \times P^* \longmapsto D$
  D-queries return transformed XML documents – this is the classical, general use of XSLT. Complex transformations that modify, enrich or produce (standoff) annotation can be used for many purposes. Examples are (1) conversion from a different XML format, (2) merging of several XML documents into one, (3) auxiliary document modifications, *e.g.* to add unique identifiers to elements, sort elements *etc*, (4) providing interface to NLP applications, (5) visualization and formatting (Thistle, HTML, PDF, ...) (6) complex computations on XML input (turning a query into a kind of NLP component itself).

More details on WHAT and examples of the query types are presented in [Schäfer, 2003].

3 Architecture of the Hybrid Deep-Shallow System

The fully online-integrated hybrid WHITEBOARD architecture consists of the efficient deep HPSG parser PET [Callmeier, 2000] utilizing tokenization, PoS, morphology, lexical, compound, named entity, phrase chunk and topological sentence field analyses from shallow components in a pipelined architecture. This integration significantly increases robustness, directs the search space and reduces processing time of the deep parser, cf. [Crysmann *et al.*, 2002; Frank *et al.*, 2003; Schäfer, 2003] for details and evaluation results.

The simplified diagram in Fig. 1 depicts the components and places where WHAT comes into play in the hybrid integration of deep and shallow processing components (V, N, D denote the WHAT query types, *i.e.* XSLT transformations). Solid boxes depict components that produce annotation, dashed boxes depict the produced annotation (XML markup). Solid-line arrows depict the transformations used in the online integration. Dashed-line arrows indicate possible access to the intermediate annotation that could be accessed from an application (bottom box), *e.g.* the Thistle [Calder, 2000]

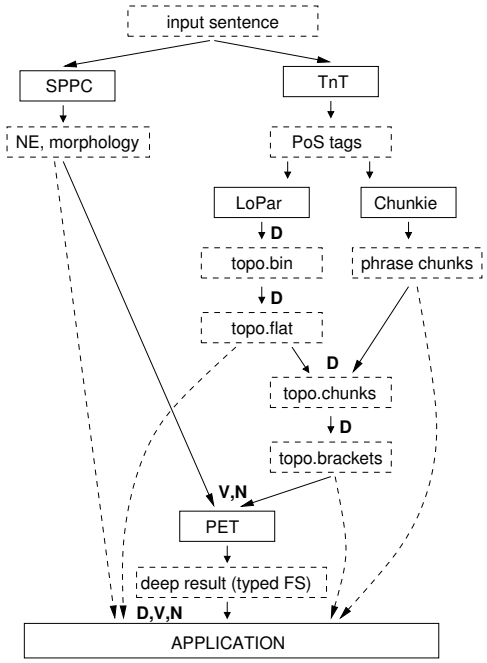

Figure 1: XSLT-based architecture of the hybrid parser.

tree visualizations that show the XML annotation tree structures (Fig. 2 through 4) have been created through WHAT D-queries out of the intermediate topo.* XML trees.

The system takes an input sentence, and runs three shallow systems on it:

- the rule-based shallow SPPC [Piskorski and Neumann, 2000] for named entity recognition, compound analysis for German, and morphology and stemming of words unknown to the HPSG lexicon,
- TnT/Chunkie, a statistics-based shallow PoS tagger and chunker [Skut and Brants, 1998],
- LoPar, a probabilistic context-free parser [Schmid, 2000], which takes PoS-tagged tokens as input, and produces binary tree representations of sentence fields, *e.g.* topo.bin in Fig. 2. For a motivation for binary vs. flat trees cf. [Becker and Frank, 2002].

The results of these components are three standoff annotations of the input sentence. Named entity, compound, morphological and stem information from SPPC is used by the deep parser to initialize the chart with prototypical feature structures that are filled with shallow information through V-queries for words unknown to the HPSG lexicon and for named entities. In addition, preference information on part-of-speech is used for prioritization of the deep parser. For details, cf. [Crysmann *et al.*, 2002]. PoS tagging from TnT is used as input for Chunkie to produce chunking and as input for the shallow topological PCFG parser [Frank *et al.*, 2003].

