Towards an Ontology-based Language Infrastructure

Major results from 2006 joint research activities under NICT international research grant

Yoshihiko Hayashi (Osaka University)
Paul Buitelaar (DFKI, Language Technology)
Thierry Declerck (DFKI, Language Technology)

June, 2007

This report describes major results obtained from the joint research framework supported by the R&D promotion fund from NICT. The ultimate goal of this research is to extract technical issues to be addressed for realizing ontology-based language infrastructures which should be truly required in broad range of intercultural collaboration activities. The participated researchers in the joint framework in 2006 Japanese fiscal year are Yoshihiko Hayashi from Osaka University, and Paul Buitelaar and Thierry Declerck from the Language Technology group of DFKI. This draft not only summarizes the major results from this year's activities, but also examines the future research agenda that should be further investigated by an international collaborative research team.

Contents

1 Architectures for Language Infrastructure

1.1 The Language Grid

Murakami et al. (Murakami, 2006) classifies users of the language grid into *language service providers,* who publish their own atomic language services, and l*anguage service users,* who construct and deploy composite language services by combining existing atomic language services. This classification can be somewhat confusing, because the term *user* usually implies 'end user.' Therefore, in this paper, we classify users of the language grid into *linguistic service providers* and *linguistic service end users*. A linguistic service provider disseminates a linguistic service on the language grid, regardless whether the service is atomic or composite. On the other hand, a linguistic service end user consumes the linguistic services deployed on the language grid, presumably by employing computational tools designed and implemented for assisting intercultural communications.

Suppose you are a potential linguistic service provider willing to provide some useful linguistic function on the language infrastructure by operating a linguistic service on your site. The two possible scenarios are summarized below.

- Utilizing an external web-based linguistic service: For now, a number of web-based linguistic services, such as text translation and dictionary access, are available. The average web user can easily access these services with his/her web browsers. You, as a linguistic service provider, can provide a web service version of such an externally existing linguistic service. To do this, you have to implement a delegation program that formulates a query in the syntax designated by the external service and extracts necessary information from the returned HTML code. Given a situation where a variety of linguistic services are available, this could be a way to provide a linguistic service to the language infrastructure. In this case, the program on your site functions as an adaptor that adjusts detailed differences between the external web-based services and the standard interface defined in the infrastructure. We call this type of program a *wrapper*.
- Running an NLP function at your site: In the second scenario you run an NLP function, such as text translation (NLP system) or morphological analysis (NLP tool), at your site. You can develop such an NLP function on your own. In this case, your program has to be compliant with the standard interface in the infrastructure; the core NLP function of the program has to be wrapped so that it functions as a standard web service. Alternatively, you can install a third-party NLP tool or NLP system at your site, and wrap it as a web service. You would be able to install a NLP tool that is available on an R&D institute's web site, and an NLP system if you acquired the license from the vendor. This scenario's significant difference from the former one is that you have to provide both a wrapper program, and a computational resource for running the NLP tool/system,

regardless of whether it is in-house or developed by a third party. In addition to these NLP functions, we also consider access functions for language resources as a type of linguistic services. Language resources here mean data-only static resources such as lexicons, or corpora. A prominent example of static language resources is a dictionary source data. For example, we can often acquire dictionary source data by downloading or purchasing it. WordNet and EDR Electronic Dictionary are examples of this. A static language resource always has to be equipped with an access function to be utilized in the language infrastructure. The language resource access function also has to be wrapped to be used as a standard web service.

Based on the discussion above, the fundamental technical components in a language infrastructure, particularly in the language grid, are: (a) external web-based services, (b) on-site NLP core functions, (c) static language resources, and (d) wrapper programs.

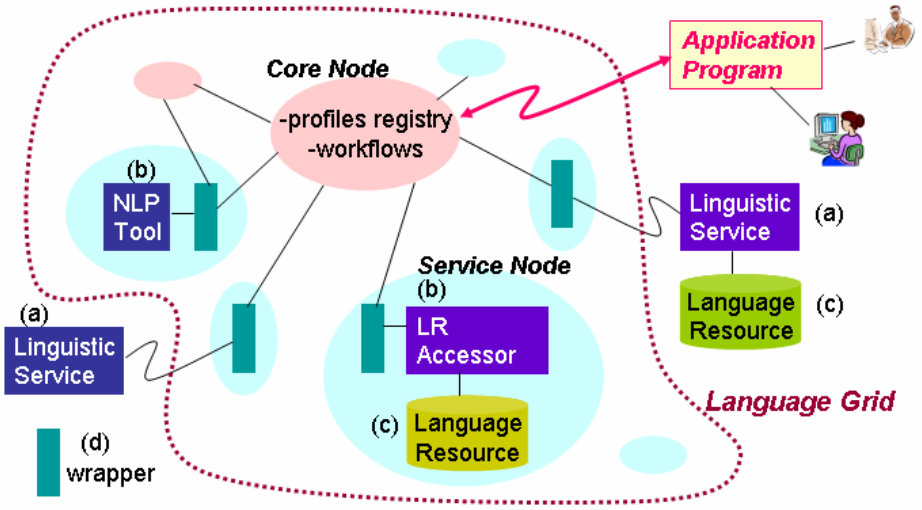

Fig. 1. General architecture of the language grid.

Figure 1 is a schematic diagram of the general architecture of the language grid. The technical components listed above are deployed as shown in the figure. Computational nodes in the language grid are classified into the following two types.

