Shallow, Deep and Hybrid Processing with UIMA and Heart of Gold

Ulrich Schäfer

German Research Center for Artificial Intelligence (DFKI), Language Technology Lab
Campus D 3 1, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
email: ulrich.schaefer@dfki.de

Abstract

The Unstructured Information Management Architecture (UIMA) is a generic platform for processing text and other unstructured, human-generated data. For text, it has been proposed and is being used mainly for shallow natural language processing (NLP) tasks such as part-of-speech tagging, chunking, named entity recognition and shallow parsing. However, it is commonly accepted that getting interesting structure and semantics from documents requires deeper methods. Therefore, one of the future goals for UIMA will be inclusion of openly available, deep linguistic parsing technology for the generation of semantics representations from documents.
Heart of Gold is a lightweight, XML-based middleware architecture that has been developed for this purpose. It supports hybrid, i.e. combined shallow and deep processing workflows of multiple NLP components to increase robustness and exploit synergy, and linguistic resources for multiple languages. The notion of explicit transformation between component input and output enables flexible interaction of existing NLP components. Heart of Gold foresees both tightly (same process) and loosely coupled (via networked services) processing modes. Assuming familarity with UIMA, we introduce Heart of Gold and propose and discuss hybrid integration scenarios in the context of UIMA. Possible applications include precision-oriented question answering, deep information extraction and opinion mining, textual entailment checking and machine translation.

1. Introduction

At last with the incubation of UIMA as an Apache project, language technology and natural language processing tools are becoming standard techniques usable in mainstream application software. More and more pre-existing tools for text processing got news clothes and found their way into the UIMA component repository[1]. So, job done – what's next?
If one looks closer at the different types of integrated tools, then only the same few types of components appear – at least those openly available: shallow tools such as part-of-speech taggers, chunkers, named entity recognizers and entity detectors, the latter ones for specific tasks or domains. But this is only half the range of natural language processing (besides the language dimension that is currently mostly English).
To get structure and semantics from unstructured text, much more is needed than identifying types of named entities or part-of-speech tags. Ultimately, one needs text understanding, getting the relations between the various entities mentioned in the text, or at least a predicate-argument structure per sentence. This cannot be provided only by shallow tools, but requires deep parsing.
Moreover, even rather shallow tasks such as template-based information extraction work better in rather fixed word-order languages such as English, but perform worse on free word-order languages. Again, deep syntactic parsing could help to improve results. While efficiency is no longer a problem for deep parsing, robustness can be overcome using a hybrid approach we will discuss below.
The distinction between shallow and deep processing is a continuum rather than a strict dichotomy. Deep means knowledge-intensive, comprehensive, generic. By shallow, we mean partial, less informed analysis, often domain-dependent. It has to be pointed out that the distinction between statistical and rule-based NLP is orthogonal to that, as deep and shallow analyses may involve both. For more in-depth discussions, cf. (Uszkoreit, 2002; Schäfer, 2007).
There is one further distinction that plays a role when characterizing the kind of analysis results and its relation to NLP software architecture. (Cunningham et al., 1997) present a classification of software infrastructures for NLP by distinguishing three models they call

- *referential* (analyses are stored as separate representations with pointer references into the original text),
- *additive* (e.g. cumulative SGML/XML annotation markup), and
- *abstraction-based* (as in typed feature structures of deep analysis where the analysis result consists of a closed, integrated information structure for larger text entities, typically a whole sentence).

Thus, architectures for shallow *and* deep components should support at least referential and abstraction-based representations. The latter is not supported by architectures such as GATE (Bontcheva et al., 2004).
Although the designers of UIMA had deep processing in mind already when they started developing their framework (Ferrucci and Lally, 2004; Götz and Suhre, 2004), at least openly available deep processing is currently less developed in UIMA than in other approaches, and so is the novel hybrid (combined deep and shallow) integration paradigm.
In this paper, we will present another framework, Heart of Gold, and discuss its relation to UIMA. This framework has been developed independently of and in parallel to UIMA. It integrates mainly openly available shallow and deep processing components and linguistic resources for many languages.
Heart of Gold (Callmeier et al., 2004; Schäfer, 2007)[2] is a lightweight, XML-based middleware architecture that has

[1] http://uima.lti.cs.cmu.edu

[2] Download, documentation: http://heartofgold.dfki.de

been developed in the context of DELPH-IN[3], a collaboration of various research groups developing and sharing open source tools and linguistic resources for the Head-driven Phrase Structure Grammar (Pollard and Sag, 1994). Being open source, Heart of Gold is also contained in the OpenNLP collection[4].

The main motivation why Heart of Gold has been devised is flexible support for the combination of multiple shallow NLP analysers with a deep HPSG parser, and for generating robust deep semantic representations of the meaning of natural language sentences. It could be shown that through integration with PoS tagging and named entity recognition, deep parsing coverage on newspaper text can be doubled, even on broad-coverage grammars with relatively large lexica (Crysmann et al., 2002; Schäfer, 2007).