The examples in Fig. 2 through 5 show the analyses of the German sentence

> *Untergebracht war die Garnison in den beiden Wachlokalen Hauptwache und Konstablerwache. (Located was the garrison at the two guard houses, the main guard house and the Constabler guard house.)*

with a fronted verb in topic position, which the topological parser identifies correctly. This macro-sentential information can be used to direct the deep parser's search space towards the correct (and

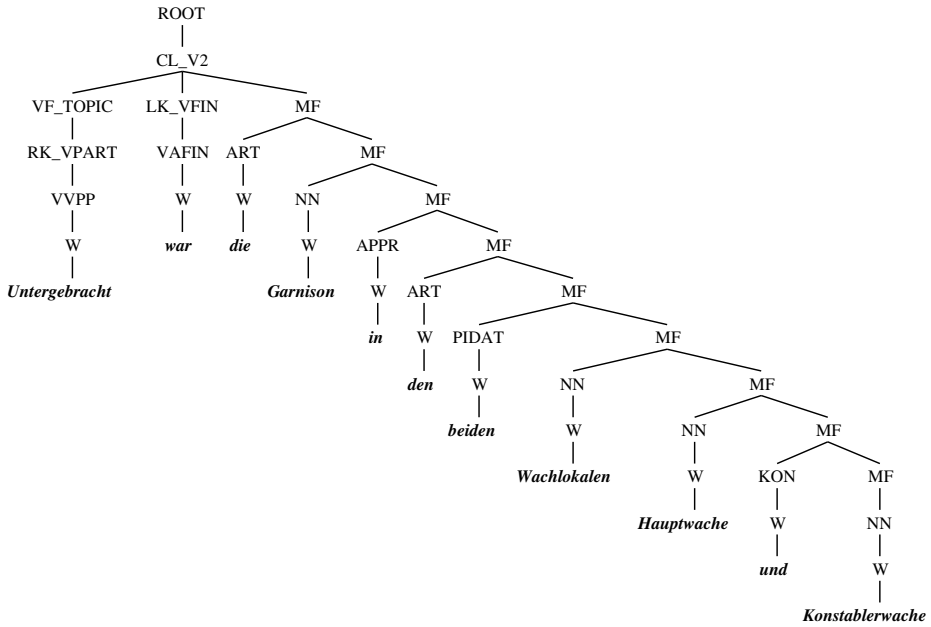

Figure 2: A binary tree as the result of the topological parser (topo.bin).

rather infrequent) construction, avoiding alternative exploration of the search space.

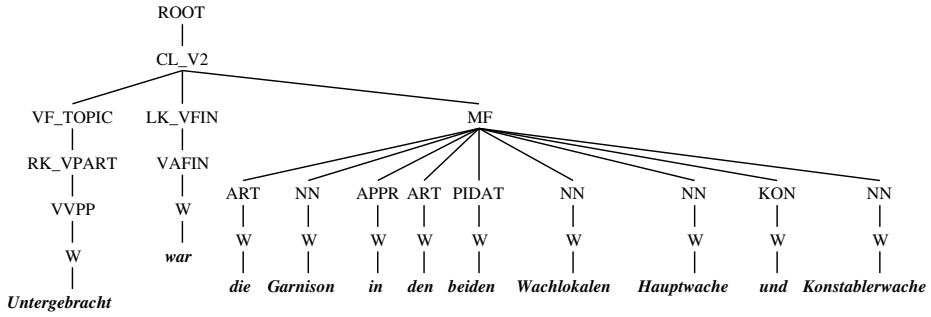

Figure 3: The topological tree after flattening (topo.flat).