- Service nodes: These accommodate atomic linguistic services based on the configuration discussed in the previous subsection. That is, they can provide functionalities of the NLP tool/system running on a node, or they can simply have a wrapper program that consults an external web-based linguistic service.
- Core nodes: These maintain a repository of the known atomic linguistic services and provide service discovery functionality to the possible users/applications.

They also maintain a workflow repository for composite linguistic services, and are equipped with a workflow execution engine.

1.2 HoG Language Service Platform

Heart of Gold (HoG) is middleware architecture for creating and combining markup produced by multiple natural language processing components in multilingual environments. It provides a uniform and flexible infrastructure for building applications that use natural language processing components based on XML and RMRS (Robust Minimal Recursion Semantics). The main design goals of HoG are:

- flexible integration of NLP components
- simple application interface
- RMRS as uniform representation language (XML-encoded)
- open to other XML standoff annotation formats
- integration of non-RMRS-aware NLP components through annotation transformation
- external annotation database for storage and retrieval of computed linguistic analyses (optional)
- network-enabled architecture with distributed components (optional)
- lightweight, platform- and programming language-independent communication through XML-RPC
- based on current technology like XML, XML-RPC, XSLT, XML:DB, XPath

HoG acts as mediator between applications and NLP components, abstracting from component-specific interfaces and representations. Applications send queries on text documents to the middleware, which in turn passes the queries to one or more components according to an application-specific configuration. The resulting annotation, which can also be taken from the annotation database, is then returned to the application. Multi-dimensional annotations are stored in a per-session markup storage that groups all annotations for an input query (a sentence or text) in *annotation collections*. The markup storage can also be made persistent by saving it to XML files or to an XML database.

HoG treats XML standoff annotations as first class citizens and natively supports XML (and only XML) markup of any kind – this is different from other NLP architectures such as GATE. Moreover, HoG does not prescribe specific DTDs or Schemata for annotations, provided that the markup is well formed. In this sense, it is a completely open framework that allows for easy integration of new components, although this may be constrained by requirements of the actually configured components. More information on HoG can be obtained from (Schäfer,

2006; Schäfer, 2005).

In the context of the SmartWeb project at DFKI, the HoG architecture has been extended towards a web service platform in which NLP components are wrapped as web services that can be accessed over SOAP, XML-RPC and REST protocols. In the context of Language Grid this web service architecture can be further extended towards a truly distributed, Semantic Web based architecture for autonomous, on-demand configuration of NLP-based applications (multilingual information extraction systems, machine translation systems, etc.). For this purpose, the language service ontology developed in Language Grid will be integrated with HoG for semantically annotating NLP web services.

2 Review of the Knowledge Portal on Language Technology

The Web-based information system LT World is a comprehensive knowledge portal for the large and multidisciplinary field of language technology (LT), a thriving hi-tech area with numerous applications for speech, text, dialogue, translation, and information retrieval. The LT World knowledge portal offers broad information on people, projects and organizations that deal with LT, collects research systems, tools, products and patents, lists events and news from research, development and market. In addition, LT World provides access to overview and background knowledge on 110 different language technologies.

To cover the complexity of this knowledge domain, the LT World system is based on a multidimensional ontology (see Figure below), which in addition to mere contents represents and supports central tasks needed within the acquisition and maintenance processes and moreover handles interoperability and user interfaces. The LT World ontology was designed to facilitate relevant functionalities and processes: (i) Acquisition (Gathering, Classification, Indexing) (ii) Maintenance (Efficient Storage, Consistency Control, Workflow) (iii) User interfaces (Access, Navigation, Presentation) (iv) Interoperability (Exchange, Search).

In the language technology field, some dimensions of classification are contributed by linguistics, others come in through the methodology of computer science. Additional to already existing classification schemes like Dublin Core, OLAC, IPC or BibTex that are supported in LT-World, a specific classification scheme covers six dimensions for language technology that are utilized in the ontology for most concepts:

- Application (e.g. Grammar Checking, Text Translation, Speech Dialogue

Systems, ...)

- Linguality (monolingual, bilingual, multilingual, translingual, language-independent)
- Languages / Language Pairs (e.g. Romanian, Thai, … / <en-fr>, <de-gr>,...)
- Technologies (e.g. Hidden Markov Model, Linear Programming, ...)
- Linguistic Area (e.g. Morphology, Syntax, Pragmatics, ...)
- Linguistic Approach (e.g. Two-Level Morphology, Systemic Functional Grammar, ...)

The first three dimensions of the LT classification scheme depend on the application. They describe the type of application, the linguality and the covered languages. The attribute Application takes as value a set of application types. The attribute Linguality describes the dependency of an application on a specific set of languages. Applications can be monolingual such as a grammar checker designed just for Finnish. They can be multilingual such as a text-to-speech product for Italian, French and Spanish. Translingual applications cross language boundaries. This is always the case for machine translation. However, there are also other applications carrying information across languages. An example is cross-lingual information retrieval, where a query is formulated in one language but relevant documents are (also) returned in other languages. Finally there exists a large number of language independent applications such as generic search engines or most speech compression programs.

The attribute Technologies takes values from a set of methods or techniques originating in computer science, mathematics, or electrical engineering. Linguistic Area is another attribute that adopts from the discipline of linguistic the levels of linguistic description in order to specify which aspects of language are covered by some project, publication, etc. The last attribute specifies the applied Linguistic Approach such as theories, models, or methods. More information on the LT-World ontology can be obtained from (Jörg & Uszkoreit, 2005; Uszkoreit et al., 2003).