We will in the following discuss the Heart of Gold approach, how it differs from and can be brought together with UIMA. The idea is that if Heart of Gold would be migrated to UIMA (hypothetically), not only single components should be migrated, but also the efforts invested in elaborated hybrid integration workflows should be preserved, e.g. for English, German and Japanese.

2. Heart of Gold

2.1. Design principles

One of the design decisions that have been made in Heart of Gold is the choice of open XML standoff markup as the only representation format for input and output of the components. It contains aspects of both *referential* (through character offset positions encoded in attributes) and *additive* representation architectures mentioned in the introduction.

Standoff markup is easy to exchange, transformable using standard XML transformation languages such as XSLT (Clark, 1999), and interoperability benefits from Unicode being part of the XML standard. The XML approach is in principle compatible with UIMA which in addition supports isomorphic object structure in the supported programming languages. The elegance of the XML approach lies in the closeness to XML corpus annotation, i.e. persistently 'multidimensionally' stored analysis results form an automatically annotated corpus.

Fig. 1 gives a schematic overview of the Heart of Gold middleware architecture in between applications (top) and external NLP components (bottom). Communication with the middleware is supported via XML-RPC web service or programmatically via a Java API. When a new application session is started, it takes a configuration specifying the wrapped NLP components to start for this session. Each component is started according to its own configuration.

An application client can send texts to the middleware and the NLP components are then queried in a numerically defined processing order ('depth'). The shallowest components (e.g. tokenizer) are assigned a low number and are started first etc. The output of each component must be

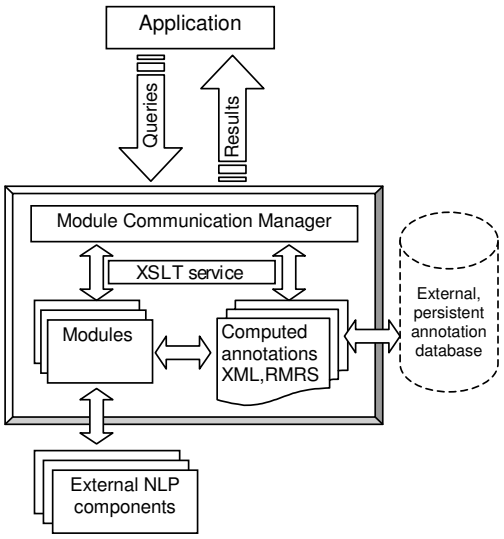

Figure 1: Middleware architecture

XML markup. Each component gets the output of the previous component as input by default, but can also request (via configuration) other annotations as input.

As there is no commonly accepted XML standard for linguistic annotation, the architecture itself makes no assumption about the XML format as long as it is well-formed XML. XML transformation is used to mediate between different I/O formats.

Components may produce multiple output annotations (e.g. in different formats). Thus, the component dependency structure in general forms a graph. In Section 2.8., we describe a further generalization of the default pipeline.

2.2. Session and annotation management

The resulting NLP annotations are stored in a per-session markup storage (Fig. 2) that groups all annotations for an input query (a sentence or text) in *annotation collections*. The markup can also be made persistent by saving it to XML files or storing it in an XML database. Annotations can be accessed uniquely via a URI of the form

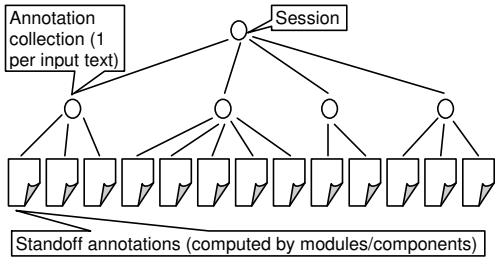

Figure 2: Session and multi-dimensional markup storage

`hog://sid/acid/aid` in XPath expressions where `sid` is a session ID, `acid` is an annotation collection ID and `aid` is an annotation identifier typically signifying the name of the producing component. Structured metadata like configuration and processing parameters (e.g. processing time and date, language ID etc.) are automatically stored within the annotation markup as first root daughter element.

[3]DEep Linguistic Processing with HPSG Initiative; http://www.delph-in.net

[4]http://opennlp.sf.net

Component	NLP Type	Languages	Implemented in
JTok	tokenizer	de, en, it,...	Java
ChaSen	Japanese segm./tagger.	ja	C
TnT	HMM tagger	de, en,...	C
Treetagger	statistical tagger	en, de, es, it,...	C
Chunkie	HMM chunker	de, en,...	C
ChunkieRMRS	chunk RMRSes	de, en	XSLT, SDL/Java
LingPipe	statistical NER	en, es,...	Java
FreeLing	morph./tagger/NER	ca, en, es, gc, it	C++
Sleepy	shallow parser	de	OCaml
SProUT	morph., shallow NLP/NER	de, el, en, ja,...	XTDL, Java
LoPar/wbtopo	PCFG parser	de	C, XSLT
Corcy	coref resolver	en	Python
RASP	shallow NLP	en	C, Lisp
PET	HPSG parser	de, el, en, ja,...	C, C++, Lisp
RMRSmerge	RMRS merger	de, en,...	XSLT, SDL/Java
SDL	generic sub-architectures		SDL/Java

Figure 3: Integrated components from shallow (top) to deep (bottom). Details and references on http://heartofgold.dfki.de.