In order to extract this type of global constituent-based information, a sequence of D-queries is applied to flatten the binary topological trees that are output by LoPar (result is topo.flat, Fig. 3) and merge the tree with shallow chunk information from Chunkie (topo.chunks, Fig. 4). In a next step, we apply the main D-query which computes bracket information for the deep parser from the merged topological tree and chunks (topo.brackets, Fig. 5).

The bracket information is used to prioritize chart elements of the deep parser that match the constituent boundaries computed by the shallow parser and chunker. The stylesheet directly generates the names of the appropriate HPSG types (value of the rule attributes in Fig. 5)[1].

Finally, the deep parser PET [Callmeier, 2000], modified as described in [Frank *et al.*, 2003], is started with a chart initialized using V-queries to access lexical (morphology, stemming, compounds, PoS preferences) and named entity information gathered from SPPC. The computed bracket information (topo.brackets; Fig. 5) is accessed through WHAT V and N-queries during parsing in order to prioritize

[1]The tree visualizations in Fig. 3 through 5 themselves have been generated through a generic WHAT D-query transforming the XML document into a Thistle tree (arbora DTD; [Calder, 2000]).

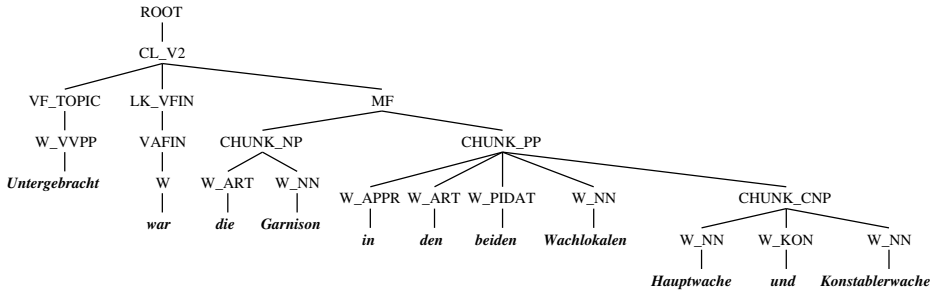

Figure 4: The topological tree merged with chunks (topo.chunks).

```
<TOPO2HPSG>
  <MAPC type="chunk_np+det" rule="chunk" left="W2" right="W3"/>
  <MAPC type="chunk_pp" rule="chunk" left="W4" right="W10"/>
  <MAPC type="v2_cp" rule="vfronted" left="W0" right="W10"/>
  <MAPC type="vfronted_vfin-rk" rule="vfronted" left="W1" right="W1"/>
  <MAPC type="vfronted_vfin+vp-rk" rule="vfronted" left="W1" right="W10"/>
  <MAPC type="v2_vf" rule="vfronted" left="W0" right="W0"/>
  <MAPC type="v2_vfin_pvp-rk" rule="vfronted" left="W1" right="W1"/>
</TOPO2HPSG>
```

Figure 5: The extracted brackets (topo.brackets).

constituent analyses motivated by the topological parser and additional syntactic information. Again, WHAT abstraction facilitates exchange of the shallow input components of PET, *e.g.* it would be possible to exchange some of the used components without rewriting of the deep parser's code.

The result of deep parsing including the constructed semantics representation of the analysed sentence can be accessed through the chart interface of PET, or through XSLT transformation – which leads to the next topic, WHAT access to and transformation of typed feature structure markup.

4 Accessing and Transforming Typed Feature Structure Markup

In the sections so far, we have shown examples for shallow XML annotation. But annotation access could also include deep analysis results. In this section, we turn to deep XML annotation. Typed feature structures provide a powerful, universal representation for deep linguistic knowledge.

While it is in general inefficient to use XML markup to represent typed feature structures during processing (*e.g.* for unification, subsumption operations), there are several applications that may benefit from a standardized system-independent XML markup of typed feature structures, *e.g.* as exchange format for (1) deep NLP component results (*e.g.* parser chart), (2) grammar definitions like in the SProUT system [Drozdzynski *et al.*, 2004], (3) feature structure renderers or editors like in SProUT or Thistle [Calder, 2000], (4) feature structure 'tree banks' of analyzed corpora.

