Elektrotehniški vestnik 75(5): 285-292, 2008
Electrotechnical Review: Ljubljana, Slovenija

Classification of the language group characteristics into a multi-layered architecture of the interlingua interpreter in multilingual translation

Grega Jakus, Sanida Omerović, Tatjana Filimonova, Sašo Tomažič

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška cesta 25, 1000 Ljubljana, Slovenija
E-pošta: grega.jakus@fe.uni-lj.si, sanida.omerovic@lkn1.fe.uni-lj.si, tatjana.filimonova@fe.uni-lj.si, saso.tomazic@fe.uni-lj.si

Abstract. The paper addresses the issue of complexity of development of the interlingua interpreters in multilingual translation when developing the interpreters of related languages. The development cost using the interlingua approach is proportional to the number of languages needed to be interpreted in. The paper presents a design of a modular interpreter architecture whose characteristic is the ability of being layered into several levels of abstraction. A level of abstraction refers to the degree of abstractness of language structures that enter a certain module as data and on which transformations are performed. We try to place characteristics of groups of natural languages in the dislayered architecture of interpreters. The use of this concept reduces the cost of development for enabling some of the modules to be reused when developing the interpreters of related languages. The idea of abstract dislayering is demonstrated with an example of interpretation of e-speranto in English and Slovene. E-speranto is a computer language intended for recording multilingual documents on the Web and also serving as an intermediate language in multilingual translation.

Key words: interlingua, multi-level abstraction, interpreter, e-speranto, multilingual translation

Umestitev lastnosti jezikovnih skupin v večslojno arhitekturo tolmača vmesnega jezika pri večjezičnem prevajanju

Povzetek. Članek obravnava problem zahtevnosti (cene) izdelave tolmačev vmesnega jezika pri večjezičnem prevajanju. Pri prevajanju z uporabo vmesnega jezika je cena izdelave tolmačev sorazmerna številu jezikov, v katerih želimo tolmačiti. Predstavljena je zasnova modularne arhitekture tolmačev, za katero je značilna razslojenost na več nivojev abstrakcije. Nivo abstrakcije se nanaša na stopnjo abstraktnosti jezikovnih struktur, ki v nek modul vstopajo kot podatki in nad katerimi se izvajajo transformacije. V razslojeno arhitekturo tolmačev skušamo umestiti lastnosti skupine naravnih jezikov. Uporaba predstavljenega koncepta zmanjša ceno izdelave tolmačev, saj lahko pri izdelavi tolmačev sorodnih jezikov ponovno uporabimo nekatere module. Ideja abstraktne razslojenosti je prikazana na primeru tolmačenja e-speranta v angleščini in slovenščini. E-speranto je računalniški jezik, ki služi za zapis večjezičnih besedil na svetovnem spletu in kot vmesni jezik pri večjezičnem prevajanju.

Ključne besede: vmesni jezik, večslojna abstrakcija, tolmač, e-speranto, večjezično prevajanje

1 Introduction

About 6900 languages that are spoken in the world [1] are divided into language groups and subgroups. The division is based on a common predecessor (the so-called proto-language), from which individual languages have developed in different ways due to various geographical and political factors. With the increasing use of Internet, the interaction among the members of different language communities is becoming more and more intense. However, this interaction is limited due to the so-called language divide. The majority of the world population namely speaks only their mother language, and only a minority speaks an additional or two foreign ones. That is why most of the Web contents are presented only in the big world languages, whereby English is still the prevailing choice.

Several commercial translators of natural languages are already available on the market, for example [2], [3], [4], [5], but again only the big world languages are supported. In order to develop translators for all the language pairs, about 47,610,000 of such units should be made. A much more appropriate approach than direct translation is the use of an intermediate language, thus making the number of translators proportional to the number of languages in which a selected content is to be accessed.

In this paper we propose an architecture designed for interlingua interpreters that further reduces the cost of developing a multilingual system. The approach is

Received 6 October 2008
Accepted 12 December 2008

based upon a modular architecture of the interpreters and arrangement of modules in layers. The modules in different layers differentiate by how “near” to the natural language are the data structures that enter the modules as data and upon which the operations are performed. Some languages are related. Their grammar and syntax are analogous. These languages usually have a common ancestor, from which some characteristics are preserved in spite of the evolution. If the characteristics of a group of languages are separated from those specific to an individual language, such “stratification” of the language characteristics can be linked with a dislayered architecture of interpreters. Treatment of common characteristics can be incorporated in a module common to all the languages mentioned. By doing so the cost of the development of interpreters for a certain group of languages is reduced.