2.3. Wrapped NLP components

NLP components are integrated through adapters called modules (either Java-based, subprocesses or via XML-RPC) that are also responsible for generating XML standoff output in case this is not supported natively by the underlying, pre-existing component. Various shallow and deep NLP components have already been integrated, cf. Fig. 3.

2.4. Integration through transformation

Heart of Gold heavily relies on the use of XSLT for combining and integrating XML markup produced by the NLP components. The general idea is to use XSLT to transform XML to other XML formats, or to combine and query annotations. In particular, XSLT stylesheets may resolve conflicts resulting from multi-dimensional markup, choose among alternative readings, follow standoff links, or decide which markup source to give higher preference.

(Carletta et al., 2003), e.g. propose the NXT Search query language (for corpus access) that extends XPath by adding query variables, regular expressions, quantification and special support for querying temporal and structural relations. Their main argument against standard XPath is that it is impossible to constrain both structural and temporal relations within a single XPath query. Our argument is that XSLT can complement XPath where XPath alone is not powerful enough, yet providing a standardized language.

Further advantages we see in the XSLT approach are portability and efficiency (in contrast to 'proprietary' and slow XPath extensions like NXT), while it has a quite simple syntax in its (currently employed) 1.0 version. XSLT can be conceived as a declarative specification language as long as an XML tree structure is preserved (not necessarily fully isomorphic to the input structure). However, XSLT is Turing-capable and therefore suited to solve in principle any markup integration or query problem.

Finally, extensions like the upcoming XSLT/XPath 2.0 version or efficiency gains through XSLTC (translet compilation) can be taken on-the-fly and for free without giving up compatibility. Technically, the built-in Heart of Gold XSLT processor could easily replaced or complemented by an XQuery processor. However, for the combination and transformation of NLP markup, we see no advantage of XQuery over XSLT.

Heart of Gold comes with a built-in XSL transformation service, and module adapters can easily implement transformation support by including a few lines of code. Stylesheets can also be generated automatically in Heart of Gold, provided a formal description of the transformation input format is available. An example is the mapping from named entity grammar output type definitions in the deep-shallow integration scenario we will describe briefly by example below.

2.5. Performance

There is a slight performance drawback Heart of Gold shares with other service-oriented architectures. It is imposed by the XML framework, yet partly countervailed by fast XSL transformation. While deep parsing alone is in the range of milliseconds per sentence thanks to the very efficient PET system, a hybrid parse may take up to 1-2 seconds including PoS tagging, named entity recognition, and some more seconds for very long sentences.

The majority of the time goes into Java-based XML processing, and there is room for optimization. However, we think this is an acceptable tradeoff for very flexible and quick experimental integration of (new) NLP components in exciting new, rapidly prototyped applications, including the benefits of Unicode given for free in multilingual integration scenarios.

2.6. Integrating shallow and deep processing

The main motivation for integrating deep and shallow processing is that deep parsing alone is not robust enough. Open class words such as names, locations, time expressions not in the deep lexicon prevent construction of full parse trees. A simple, yet very efficient way of making parsing more robust to gaps in the lexicon is using PoS tagging as pre-processing. From the PoS information for a word unknown to the deep lexicon, one or more generic