In the context of LanguageGrid, the LT-World ontology will contribute to refine the language service ontology as defined currently. The LT-World ontology represents a fine-grained analysis of technologies in the NLP field (different approaches to part-of-speech tagging, constraint-based vs. statistical vs. rule-based parsing, etc.) as well as neighboring fields (knowledge representation, knowledge discovery, text mining, etc.) and is therefore a valuable addition to the language service ontology.

3 Conceptual Framework of the Upper Ontology

This section provides an overview of the conceptual framework of the proposed linguistic service ontology and reviews its top level.

3.1 Conceptual Framework

The two necessities for a linguistic service ontology can be summarized as below.

- Composition of composite linguistic services: A composite linguistic service that is tailored to particular user requirements should be composable by humans now and automatically composable in the near future. To make this possible, each technical component has to be described properly before service composition time. For example, if a process pipeline is to be composed of cascading existing linguistic services, the author has to check whether the input constraints for some linguistic services in the pipeline are compatible with the output of the immediately preceding linguistic service. The ontology should provide sufficient vocabulary for stating such input/output constraints. Furthermore, the preconditions, conditions that must hold when a service is being invoked, and the effects of the change of the states brought about by the service invocation of a linguistic service, should also be properly defined for any possible future configuration that utilizes automatic planning.
- Efficient wrapper generation: Wrapper programs have to be properly prepared and deployed in the language infrastructure in order to incorporate both existing linguistic services and recently developed language resources that are being disseminated. A linguistic service ontology will facilitate such a process. For example, a linguistic service provider who wants to disseminate a new language resource can benefit from the ontology; the relevant part of the taxonomy in the ontology provides guidelines for placing the service in the whole universe of linguistic services. Furthermore, the infrastructure may be able to provide a 'skeleton code' of the wrapper program for a major linguistic service type that has a position in the taxonomy. The service provider would be able to develop his/her own wrapper program code by simply refining the provided skeleton program.

Based on the above discussion about the necessities for a linguistic service ontology, the most important desideratum for it is that it be able to specify the input/output constraints of a linguistic service properly. The taxonomy of linguistic service can be at least partly designed by classifying the input/output constraints. Further desiderata for the ontology can be summarized as follows.

- Vocabulary to describe language resources and linguistic objects: The linguistic service ontology has to provide a vocabulary for describing both processing

elements and static language resources that are utilized by linguistic services. Furthermore, it should be able to encode linguistic objects on a variety of levels, such as lexicon entry, linguistic annotation, and even more abstract objects such as those that denote meaning.

- Conforming to technical standards: The linguistic service ontology should be compliant with relevant technical standards, especially for its formal representation, given that the major concern with the language infrastructure is interoperability of the technical components.

3.2 The Top Level of the Ontology

We have developed the upper part of the service ontology so far and have been working on detailing some of its core parts. We have adopted OWL as the working description language for the moment. This does not mean, however, that we will not be based on OWS-S or some other semantic web service description languages in the future. We use Protégé1 as the main tool for working with the service ontology. Figure 2 shows the top level of the proposed linguistic service ontology. In the figure and alike in this paper, a rectangle denotes a class, and a directed edge connecting classes indicates the relation between the classes. The label attached to an edge represents the type of the relation; among them, black edges with 'isa' labels depict the class hierarchy.

The topmost concept is **NL_Resource**, which is partitioned into **ProcessingResource**, and **LanguageResource**. Here, as in GATE (Cunningham, 2003), processing resource refers to programmatic or algorithmic resources, while language resource refers to data-only static resources such as lexicons or corpora. Although the details have not yet been worked out, meta-information for these two types of resources should be available. The innate relation between these two classes is: a processing resource can use language resources. This relationship is specifically introduced to properly define linguistic services that are intended to provide language resource access functions.

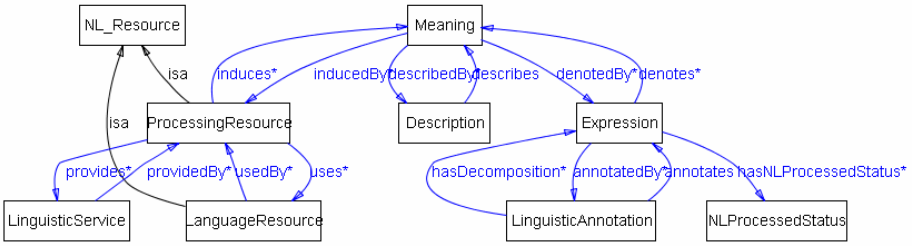

Fig. 2. Top level of linguistic service ontology.

[1] http://protege.stanford.edu/

As shown in the figure, **LinguisticService** is provided by a processing resource, stressing that any linguistic service is realized by a processing resource, even if its prominent functionality is accessing language resources in response to a user's query. It also has the meta-information for advertising its non-functional descriptions.

The fundamental classes, **Expression**, **Meaning**, and **Description** and the innate relations among them are also illustrated in Fig. 2. These play crucial roles in defining functionalities of some types of processing resources and associated language resources. As shown in Fig. 2, a language expression may denote a meaning, and the meaning can be further explained by a description, especially for human uses. These classes and the relations among them will be detailed in 4.3.

In addition to these, **NLProcessedStatus** and **LinguisticAnnotation** are important classes. NLP status is introduced especially to represent the so-called IOPE (Input-Output-Precondition-Effect) parameters of a linguistic processor, which is a subclass of processing resource. The results of a linguistic analysis are encoded as instances of a linguistic annotation class, and the instances are attached to the instance of the expression class. That is, an expression can be annotated by linguistic annotations. Respecting the recent direction of 'NLP as document annotation', we also consider that any linguistic analysis is additive. Issues with these two classes in the context of our linguistic service ontology are discussed in the next section.