2 Interlingua and its interpretation

An intermediate language or interlingua is an abstract presentation of the content that is independent from any natural language [6]. The record in interlingua must contain the whole information required for generating text in a natural language. Thus, the entire meaning we want to express in a natural language must be captured in interlingua.

The advantage of using the interlingua is a two-phase course of translation between two natural languages. During the process, the modules that perform the conversion from a natural language to the interlingua (translators) are independent of those that perform the opposite conversion (interpreters). Moreover, the interpreters and translators of different languages are also mutually independent. The effect of this independence is the reduction of the number of units that would be needed in case of direct mapping among the individual languages. The cost of the latter approach is as high as $n(n-1)$, where n denotes the number of languages among which we want to translate. By using the interlingua approach, the cost of the interpreter development reduces to $2n$, since only a translator and an interpreter for each language must be made.

In the past, numerous attempts of creating an interlingua that would be truly universal and independent of natural languages were conducted. In most cases it was established that it is difficult to determine and present the meaning in a text. The majority of interlinguas has a language-independent structure and a vocabulary (a set of contained concepts) that is not entirely independent of natural languages. Some more notable implementations of interlinguas are presented in the next paragraphs.

DLT (*Distributed Language Translation*) [7] was a project of the development of a multilingual system in the 1980s that used an adapted version of Esperanto as an interlingua. The document written in Esperanto would be carried over the network and inerpreted in a chosen language by the target computer. Although DLT presented a novel and interesting approach to machine translation, the results were not promising in practice.

The interlingua KANT [8] is based upon controlled English (a language with a limited scope of vocabulary) and was created with the intention of translating technical documentation. Its interpretation produces very accurate sentences, but due to the limited field of use it is not directly applicable for general multilingual translation.

UNL (*Universal Networking Language*) is a computer language for recording and exchanging information [9] and it is basically intended for communication on the Web. It supports 15 languages, which makes it currently the largest multilingual system intended for use on the Web. Its main deficiency is the limited power of expressiveness [10] and poor intelligibility of texts written in this language, which already proved as a disadvantage during the development of Internet standards in the past.

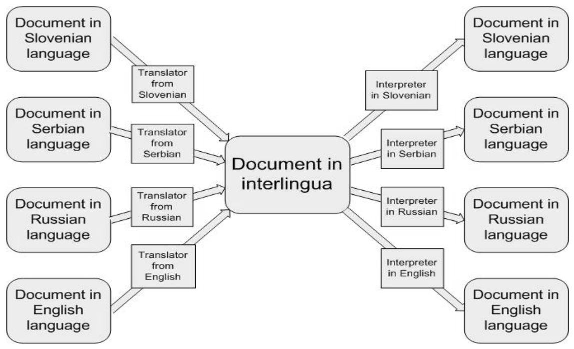
Document in Slovenian language
Translator from Slovenian
Document in Serbian language
Translator from Serbian
Document in Russian language
Translator from Russian
Document in English language
Translator from English
Document in interlingua
Interpreter in Slovenian
Interpreter in Serbian
Interpreter in Russian
Interpreter in English
Document in Slovenian language
Document in Serbian language
Document in Russian language
Document in English language

Figure 1: Translation by using the interlingua approach takes place in two phases. In the first phase, the translators perform the conversion from a natural language to the interlingua. In the second phase, the interpreters perform the opposite conversion. By using the interlingua, the cost of the development reduces to *2n*, since only a translator and an interpreter for each language must be made.

Generation of a text from an interlingua can be carried out in different manners [6]. Most often it is based on language rules (*rule-based*) that define the conversion from a source presentation to the target one. Another widely used approach is based upon the semantic and pragmatic knowledge of a certain field (*knowledge-based*). The *Statistical* and *example-based* approach are not used widely in generation of texts from an interlingua because they both require a bilingual corpus which is, however, hard to create when one of the languages is an interlingua.