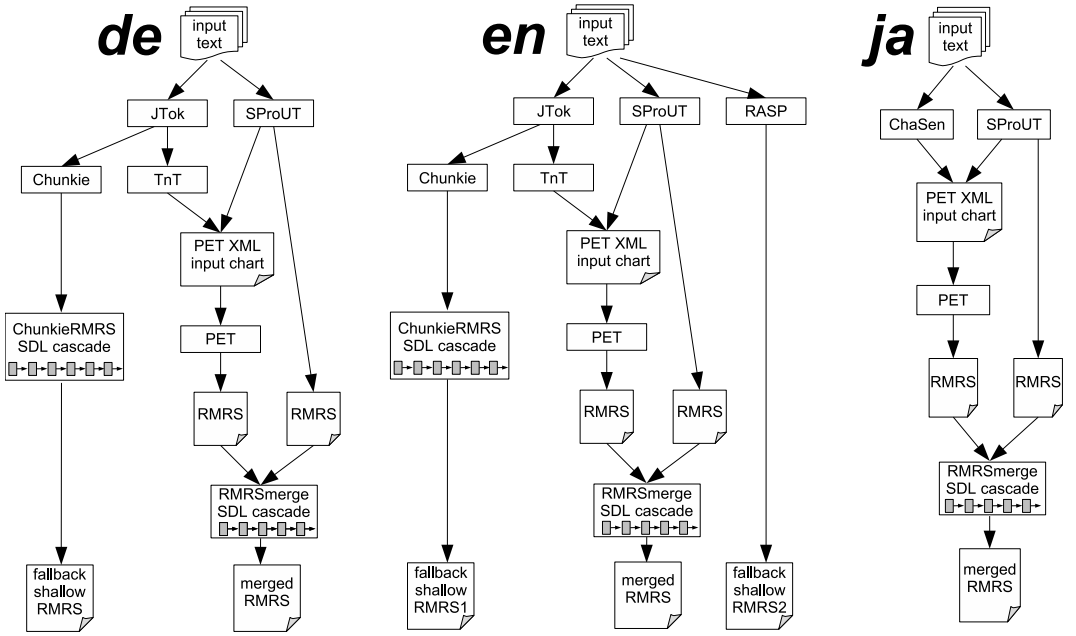

Figure 4: Hybrid workflows for German, English, Japanese

lexicon entry is put on the deep parser's chart containing at least the information about the word class and maybe other information such as morphological or basic semantics features.

In the same way, named entity recognizers and gazetteers may contribute e.g. domain-specific information missing in the deep grammars. This forms a division of labor: the (expensive) deep grammar is responsible for modelling correct general language use, syntax and generating a sentence-semantic representation, while the shallow components add domain-specific information that does not need to be maintained in the deep lexicon and can be easily changed for a different application domain.

We now give an example for such a hybrid workflow, depicted for English in the middle of Figure 4. The configuration for German is analogous except that there is no secondary shallow fallback component.

The raw input sentence text is sent to the JTok tokenizer and the named entity recognizer SProUT (Drożdżyński et al., 2004), because SProUT comes with its own tokenizer with a finer-grained token classification. Chunkie (HMM chunker) and TnT (HMM tagger) use the tokenized output from JTok as input, Chunkie output is used as secondary input for the ChunkieRMRS cascade (left branch in Figure 4 for German and English) we will be explain Section 2.8.

The output of this cascade (shallow RMRS) can be used as shallow fallback result in case the deep parser fails to parse the input sentence. Similarly, RASP (English only) produces another shallow RMRS as fallback annotation.

Back to the middle pipeline, the tagger output for the sentence 'George Washington was born in Virginia'

```
<w id="TNT0" cstart="0" cend="5">
  <surface>George</surface>
  <pos tag="NNP" prio="1.000000e+00"/>
</w>
<w id="TNT1" cstart="7" cend="16">
  <surface>Washington</surface>
  <pos tag="NNP" prio="1.000000e+00"/>
</w>
<w id="TNT2" cstart="18" cend="20">
  <surface>was</surface>
  <pos tag="VBD" prio="1.000000e+00"/>
</w>
<w id="TNT3" cstart="22" cend="25">
  <surface>born</surface>
  <pos tag="VBN" prio="1.000000e+00"/>
</w>
<w id="TNT4" cstart="27" cend="28">
  <surface>in</surface>
  <pos tag="IN" prio="1.000000e+00"/>
</w>
<w id="TNT5" cstart="30" cend="37">
  <surface>Virginia</surface>
  <pos tag="NNP" prio="1.000000e+00"/>
</w>
```

as well as the recognized named entities from SProUT

```
<w id="SPR1" cstart="0" cend="16" prio="0.5"
   constant="yes">
  <surface>George Washington</surface>
  <typeinfo id="TIN1" baseform="no">
    <stem>$genericname</stem>
  </typeinfo>
</w>
<w id="SPR2" cstart="30" cend="37" prio="0.5"
   constant="yes">
  <surface>Virginia</surface>
  <typeinfo id="TIN2" baseform="no">
    <stem>$genericname</stem>
  </typeinfo>
</w>
```

are transformed into the deep parser's (PET; (Callmeier, 2000)) input chart format using XSLT (shown above is already the transformed version). Another XSLT stylesheet