4 Some Details of the Upper Ontology

This section details core parts of the proposed linguistic service upper ontology.

4.1 Taxonomy of NL Resources

A natural language resource can be either a language resource or a processing resource depending on its nature, as shown in Fig. 2. Figure 3 shows the taxonomy of the language resource class.

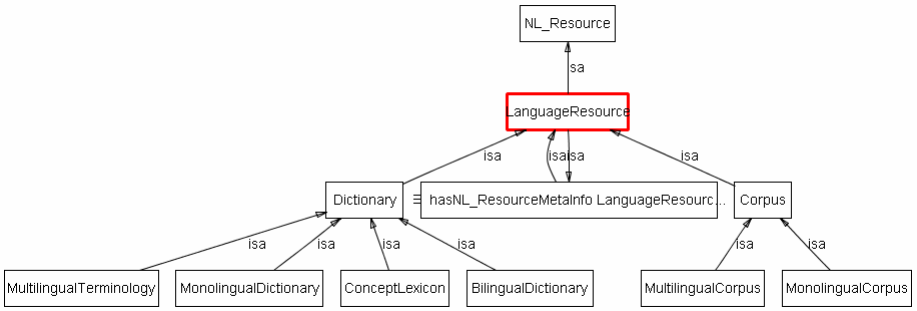

Fig. 3. Taxonomy of language resources.

As shown in Fig. 3, the language resource class is partitioned into subclasses for **Corpus** and **Dictionary**. The corpus class is divided into, for the moment, **MultilingualCorpus** and **MonolingualCorpus**. The immediate subclasses of the dictionary class are: (1) **MonolingualDictionary**, (2) **BilingualDictionary**, (3) **MultilingualTerminology**, and (4) **ConceptLexicon**. The major instances of (1) and (2) are so-called machine-readable dictionaries (MRDs). Many of the community-based language resources should fall into (3), including multilingual medical term lists. For subclass (4), we consider the computational concept lexicons, which have a WordNet-like structure2.

Figure 4 illustrates the top level of the processing resource taxonomy: it consists of the following four subclasses, which take into account the input/output constraints of processing resources and the language resources they utilize.

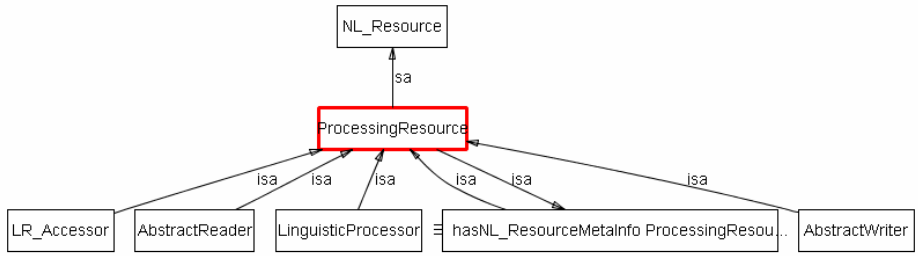

Fig. 4. Taxonomy of processing resources.

- **AbstractReader**: This class is introduced to describe computational processes that convert non-textual representation (e.g. speech) into textual representation (character strings). Speech-To-Text systems (STT; speech recognizer) are an obvious subclass of the abstract reader class. For a speech recognizer, its input

[2] This means that a whole concept system is organized as a network with a set of synset nodes, and the lexical semantic links that connect the synset nodes.

should have “audio” as its main MIME type.

- **AbstractWriter**: This class covers a class of computational processes whose processing direction is the opposite of the abstract reader class. This class is introduced to describe text to non-text conversion. Text-To-Speech (TTS) is the representative example. In parallel to the STT, TTS’s output should be audio.
- **LR_Accessor**: This class is introduced to describe language resource access functionalities. It is first partitioned into **CorpusAccessor** and **DictionaryAccessor**, depending on the type of language resource it accesses. Figure 5 shows the upward view of the **DictionaryAccessor** taxonomy. As shown in the figure, the input to a language resource accessor is a query (**LR_AccessQuery**), and the output of an accessor is a kind of ‘dictionary meaning’, which is divided into further sub-classes by referring to the taxonomy of a dictionary. Also stated in the figure is that a dictionary accessor uses dictionaries.

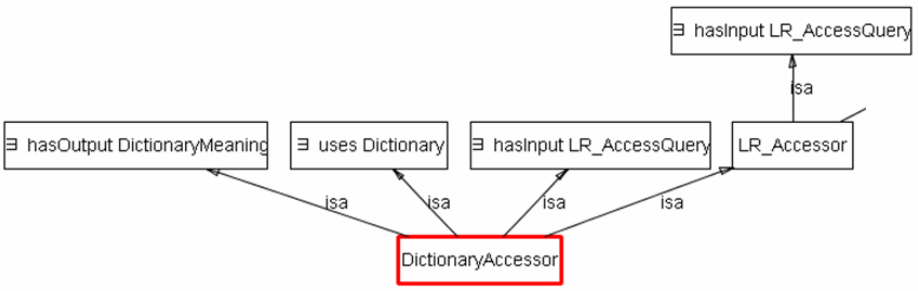

Fig. 5. Taxonomy of dictionary accessors (upward).