We adopt a DAG-like XML markup for embedded typed feature structures originally developed by the Text Encoding Initiative (TEI) which is compact and widely accepted (cf. [Lee *et al.*, 2004]). An in-depth justification for the naming and structure of the TEI feature structure DTD is presented in [Langendoen and Simons, 1995]. We focus here on the feature structure DTD subset that is able to encode the basic data structures of deep systems such as LKB [Copestake, 2002], PET [Callmeier, 2000], PAGE, or the shallow system SProUT [Drozdzynski *et al.*, 2004] which have a subset of TDL [Krieger and Schäfer, 1994] as their common basic formalism.

```
<MATCHINFO rule="en_city" cstart="3" cend="7">
  <FS type="sprout_rule">
    <F name="OUT">
      <FS type="ne-location">
        <F name="LOCNAME">
          <FS type="&quot;Paris&quot;"/>
        </F>
        <F name="LOCTYPE">
          <FS type="city"/>
        </F>
      </FS>
    </F>
  </FS>
</MATCHINFO>
```

```
-->
```

```
<rmrs cfrom="3" cto="7">
  <label vid="1"/>
  <ep cfrom="3" cto="7">
    <gpred>ne-location</gpred>
    <label vid="2"/>
    <var sort="x" vid="2"/>
  </ep>
  <rarg>
    <label vid="2"/>
    <rargname>CARG</rargname>
    <constant>"Paris"</constant>
  </rarg>
</rmrs>
```

Figure 6: Transforming feature structure XML markup (SProUT) to RMRS (DEEPTHOUGHT).

```
<?xml version="1.0" ?>   <!-- minimal typed feature structure DTD -->
<!ELEMENT FS ( F* ) >
<!ATTLIST FS type  NMTOKEN #IMPLIED
             coref NMTOKEN #IMPLIED >
<!ELEMENT F  ( FS ) >
<!ATTLIST F  name  NMTOKEN #REQUIRED >
```

The FS tag encodes typed feature structure nodes, F encodes features. Atoms are encoded as typed feature structure nodes with empty feature list. An important point is the encoding of coreferences (reentrancies) between feature structure nodes which denote structure sharing.

An application of WHAT access or transformation of deep annotation would be to specifiy a feature path under which a value (type, atom, or complex FS) is to be returned. Because of limited space, we give only a short example here, namely access to output of the shallow SProUT system [Drozdzynski *et al.*, 2004] that produces disjunctions of typed feature structures as output. The `select` expression assigns the value (type) of a feature under the specified attribute path YEAR in the feature structure typed 'point' to an XSLT variable.

```
<xsl:variable name="year" select='FS[@type="point"]/F[@name="YEAR"]/FS/@type'/>
```

The complex stylesheet from which this example is taken, translates SProUT analysis results of named entity recognition encoded as typed feature structures into XML-encoded RMRS structures (Fig. 6) which form the common representation for deep and shallow NLP processing results in the DEEPTHOUGHT architecture [Callmeier *et al.*, 2004; Copestake, 2003]. Deep-shallow integration is then performed on the basis of these RMRS representations. For a list of further applications of XML-based feature structure transformation cf. section 5 in [Lee *et al.*, 2004].

To complete the picture of abstraction through WHAT queries, we can imagine that the same types of query are possible to access, *e.g.* the same morphology information in both shallow and in deep annotation, although their representation within the annotation might be totally different.

5 Related Work

Architectures that combine deep and shallow processing are emerging, but none of the systems described so far cover as much different NLP processing layers as WHITEBOARD. E.g., integration in

[Grover *et al.*, 2002] was limited to tokenization and PoS tagging, in [Daum *et al.*, 2003] to PoS tagging and chunks. DEEPTHOUGHT [Callmeier *et al.*, 2004] is a recent and promising approach that concentrates integration on the level of uniform underspecified semantics representation.