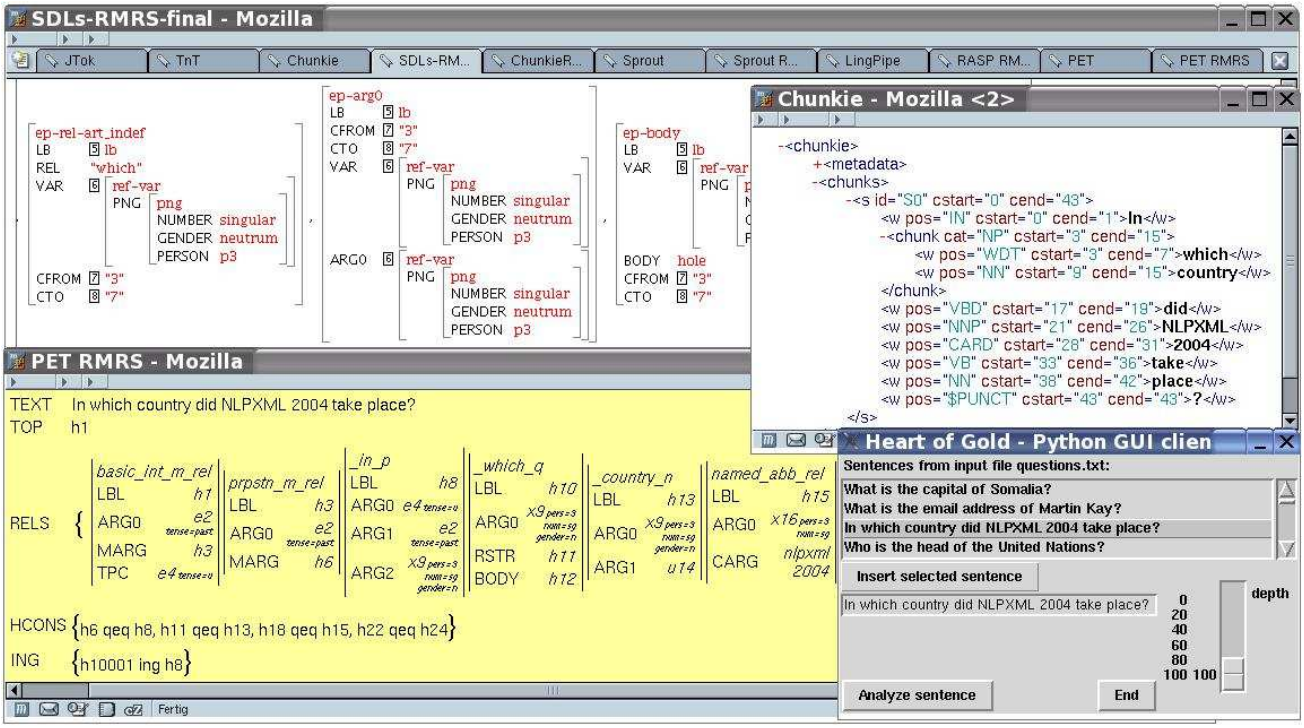

Figure 5: Heart of Gold analysis results in GUI with specialized XML visualizations

is used to combine these and possibly other annotations in a single PET input chart document [5].
From this XML input chart, the deep parser generates or looks up deep lexicon entries, then starts HPSG parsing.

2.7. Output: semantics representation

Instead of huge typed feature structures containing the monotonically assembled unification result of the HPSG parse tree per sentence, applications are rather interested in a distilled sentence semantics representation. This distillate largely omits linguistics details from morphology and syntax, but provides a graph structure of connected semantic entities for the whole sentence, including its predicate-argument structure.
One such representation generated by many modern HPSG grammars is MRS - minimal recursion semantics (Copestake et al., 2005) or its robust XML variant RMRS (Copestake, 2003). RMRS turns the semantics representation of a sentence into an XML standoff format as well (including references back into character positions of the input sentence) and thus is appropriate for being processed by the middleware and forwarded to applications.
An RMRS contains EPs (elementary predications) with argument connected via handle and individual variables. The idea is that shallow NLP components may deliver equivalent where possible, but maybe underspecified representations, e.g. the argument positions of a transitive verb may be empty when a shallow parser cannot find the appropriate object. The HCONS (handle constraints) attribute allows to concisely express scopus ambiguities via handles. The ING (in-group) attribute explicitly indicates conjunction of its contained pairs.
A sample RMRS as produced by the deep parser PET running the HPSG grammar ERG[6] in Heart of Gold is shown in Figure 6, depicted in the MRS matrix format instead of raw XML for better readability.
Figure 7 shows a structured result from the named entity recognizer SProUT transformed to the RMRS format. It contains information such as name variants or the indication that Virginia is of type province. This information was not passed to the deep grammar as it is irrelevant for parsing in this case, but it might be interesting for consuming applications.
Thus, RMRS is used as a uniform, though not mandatory output format of both deep and shallow components. The RMRSmerge module at the end of the shallow-deep pipeline can be used to merge RMRSes produced by multiple components into a single representation ('merged RMRS' in Figure 4).

2.8. Sub-architectures

Heart of Gold modules roughly correspond to TAEs (Text Analysis Engines) in UIMA. The equivalent to UIMAs composed TAEs are sub-architectures in Heart of Gold.
The SDL module enhances Heart of Gold with a compilable NLP module control flow for sub-architectures, i.e., enabling declarative specification of modules that are composed of other modules. SDL (System Description Language) has been developed independently of Heart of Gold by (Krieger, 2003).
SDL generates Java code for declaratively defined architectures of NLP systems obeying a class interface imposed by

[5] Stylesheets are also employed to visualize the linguistic markup, e.g. by transforming analysis results to HTML (Fig. 5) or LaTeX.