- **LinguisticProcessor**: This class is introduced to represent NLP tools/systems. Figure 6 shows the upward view of the taxonomy of **LinguisticProcessor**. As clearly stated in the figure, the IOPE parameters of a linguistic processor are constrained to the corresponding classes. The linguistic processor class is first partitioned into **Transformer** and **Analyzer**[3]. Figure 7 develops the taxonomy of the transformer subclass. Although it is not shown in Fig.7, the transformer class is subclass of an anonymous class whose **hasEffect** property is constrained to **Transformed** class, which is a subclass of **NLPStatus** class. A transformer somehow transforms the input expression into another expression: the meaning of the output could be either different[4] from the one denoted by the input (**MeaningNonPreservingTransformer**), or the same (**MeaningPreservingTransformer**), and the language of the input/output for a meaning preserving transformer could be either different (**Translator**) or the

[3] As seen from the current taxonomy, we have not yet considered any NLP tools that generate an expression from a representation at some level of linguistic representation.

[4] However, we still do not have such a ‘meaning transformer’.

same (**Paraphraser**). In Fig. 7, just to give an idea of how a low-level class that will represent a concrete software is defined, classes such as **J-to-E_Translator_1** and **J-to-J_Paraphrasor_1** are introduced.

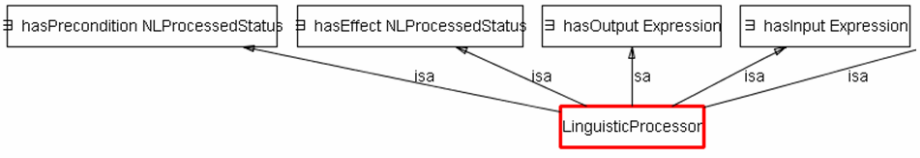

Fig. 6. Taxonomy of linguistic processors (upward).

4.2 Linguistic Analyzer and NLP Status

Figure 8 develops the taxonomy of the analyzer class. While it is not depicted in the figure, the input/output constraints of a linguistic analyzer are specified by the **Expression** class, while its precondition/effect parameters are defined by **NLProcessedStatus** class, as inherited from the linguistic processor class. The details are also not shown in this figure, these constraints are further restricted with respect to the taxonomy of the processing resource. We also assume that any linguistic analyzer adds some linguistic annotations to the input, as proposed elsewhere [9, 12]. That is, an analyzer working at a certain linguistic level (or 'depth') adds the corresponding level of annotations to the input. In this sense, any natural language expression can have a layered/multiple linguistic annotation. To make this happen, a linguistic service ontology has to appropriately define a sub-ontology for the linguistic annotations by itself or by incorporating some external standard. The linguistic annotation class is currently much too underspecified to import the international standards that will be specified in the near future. For the moment, it can only accommodate faceted attribute-value pairs and morph-syntactic annotations, such as constituent decompositions and syntactic dependencies.

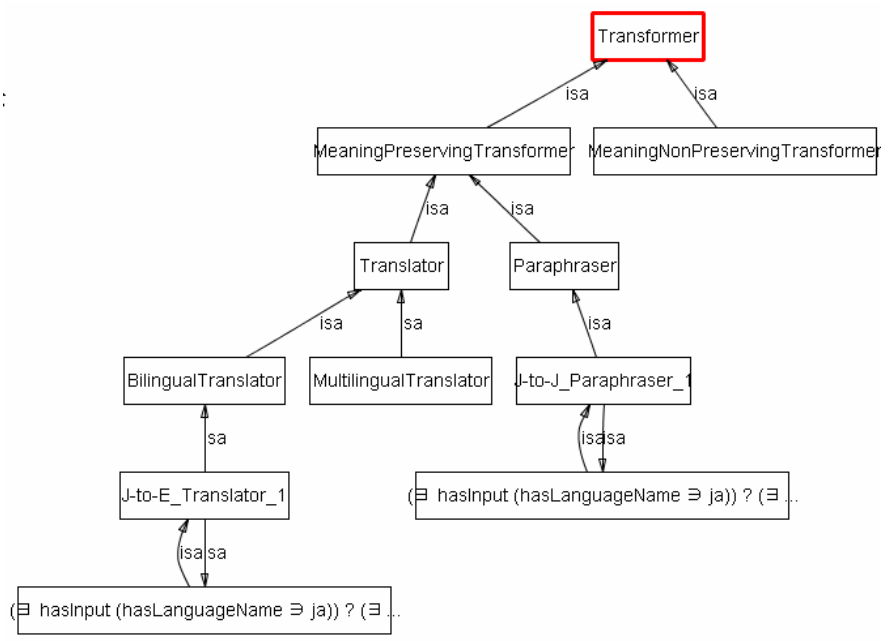

Fig. 7. Taxonomy of transformers.

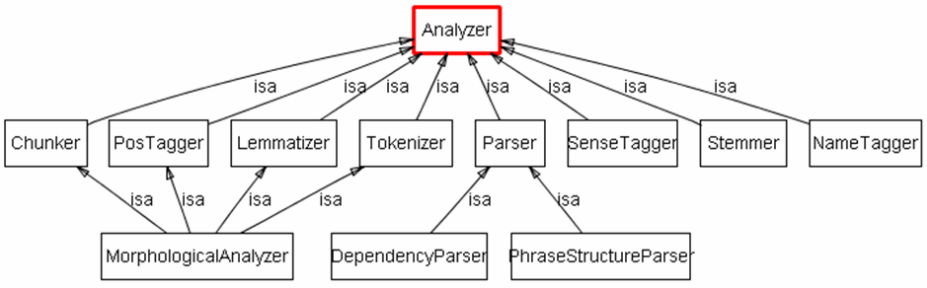

Fig. 8. Taxonomy of linguistic analyzers.