Several XML-based or XML-supporting architectures and tools have been proposed and developed for natural language processing, *e.g.* LT-XML [Brew *et al.*, 2000], RAGS [Cahill *et al.*, 1999] and XCES [Ide and Romary, 2001]. Overviews are also given by [McKelvie *et al.*, 1998; Ide, 2000; Carletta *et al.*, 2002].

As argued in [Thompson and McKelvie, 1997], standoff annotation is a viable solution in order to cope with the combination of multiple overlapping hierarchies and the efficiency problem of XML tree modification for large annotations. We adopt the pragmatic view of [Carletta *et al.*, 2002], who see that computational linguistics greatly benefits from general XMLification, namely by getting for free standards and advanced technologies for storing and manipulating XML annotation, mainly through W3C and various open source projects. The trade-off for this benefit is a representation language somewhat limited with respect to linguistic expressivity.

NiteQL [Evert and Voormann, 2002] can be seen as an extension to XPath within XSLT, it comes with a more concise syntax especially for document structure-related expressions and a focus on timeline support with specialized queries (for speech annotation). The query language in general does not add expressive power to XSLT and the implementation currently only supports Java XSLT engines.

6 Conclusion and Future Work

We have presented WHAT, an open, flexible and powerful infrastructure based on standard XSLT technology for the online and offline combination of natural language processing components, with a focus on, but not limited to, hybrid deep and shallow architectures.

The infrastructure is portable. As the programming language-specific wrapper code is relatively small, the framework can be quickly ported to any programming language that has XSLT support (which holds for most modern programming and scripting languages). XSLT makes the transformation code portable which it could not be when being based on a DOM implementation (which is always programming language-dependent).

The WHAT framework can easily be extended to new NLP components and document DTDs. This has to be done only once for a component or DTD through XSLT query library definitions, and access will be available immediately in all programming languages for which a WHAT implementation exists.

WHAT can be used to perform computations and complex transformations on XML annotation, provide uniform XML annotation access in order to abstract from component-specific namings and DTD structure. WHAT makes it easier to exchange results between components (*e.g.* to give non-XML-aware components access to information encoded in XML annotation), and to define application-specific architectures for online and offline processing of NLP XML annotation.

Due to its flexibility, the infrastructure is well suited for rapid prototyping of hybrid NLP architectures as well as for developing NLP applications, and can be used to both access NLP XML markup from programming languages and to compute or transform it.

Besides the integration within NLP architectures described in this article, the XSLT-based infrastructure (WHAT) could also be used for interfacing applications, *e.g.* to translate to Thistle [Calder, 2000] for visualization of linguistic analyses and back from Thistle in editor mode, *e.g.* for manual, graphical correction of automatically annotated texts for training *etc*.

Because of unstable standardization and implementation status, we did not yet make use of XQuery [Boag *et al.*, 2002]. XQuery is a powerful, SQL-like query language on XML documents with XPath being a subset of XQuery rather than a sublanguage like of XSLT. Because the WHAT framework is open, it is worth considering XQuery as a future extension. Which engine to ask, an XSLT or an XQuery processor, could be coded in each `<query>` element of the template library. Similarly, extension of the current framework to XSLT 2.0 which among other things supports user-definable functions that can be part of XPath expressions, should be straightforward.

It should be possible to combine WHAT with SDL [Krieger, 2003] to declaratively specify XSLT-based NLP architectures (pipelines, loops, parallel transformation) that can be compiled to Java code. Here, WHAT queries can be used within mediators that convert between modules, or as proper modules that perform computations.

The proximity to W3C standards suggests using XSLT also for transformation of NLP results into application-oriented (W3C) markup like VoiceXML or into RDF or OWL for semantic web integration.