[6] English Resource Grammar; http://www.delph-in.net/erg/

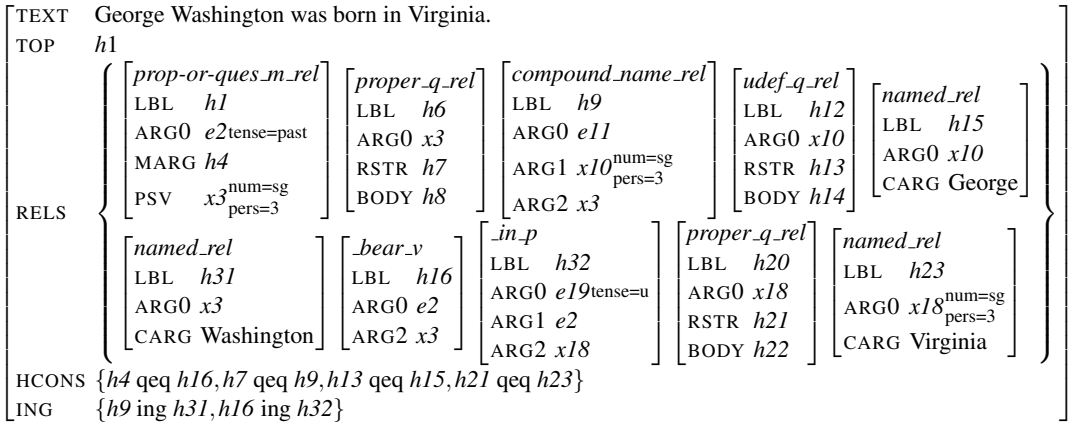

Figure 6: Deep semantics representation (RMRS) by ERG and PET for "George Washington was born in Virginia".

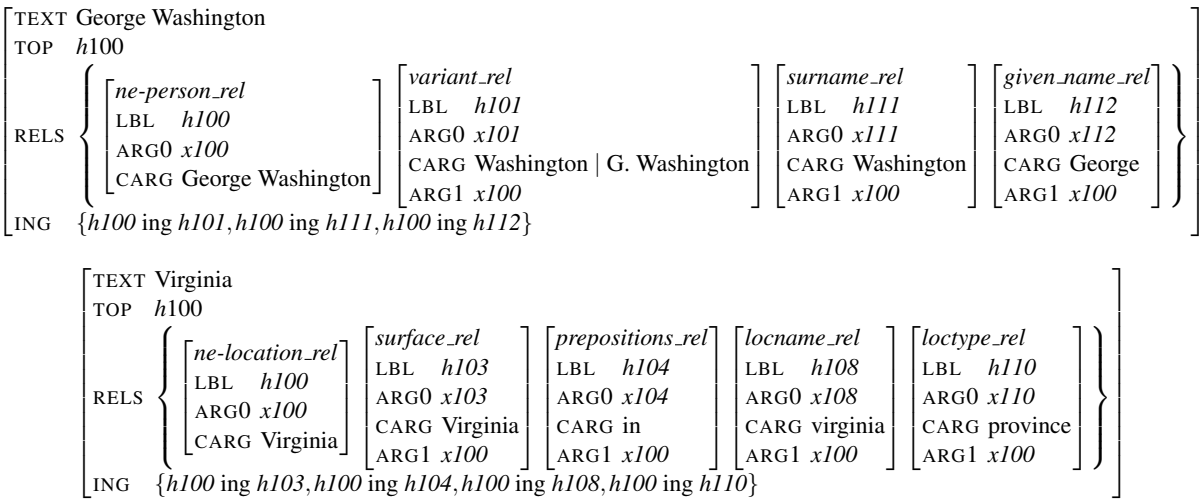

Figure 7: Shallow RMRS by SProUT for the named entities "George Washington" and "in Virginia".

the SDL framework. The initial intention was to be able to declaratively define cascaded SProUT instances, e.g. for shallow chunk parsing. An application are e.g. cascades of (shallow) NLP modules and XSL transformations.

Although the described mainly sequential control flow approach in Heart of Gold for NLP modules by defining a depth and canonical processing order based upon, augmented with potentially multiple input and multiple output annotations in each processing step, was flexible enough for deep-shallow integrations for many languages, it turned out that some envisaged, RMRS-related shallow processing applications required additional features such as loops and parallelism – which SDL supports.

The declarative specification of the architecture is a single expression consisting of symbolic module names connected via operators, plus assignment of these symbolic module names to Java class names, constructor arguments, and some processing options.

The SdlModule is a generic wrapper plugging SDL subarchitectures into the Heart of Gold. SdlModule acts like any other Heart of Gold module in that it takes a (configurable) XML annotation as input, and returns an output annotation.

The name of the embedded SDL Java class containing the compiled architecture description (previous section) is part of the SdlModule configuration. The generated Java code of the SDL description is compiled and executed at runtime in the SdlModule code using Java reflection.

ChunkieRMRS (Frank et al., 2004), left branch of the German and English workflows in Figure 4, shall now serve as an example of such a compound, SDL-based component. Externally, it acts like a single component, but consists of eight sub-modules in this case (Fig. 8).

A robust, partial semantics representation is generated from a shallow chunker's output and morphological analysis by means of a processing cascade consisting of four SProUT grammar instances with four interleaved XSLT transformations. SProUT is used here for intermediate, rule-based transformation of complex typed feature structures.