Figure 9 illustrates our working taxonomy of NLP status. Note that, in this figure, only the portion related to linguistic analyzer is detailed, and the sub-taxonomy rooted by **Transformed** is not shown. Benefits from the NLP status class will be twofold: (1) as a part of the description of a linguistic analyzer, we assign corresponding instances of this class as its precondition/effect parameters, (2) any instance of the expression class can be concisely 'tagged' by instances of the NLP status class, according to how 'deeply' the expression has been linguistically analyzed so far. Essentially, such information can be retrieved from the attached linguistic annotations. In this sense, the NLP status class might be redundant. Tagging an instance of expression in that way, however, is reasonable: we can define the input/output constraints of a linguistic analyzer concisely with this device. This should be effective, even when we adopt, for example, OWL, which is restricted in its expressiveness, as the ontology language.

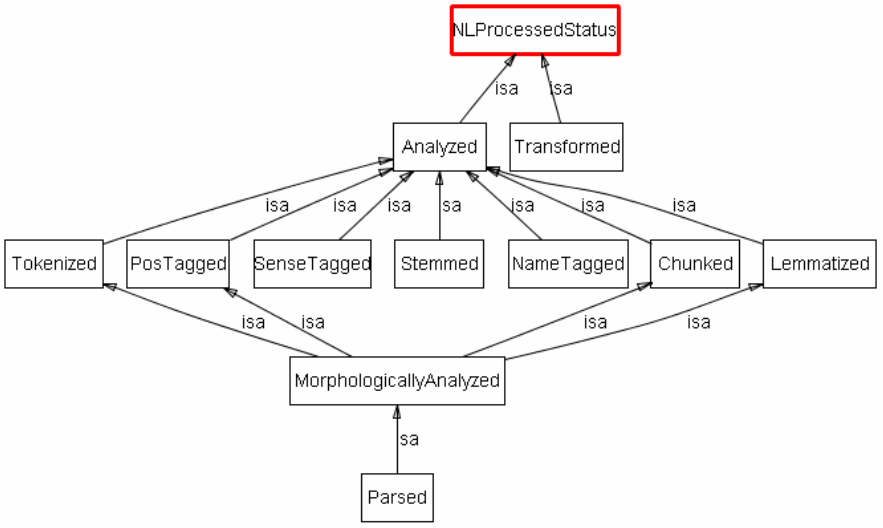

Fig. 9. Taxonomy of NLP states.

Each subclass in the taxonomy represents the type or level of a linguistic analysis, and the hierarchy depicts the relations among them. For example, if an expression has been parsed, it would already have been morphologically analyzed because parsing usually requires the input to be morphologically analyzed beforehand. Just as easily imagined, the taxonomy is partly isomorphic to the type for the linguistic analyzers shown in Fig. 8.

Note that the taxonomy in Fig. 9 is only partial and preliminary. The arrangement of the subclasses within the hierarchy may end up being far different, depending on the languages considered[5] and the actual NLP tools that are at hand. For example, the notion of 'chunk' is different from language to language. Also, NLP tools are essentially idiographic. However, if we go too far in this direction constructing a taxonomy would be meaningless and we would forfeit reasonable generalities. We probably need to seek a modest taxonomy that is sufficient for describing actually available NLP tools.

4.3 Expression, Meaning, and Description

Any actual natural language expression of arbitrary length can be represented as an instance of the **Expression** class. We also introduced the **Meaning** class in order to represent some kind of meaning that may be denoted by an expression. Here we are not going into the philosophical question: "what is a meaning?" Instead, we simply assume that any linguistic expression denotes 'some' meaning. Sometimes just the existence of the meaning class instance is sufficient for expressing that

[5] Linguistic analyses can be highly language dependent: this point was made by Thierry Declerck.

"there is surely some meaning, but the details are not clear." This kind of placeholder is useful when we are to represent two different expressions that share the same meaning, which is a characteristic of the meaning preserving transformer. The meaning class will be further refined by introducing subclasses based on an abstract meaning representation formalism, or more concrete information presented in a language resource such as a dictionary or even in more general knowledge resource such as an encyclopedia. Currently we only have **DictionaryMeaning** as a subclass of the meaning class, as discussed in the next subsection.

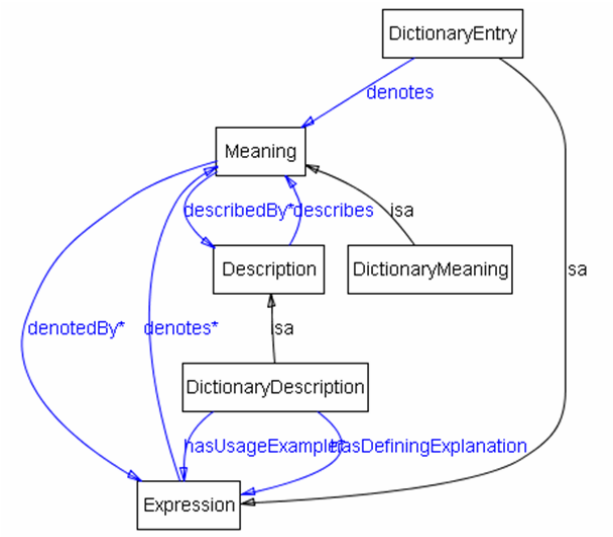

Fig. 10. Relations among **DictionaryEntry**, **DictionaryMeaning**, and **DictionaryDescription**.