7 Acknowledgements

I would like to thank my collegues, especially Anette Frank, Bernd Kiefer, Hans-Ulrich Krieger, Günter Neumann and Melanie Siegel, for cooperation and many discussions. I would also like to thank three anonymous reviewers and the audience at the SEALTS workshop at HLT-NAACL 2003 for helpful comments. This work has been supported by grants from the German Federal Ministry of Education and Research (FKZ 01IW002, 01IWC02). This document was generated partly in the context of the DEEPTHOUGHT project, funded under the Thematic Programme User-friendly Information Society of the 5th Framework Programme of the European Community (Contract No. IST-2001-37836).

References

[Becker and Frank, 2002] Markus Becker and Anette Frank. A Stochastic Topological Parser of German. In *Proceedings of COLING 2002*, pages 71–77, Taipei, Taiwan, 2002.

[Boag *et al.*, 2002] Scott Boag, Don Chamberlin, Mary F. Fernandez, Daniela Florescu, Jonathan Robie, and Jérôme Siméon. *XQuery 1.0: An XML Query Language*. W3C, http://w3c.org/TR/xquery, 2002.

[Brew *et al.*, 2000] Chris Brew, David McKelvie, Richard Tobin, Henry Thompson, and Andrei Mikheev. *The XML Library LT XML. User documentation and reference guide*. LTG, Univ. of Edinburgh, 2000.

[Cahill *et al.*, 1999] Lynne Cahill, Christy Doran, Roger Evans, Chris Mellish, Daniel Paiva, Mike Reape, Donia Scott, and Neil Tipper. Towards a reference architecture for natural language generation systems (the RAGS project). Technical report, HCRC, University of Edinburgh, 1999.

[Calder, 2000] Joe Calder. *Thistle: Diagram Display Engines and Editors*. HCRC, U. of Edinburgh, 2000.

[Callmeier *et al.*, 2004] Ulrich Callmeier, Andreas Eisele, Ulrich Schäfer, and Melanie Siegel. The DeepThought core architecture framework. In *Proceedings of LREC-2004*, pages 1205–1208, Lisbon, 2004.

[Callmeier, 2000] Ulrich Callmeier. PET — A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6 (1):99 – 108, 2000.

[Carletta *et al.*, 2002] Jean Carletta, David McKelvie, Amy Isard, Andreas Mengel, Marion Klein, and Morten Baun Møller. A generic approach to software support for linguistic annotation using XML. In G. Sampson and D. McCarthy, editors, *Readings in Corpus Linguistics*, London and NY, 2002. Continuum International.

[Clark and DeRose, 1999] James Clark and Steve DeRose. *XML Path Language (XPath)*. W3C, http://w3c.org/TR/xpath, 1999.

[Clark, 1999] James Clark. *XSL Transformations (XSLT)*. W3C, http://w3c.org/TR/xslt, 1999.

[Copestake, 2002] Ann Copestake. *Implementing Typed Feature Structure Grammars*. CSLI publications, Stanford, CA, 2002.

[Copestake, 2003] Ann Copestake. Report on the design of RMRS. Technical Report D1.1b, University of Cambridge, Cambridge, UK, 2003.

[Crysmann *et al.*, 2002] Berthold Crysmann, Anette Frank, Bernd Kiefer, Stefan Müller, Jakub Piskorski, Ulrich Schäfer, Melanie Siegel, Hans Uszkoreit, Feiyu Xu, Markus Becker, and Hans-Ulrich Krieger. An Integrated Architecture for Deep and Shallow Processing. In *Proceedings of ACL 2002*, Philadelphia, PA, 2002.

[Daum *et al.*, 2003] M. Daum, K.A. Foth, and W. Menzel. Constraint Based Integration of Deep and Shallow Parsing Techniques. In *Proceedings of EACL 2003*, Budapest, 2003.

[Drozdzynski *et al.*, 2004] Witold Drozdzynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz*, 1:17–23, 2004. http://www.kuenstliche-intelligenz.de /archiv/2004_1/sprout-web.pdf.

[Evert and Voormann, 2002] Stefan Evert and Holger Voormann. *NITE Query Language, NITE Project Document*. University of Stuttgart, Stuttgart, 2002.