The scenario is equally a good example for XSLT-based annotation integration. Chunker analysis results are included in the RMRS to be built through an XSLT stylesheet using the XPath expression

```
document($uri)/chunkie/chunks/chunk[
   @cstart=$beginspan and @cend=$endspan]
```

where `$uri` is a variable containing an annotation identifier of the form `hog://sid/acid/aid` as explained in Section 2.2.

2.9. Applications

A recent application of the middleware for English is hybrid processing of scientific papers in the field of language

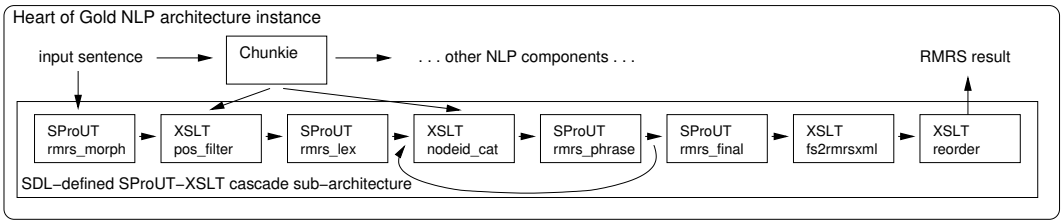

Figure 8: SDL sub-architecture for constructing RMRSes from chunks in Heart of Gold

technology (Schäfer et al., 2008). Currently abstracts, later full papers from the ACL Anthology (Bird et al., 2008) are extracted from PDF, parsed with Heart of Gold, and so-called quriples are extracted from the RMRS. Quriples are query-oriented subject-verb-object-rest tuples that are indexed and made searchable from a GUI application called the Scientist's Workbench. 62.5% full parse coverage could be reached with out-of-the-box components and lingware resource in a pipeline as described in Section 2.6.

Another application is question answering from structured knowledge sources such as ontologies or databases. In the QUETAL system (Frank et al., 2006), the hybridly computed RMRSes of natural language questions, both German and English, are directly translated to SPARQL ontology queries of which the results are returned as answers formulated by a template-based generator.

There are various further applications of purely shallow configuration instances of Heart of Gold, e.g. for information extraction on soccer game descriptions (Buitelaar et al., 2006) and opinion mining.

2.10. Related Work

There is few related work on hybrid NLP architecture. Most others such as (Grover and Lascarides, 2001) are systems that integrate specific instances of shallow and deep tools without having the right or claiming themselves to form generic architectures. GATE is shallow by design and (without modification) not suited for abstraction-based components such as deep parsers.

An interesting approach from a research area unrelated to language technology by (Löwe and Noga, 2002) bears some similarity with Heart of Gold. They describe a generic XML-based, network-enabled middleware architecture for re-usable components that explicitly makes use of XSLT as adapter language between components. It has been proposed as a generic middleware in the spirit of CORBA, DCOM or EJB. However, it can well be conceived as a supporting, independent argument that the XML and XSLT-based middleware approach is a useful design pattern for software architecture.

3. UIMA Integration Scenarios

In this section, we discuss a hypothetical migration of hybrid processing in Heart of Gold to UIMA. The cheap way of migrating to UIMA would be to wrap Heart of Gold configuration instances as a whole in a UIMA TAE (text analysis engine). But this would probably not add any value. There is no doubt that components currently integrated in Heart of Gold could be migrated to UIMA, each in a separate TAE, as well as the simple, 'direct' pipelines for hybrid processing, as composed TAEs.

Going this way would require more implementation work, but the result would be (hopefully) analogous configurability, then UIMA-enabled. To keep the same flexibility as in Heart of Gold, the configurable stylesheets for transformation between components could be put in separate TAEs or as adapters. At the end, UIMA would benefit from new (mostly open source) TAEs, and the new paradigm of hybrid analysis.

An interesting, but even more implementation-intensive approach would be separating linguistic resources such as grammars or lexica specific to components by putting them behind KSAs (knowledge source adapters). Currently, each component comes with its own resources and resource format. There is some synergetic gain forseeable through KSAs, but there is doubt that this will be worth the effort for every component.

Another interesting approach would be sharing the type hierarchy among deep and shallow components. Currently, this is possible for the deep parser PET and the generic NLP engine SProUT. Both use the same very efficient bit-vector encoding technique for their type system (Callmeier, 2000). As it is for HPSG, it necessarily supports multiple inheritance, while in the UIMA, only single-inheritance type systems seem to be supported which would cause a problem e.g. for the feature structure structure representation of parse results.

The biggest effort will probably have to be invested in the CAS (Common Analysis Structure). The lightweight Heart of Gold proposes and supports RMRS as optional common format, but is also open to any other standoff format. Agreements on the formats are only necessary between connected components.

In UIMA, the I/O of TAEs has to be specified more rigidly as part of the CAS. In the ideal case, this could result in systems where the workflow can be computed automatically (in the ideal case) from a global I/O specification, e.g. by an application. Currently, this is a manual task in Heart of Gold.