The **Description** class is introduced to encode additional information that may be useful for a human user to better understand the meaning associated with an expression. That is, an instance of the description class is always attached to an instance of the meaning class. Because, for the moment, we only have the dictionary meaning as a subclass of the meaning class, we only have **DictionaryDescription** as a subclass of the description class. The dictionary meaning class and the dictionary description class are further refined in concordance with the taxonomy of the dictionary class.

Figure 10 summarizes the innate relations among the three classes described above. Note that the dictionary entry class, a subclass of expression class, is introduced for representing a dictionary entry, which is defined as a set of: a canonical written form, a pronunciation, possibly a part-of-speech, and a word sense number.

4.4 Dictionary Meaning, Dictionary Description, and Dictionary Model

Following the previous discussion, one question might arise: which information items in a dictionary entry should be considered elements of the meaning class, and which elements of the description class? This question is important, because the issue is closely related to the topic of dictionary modeling, which has been considered crucial in the context of lexical resource interoperability.

Table 1. Classification of dictionary information items.

Dictionary Class	Meaning	Description
Monolingual Dictionary	a set of synonyms	explanations, usage examples, etymological information, etc.
Bilingual Dictionary Multilingual Terminology	a set of translations	usage examples, derivational forms, collacations, etc.
Concept Lexicon	synset, lexical semantic relations	gloss, usage examples

Our answer to the question is summarized in Table 1, which classifies the relations between the dictionary class and the information items with respect to the meaning/description distinction. The fundamental idea here is that a set of expressions that (seem to) denote a single meaning or a lexicalized concept, is classified as an element of the meaning class, whereas other information items are classified as elements of the description class. For example, as shown in Fig.10, usage example and gloss (defining expression) are given by instance of the expression class. For concept lexicons like WordNet, we also consider the associated lexical semantic relations to be part of the meaning class, rather than part of the description class.

This distinction is in concordance with the three-layered (lemma/meaning/concept) dictionary meta-model recently proposed by (Hayashi, 2006). The lemma layer there is represented by the dictionary entry class here, whereas items associated with meaning and concept layers there are represented by instances of the meaning and the description classes as summarized in Table 1.

5 Relevant International Standards

DFKI is strongly involved in standardization activities of the ISO committee TC37/SC4 on the management of language resources. This takes also partly place in the context of the European eContent project LIRICS (see http://lirics.loria.fr). LIRICS is about the development of a **L**inguistic **I**nf**r**astructure for **I**nteroperable **R**esour**c**es and **S**ystems. More specifically LIRICS is dealing with NLP and MRD lexicons, morpho-syntactic, syntactic and semantic annotation.

ISO TC37/SC4 is also aiming at having a common background for the 4 representation frameworks mentioned just above. This is realized on the one hand by the use of already existing recommendations of the World Wide Web Consortium (W3C), like XML (Bray et al., 2004), RDF (Beckett, 2004), OWL (McGuinness and van Harmelen, 2004). On the base of those recommendations the ISO TC37/SC4 initiative "Linguistic Annotation Framework" (LAF) is proposing an unified base for the annotation of linguistic data (see Ide & Romary, 2006), which is being specialized for the four domains mentioned above: the lexicon (Lexical Mark-Up Framework – LMF), morpho-syntax (Morpho-Syntactic Annotation Framework – MAF)), syntax (Syntactic Annotation Framework – SynAF) and semantics (Semantic annotation Framework – SemAF).

We should stress at this place that the work of LIRICS is directed towards offerings standardization for linguistic *annotation*, as the different levels of linguistic analysis. LIRICS is thus not about a theory of linguistics, and is also not proposing an ontology of the linguistic domain. But while LIRICS is aiming mainly at overcoming the incompatibilities of distinct annotation tagsets proposed in the context of various projects and initiatives, there is a need for grounding the proposals for interoperability of annotation strategies on a stable and acknowledged understanding of "what" should be annotated. LIRICS (and generally the ISO TC37/SC4 activities) propose for this purpose so-called meta-models for the distinct annotation frameworks: The meta-models are a XML format that allows the description of linguistic annotation as such. Already existing annotation formats can be mapped onto those meta-formats for lexical, morpho-syntactic, syntactic and semantic annotation.

The elements used in the meta-models for concretely annotating the language resources are introduced and defined as so-called data-categories (which are also being standardized within a special Working group within ISO TC37, see Ide & Romary, 2004). To avoid the uncontrolled expansion (and possible redundancies) of data-categories, those are stored in a specific data-category repository and can be edited by the community in a supervised manner.

Data-categories are obligatorily associated with a specific definition, similar to the comments in ontologies or the glosses in the WordNets.

The list of data-categories is proposed as a flat list, in order to make them immediately accessible, without "relational" commitments, as those are intrinsically presented in structured presentation. Two aspects are problematic in this strategy:

- The data-categories (and the meta-models that can use them) are not available in a distributed manner, but only in a centralized repository. It is wishful to be able to aggregate data-categories that are distributed over the web, and to have a way for checking automatically their compatibility for a specific annotation task.
- Developers and applications searching for tags/data-categories can fetch only a flat list, although they need structured representations for their annotation purposes. There is need to have either a basic ontology describing the possible relations between the data-categories, or, still more ambitious, to have a method for building tailored linguistic ontologies on the base of the information encoded in the data-category repository (and maybe data-categories found in a distributed manner).