The majority of the approaches of the rule-based natural language generation from the intermediate abstract presentation has a modular design with two basic steps of conversion. They represent the lexical transformation (conversion of the interlingua concepts into language units of a natural language) and structural transformation between the language structures in both languages [6]. An example of a well-established framework that uses such design is ARIANE [11]. ARIANE is a flexible framework for the development of machine translation systems between language pairs which can also be an interlingua and a natural language. The system separates the algorithms from the linguistic content as the parameterization of algorithms. Several interpreters of the interlingua are based on ARIANE, for example [12], [13]. The multi-layered architecture introduced in the continuation is in fact an upgrade of the approach used in the ARIANE system.

3 E-speranto

E-speranto, also named HTDL (*Hyper Text Description Language*) [14], [15], is a design of a formal computer language for recording multilingual texts, which can also act as an intermediate language for multilingual translation. The field of use of e-speranto is Web communications. Its advantage over similar approaches (e.g. UNL) is especially the intelligibility of documents both to computers as well as to people. The latter already proved as an advantage many times in the past in the development of various Internet protocols. The basic syntax of e-speranto is based on the extendable markup language (XML), while the grammatical rules are based on Esperanto [16].

Unlike Esperanto (and some other multilingual systems, e.g. DLT), the grammatical characteristics in e-speranto are expressed explicitly by means of metadata, as this is more suitable for computer handling. XML is compatible with HTML (*Hyper Text Markup Language*), which enables the inclusion of e-speranto into webpages. E-speranto[1] is a computer language whose functionality can be classed within the presentation layer of the ISO-OSI (*International Standards Organization - Open System Interconnect*) model.

4 Multi-level abstraction of procedures and language representation structures in the interpreter

Since interlingua is an abstract representation of the content, the whole specificity of the language must be contained in the process of interpretation in a target, i. e. natural language. The most frequent approach to natural language generation is the use of language-independent (generic) algorithms and language-specific rules that act as the parameters of these algorithms. The advantage of such an approach is obvious. By using general algorithms and language-specific rules, we can theoretically make interpreters for different natural languages by parameterizing the same algorithms with a linguistic content of the language we want to interpret in.

On the other hand, the efficiency of such an approach is questionable. We find the following deficiencies:

- It is very hard to determine the algorithms that would be so general to translate language structures from the abstract intermediate form to any target language. Such mappings would have to consider all particularities of all target languages;
- Even if we could define such mappings, the interpretation would contain many redundant steps, because the majority of procedures would not have any impact on certain languages (e.g. determining the case specifying extensions in English is meaningless, since English does not use parts of speech inflection for this purpose);
- The interpreter with all implemented procedures would be very extensive and difficult to maintain;
- With real-time interpretation, such as interpretations of Web pages, the redundant procedures would cause a longer response time and as a consequence worse user experiences.

```
<sentence original="E-speranto is a design of a computer language."
                  feelings="declarative" organization="simple">
  <subject detail="personal_name" number="singular">
    <word>E-speranto</word>
  </subject>
  <predicate detail_predicate="main" mood="indicative" voice="active"
                  tense="present" person="third">
    <word>be</word>
    <subordinate>
      <predicate detail_predicate="predicate_noun" number="singular">
        <word>design</word>
        <subordinate>
          <object detail_object="of_genitive" number="singular">
            <word>language</word>
            <subordinate>
              <attribute>
                <word>computer</word>
              </attribute>
            </subordinate>
          </object>
        </subordinate>
      </predicate>
    </subordinate>
  </predicate>
</sentence>
```

Figure 2: Record of a sentence in e-speranto. The basic building element in e-speranto is a clause. A clause is a semantic unit that corresponds to a sentence in a natural language. Clauses are composed of sentence elements introduced by XML tags. The grammatical characteristics are expressed explicitly by means of XML attributes. The concepts representing the essence of e-speranto are marked in English for the sake of better intelligibility.

To avoid the deficiencies mentioned, we suggest dislayering of interpreters in several layers (Figure 3). The individual layers contain modules with procedures that are applied on the language structures which are on

[1] *The e-speranto project is in the development stage, that is why its specifications are subject to change. The current specifications are available on the Web page http://www.e-speranto.org.*

the same level of abstraction according to the target natural language. In every phase of interpretation, execution is carried between modules on different layers, whereby a module on a higher level is compatible with several modules on a lower level. In general, a module on a higher level can contain the content (algorithms and language rules) that would otherwise be common to several distinct modules (e.g. for different languages of interpretation) on a lower level.