4. Summary and Outlook

We have presented Heart of Gold and discussed its relation to and possible connection with UIMA. UIMA is an emerging, industrial-strength platform for application-oriented processing of unstructured data such as natural language text. It has been designed very thoroughly and now constitutes a rather complex framework. Therefore, mainly shallow NLP tools have been migrated to UIMA so far.

Heart of Gold is meant mainly as a lightweight research instrument for flexible experimentation with hybrid, XML-

based NLP component integration and for rapid prototyping of applications using semantic analyses of text. Research on deep processing and improving it with respect to robustness through various approaches, also other than integrating it with shallow tools, e.g. through additional statistical models and extensions, is a hot research topic.

Now that hybrid processing has turned out promising and proven successful for a range of applications, UIMA may help to bring deep and hybrid processing faster to a broader community and market. And vice versa: UIMA and UIMA-based applications will benefit from increased analysis depth gained through hybrid processing.

5. Acknowledgments

This work has been supported by a grant from the German Federal Ministry of Education and Research (FKZ 01 IW F02). Thanks to the anonymous reviewers for their valuable, concise and encouraging comments.

6. References

Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. 2008. The ACL anthology reference corpus: a reference dataset for bibliographic research. In *Proceedings of LREC-2008*, Marrakech, Morocco.

Kalina Bontcheva, Valentin Tablan, Diana Maynard, and Hamish Cunningham. 2004. Evolving GATE to meet new challenges in language engineering. *Natural Language Engineering*, 10(3-4).

Paul Buitelaar, Thomas Eigner, Greg Gulrajani, Alexander Schutz, Melanie Siegel, Nicolas Weber, Philipp Cimiano, Günter Ladwig, Matthias Mantel, and Honggang Zhu. 2006. Generating and visualizing a soccer knowledge base. In Frank Keller and Gabor Proszeky, editors, *Proceedings of the EACL06 Demo Session*, Trento, Italy.

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

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

Jean Carletta, Stefan Evert, Ulrich Heid, Jonathan Kilgour, Judy Robertson, and Holger Voormann. 2003. The NITE XML toolkit: flexible annotation for multimodal language data. *Behavior Research Methods, Instruments, and Computers, special issue on Measuring Behavior*, pages 353–363.

James Clark, 1999. *XSL Transformations (XSLT)*. World Wide Web Consortium, http://w3c.org/TR/xslt.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: an introduction. *Journal of Research on Language and Computation*, 3(2–3):281–332.

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

Berthold Crysmann, Anette Frank, Bernd Kiefer, Stefan Müller, Jakub Piskorski, Ulrich Schäfer, Melanie Siegel, Hans Uszkoreit, Feiyu Xu, Markus Becker, and Hans-Ulrich Krieger. 2002. An Integrated Architecture for Deep and Shallow Processing. In *Proceedings of ACL 2002*, pages 441–448, Philadelphia, PA.

Hamish Cunningham, Kevin Humphreys, Robert Gaizauskas, and Yorick Wilks. 1997. Software infrastructure for natural language processing. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 237–244, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Witold Drożdżyński, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. 2004. Shallow processing with unification and typed feature structures – foundations and applications. *Künstliche Intelligenz*, 2004(1):17–23.

David Ferrucci and Adam Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348.

Anette Frank, Kathrin Spreyer, Witold Drożdżyński, Hans-Ulrich Krieger, and Ulrich Schäfer. 2004. Constraint-based RMRS construction from shallow grammars. In *Proceedings of the HPSG-2004 Conference, Center for Computational Linguistics, Katholieke Universiteit Leuven*, pages 393–413. CSLI Publications, Stanford, CA.

Anette Frank, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg, and Ulrich Schäfer. 2006. Question answering from structured knowledge sources. *Journal of Applied Logic*, pages 20–48. DOI: 10.1016/j.jal.2005.12.006.

Thilo Götz and Oliver Suhre. 2004. Design and implementation of the UIMA common analysis system. *IBM Systems Journal*, 43(3). DOI: 10.1147/sj.433.0476.

Claire Grover and Alexis Lascarides. 2001. XML-based data preparation for robust deep parsing. In *Proceedings of ACL/EACL 2001*, pages 252–259, Toulouse, France.

Hans-Ulrich Krieger. 2003. SDL – A description language for building NLP systems. In *Proc. of the HLT-NAACL Workshop on the Software Engineering and Architecture of Language Technology Systems*, pages 84–91.

Welf Löwe and Markus L. Noga. 2002. A lightweight XML-based middleware architecture. In *Proceedings of IASTED AI 2002*, Innsbruck. ACTA Press.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. University of Chicago Press, Chicago.

Ulrich Schäfer, Hans Uszkoreit, Christian Federmann, Torsten Marek, and Yajing Zhang. 2008. Extracting and querying relations in scientific papers on language technology. In *Proc. of LREC-2008*, Marrakesh, Morocco.

Ulrich Schäfer. 2007. *Integrating Deep and Shallow Natural Language Processing Components – Representations and Hybrid Architectures*. Ph.D. thesis, Faculty of Mathematics and Computer Science, Saarland University, Saarbrücken, Germany.

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