We see in those two points a clear place at which to organize further cooperation between DFKI and the Language Grid initiative. The latter would among others provide for central contributions on the methodologies for accessing large collection of linguistically annotated data via distributed computing capabilities, whereas access (and aggregation) is being directed by a linguistic ontology (or various tailored linguistic applications ontologies). The joint development of a linguistic ontology, and its operational deployment, is a central point of collaboration between our institutes in the LanguageGrid project.

6 Outlook

The development of the language service ontology is central to the Language Grid project as it ties in with the development of a truly distributed platform of autonomous NLP components than can be automatically integrated based on semantic work flow descriptions. Semantic Web Service access to semantically annotated NLP components based on the language service ontology will allow for advanced multilingual Semantic Web applications such as the next generation of machine translation systems that can easily mix-and-match NLP components for different languages and on different levels on-demand. DFKI and the Language Grid initiative can extend cooperative work on this based on the experiences of the

HoG web service architecture for NLP components, the LT-World ontology of NLP technologies and the ISO/LIRICS work on linguistic data categories. Also the joint development of a linguistic annotation ontology, and its operational deployment, is a central point of collaboration.

References

1. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP tools and applications. Proceedings of ACL2002, pp.168-175. (2002).
2. Deborah L.M, Harmelen, F.V. (eds.): OWL Web Ontology Language Overview. W3C Recommendation, http://www.w3.org/TR/2004/REC-owl-features-20040210/ (2004).
3. Declerck, T.: Extending NLP Tools Repositories for the Interaction with Language Data Resource Repositories. Proceedings of ACL/EACL Workshop on Sharing Tools and Resources for Research and Education. (2001).
4. Declerck, T., Buitelaar, P., Calzolari, N., Lenci, A.: Towards A Language Infrastructure for the Semantic Web. Proceedings of LREC2004, pp.1481-1484. (2004).
5. EDR: EDR Electronic Dictionary Technical Guide. http://www2.nict.go.jp/kk/e416/EDR/ (2003).
6. Fellbaum, C. (eds.): WordNet: An Electronic Lexical Database. MIT Press. (1998).
7. Francopoulo, G., George, M., Calzolari, N., Monachini, M., Bel, N., Pet, M., Soria, C.: Lexical Markup Framework (LMF). Proceedings of LREC2006, pp.233-236. (2006).
8. Hayashi, Y., Ishida, T.: A Dictionary Model for Unifying Machine Readable Dictionaries and Computational Concept Lexicons. Proceedings of LREC2006, pp.1-6. (2006).
9. Ide, N., Romary, L.: International standard for a linguistic annotation framework. Journal of Natural Language Engineering, 10:3-4, pp.211-225. (2004).
10. Ishida, T.: Language Grid: An Infrastructure for Intercultural Collaboration. Proceedings of SAINT2006, pp.96-100. (2006).
11. Jörg, B., Uszkoreit, H.: The Ontology-based Architecture of LT World, a Comprehensive Web Information System for a Science and Technology Discipline. In: Leitbild Informationskompetenz: Positionen - Praxis - Perspektiven im europäischen Wissensmarkt. 27. Online Tagung (zugleich 57. Jahrestagung) der DGI. Frankfurt am Main, 23.-25. Mai, 2005.
12. Klein, E., Potter, S.: An ontology for NLP services. Proceedings of LREC

Workshop on a Registry of Linguistic Data Categories within an Integrated Language Resource Repository Area. (2004).
13. Murakami, Y., Ishida, T., Nakaguchi T.: Infrastructure for Language Service Composition. Proceedings of SKG2006. (2006).
14. Romary, L (eds.): Towards a Data Category Registry for ISO TC37. http://www.tc37sc4.org/new_doc/ISO_TC_37-4_N133_DCR_for_TC37.pdf (2004)
15. Schäfer, U.: Middleware for Creating and Combining Multi-dimensional NLP Markup. Proceedings of the EACL2006 Workshop on Multi-dimensional Markup in Natural Language Processing. (2004).
16. Schäfer, U. Heart of Gold User and Developer Documentation. DFKI, Saarbrücken, 2005.
17. Lionel Clément and Éric de la Clergerie. Maf: a morphosyntactic annotation framework. In Proceedings of the 2nd Language and Technology Conference (LT'05), pages 90_94, Poznan, April 2005.
18. Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview. URL: http://www.w3.org/TR/2004/REC-owl-features-20040210/, February 2004.
19. Dave Beckett. RDF/XML syntax specifcation (revised) http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/, February 2004.
20. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (XML) 1.0 (third edition). URL: http://www.w3.org/TR/2004/REC-xml-20040204/, February 2004.
21. Thierry Declerck. SynAF: Towards a standard for Syntactic Annotation, Proceedings of LREC 2006
22. Nancy Ide and Laurent Romary. Representing Linguistic Corpora and Their Annotations, Proceedings of LREC 2006.
23. Paul Buitelaar, Thierry Declerck, Anette Frank, Stefania Racioppa, Malte Kiesel, Michael Sintek, Ralf Engel, Massimo Romanelli, Daniel Sonntag, Berenike Loos, Vanessa Micelli, Robert Porzel, Philipp Cimiano. LingInfo: Design and Applications of a Model for the Integration of Linguistic Information in Ontologies. In: Proc. of OntoLex06, a Workshop at LREC, Genoa, Italy, May 2006.
24. The OWL Services Coalition: OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.html, 2003.