[Frank *et al.*, 2003] Anette Frank, Markus Becker, Berthold Crysmann, Bernd Kiefer, and Ulrich Schäfer. Integrated shallow and deep parsing: TopP meets HPSG. In *Proceedings of ACL-2003*, pages 104–111, Sapporo, Japan, 2003.

[Grover *et al.*, 2002] Claire Grover, Ewan Klein, Alex Lascarides, and Maria Lapata. XML-based NLP tools for analysing and annotating medical language. In *Proceedings of the Second International Workshop on NLP and XML (NLPXML-2002)*, Taipei, Taiwan, 2002.

[Ide and Romary, 2001] Nancy Ide and Laurent Romary. A common framework for syntactic annotation. In *Proceedings of ACL-2001*, pages 298–305, Toulouse, 2001.

[Ide, 2000] Nancy Ide. The XML framework and its implications for the development of natural language processing tools. In *Proceedings of the COLING Workshop on Using Toolsets and Architectures to Build NLP Systems*, Luxembourg, 2000.

[Krieger and Schäfer, 1994] Hans-Ulrich Krieger and Ulrich Schäfer. $\mathcal{TDL}$—a type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, pages 893–899, 1994.

[Krieger, 2003] Hans-Ulrich Krieger. SDL—a description language for building nlp systems. In *Proceedings of the HLT-NAACL Workshop on the Software Engineering and Architecture of Language Technology Systems, SEALTS*, pages 84–91, 2003.

[Langendoen and Simons, 1995] D. Terence Langendoen and Gary F. Simons. A rationale for the TEI recommendations for feature structure markup. In Nancy Ide and Jean Veronis, editors, *Computers and the Humanities 29(3)*. Kluwer Acad. Publ., The Text Encoding Initiative: Background and Context, Dordrecht, 1995. Reprint.

[Lee *et al.*, 2004] Kiyong Lee, Lou Burnard, Laurent Romary, Eric de la Clergerie, Ulrich Schaefer, Thierry Declerck, Syd Bauman, Harry Bunt, Lionel Clément, Tomaz Erjavec, Azim Roussanaly, and Claude Roux. Towards an international standard on feature structure representation (2). In *Proceedings of the LREC-2004 workshop on A Registry of Linguistic Data Categories within an Integrated Language Resources Repository Area*, pages 63–70, Lisbon, Portugal, 2004.

[McKelvie *et al.*, 1998] David McKelvie, Chris Brew, and Henry Thompson. Using SGML as a basis for data-intensive natural language processing. *Computers and the Humanities*, 31(5), 1998.

[Piskorski and Neumann, 2000] Jakub Piskorski and Günter Neumann. An intelligent text extraction and navigation system. In *Proceedings of 6th RIAO-2000, Paris*, 2000.

[Schäfer, 2003] Ulrich Schäfer. WHAT: An XSLT-based Infrastructure for the Integration of Natural Language Processing Components. In *Proc. of the Workshop on the Software Engineering and Architecture of LT Systems (SEALTS), HLT-NAACL03*, pages 9–16, Edmonton, Canada, 2003.

[Schmid, 2000] Helmut Schmid. *LoPar: Design and Implementation*. IMS, University of Stuttgart, Stuttgart, 2000. Arbeitspapiere des SFB 340, Nr. 149.

[Skut and Brants, 1998] W. Skut and T. Brants. Chunk tagger: statistical recognition of noun phrases. In *ESSLLI-1998 Workshop on Automated Acquisition of Syntax and Parsing*, Saarbrücken, 1998.

[Thompson and McKelvie, 1997] Henry S. Thompson and David McKelvie. Hyperlink semantics for standoff markup of read-only documents. In *Proceedings of SGML-EU-1997*, 1997.

[Uszkoreit, 2002] Hans Uszkoreit. New Chances for Deep Linguistic Processing. In *Proceedings of COLING 2002*, pages xiv–xxvii, Taipei, Taiwan, 2002.