Some typical procedures on individual layers can be identified. The first phase of abstraction contains the procedures that are in general characteristic of machine translation. The procedures are language-independent and in general dictate the course in which the interpretation is executed. An example of the modules on this level are the two modules that separate the interpretation on the transformation of concepts from an interlingua to language units in a natural language (lexical transformation) and the structural transformation of the intermediate representation structure to its target form.

Actual realizations of the specified transformations take place in the next layers of abstraction. By passing through the layers, the transformations become more and more language-specific. The last layer consists of modules that perform the processing on the level of individual languages. These modules can also access dictionaries and are parameterized with the language rules of the target natural language.

The features of the described approach are:

- High modularity of the system;
- Every module has its specific place in a certain layer of abstraction and phase of interpretation;
- The modules combine procedures that perform specific content-related transformation of the language structures (e.g. the rearrangement of the tree edges in accordance with the word order in a natural language determined by sentence inclination);
- As the modules need to be interconnectable, this approach requires a uniform definition of the interfaces through which the modules are connected.

The described approach actually implies segmentation of algorithms over the abstract layers in dependence on "how close" the language structure that is the subject of processing is to a target language. In this process, every algorithm in principle remains language-independent; however, a set of algorithms that is used for interpretation in an individual language becomes language-dependent. For this purpose, a set of modules needed for the interpretation of interlingua in a natural language must be specified before the beginning of the interpretation process.

By using this approach to organization of the building blocks of the interpreter, the paradigm of the language-specific rules as the parameters of generic algorithms is preserved. The use of the paradigm is not obligatory, since a module on a certain abstraction level can be completely specific for a particular intention of use. For instance, if the properties of a certain language distinguish it completely from other languages for which the modules already exist, we can make a completely specific module if it is more convenient from the developmental point of view or the point of view of efficiency.

The advantages of dislayering the interpreter can be exploited provided the content that can be placed in individual layers is identified. In the next chapter, separation (abstraction) of the characteristics of a group of languages from the actual characteristics of individual languages in the group is suggested as one of the possible ways of classification.

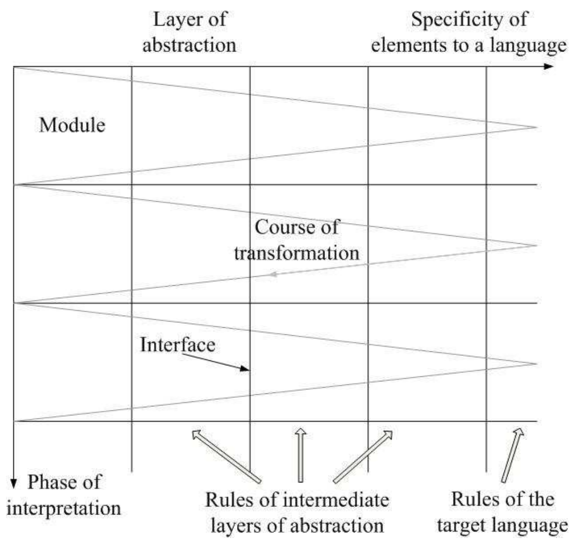
Layer of abstraction
Specificity of elements to a language
Module
Course of transformation
Interface
Phase of interpretation
Rules of intermediate layers of abstraction
Rules of the target language

Figure 3: Dislayering of the interpreter based on abstractness of data structures on which the procedures in individual modules are applied. The procedures perform specific content-related transformation of the language structures and are parameterized with language rules. Interpretation takes place in several phases. A module on a higher level of abstraction passes the execution to lower levels which process a more and more language-specific content. The algorithms on individual levels use rules that become more specific with the descending degree of abstraction.

5 Placement of natural languages into the layers of the interpreter

Despite the common misconception that there exists only a handful of language groups, languages can be roughly divided in some ten language families that are further divided in subgroups. The Roman, Slavic and Germanic subgroups that include the majority of European languages are subgroups of Indo-European languages. Some features of the Slavic subgroup, for which the e-speranto is intended in the first place, are:

- *Fusional morphology* (Slavic languages have the tendency to form new words by adding one or several different morphemes to the already existing words);
- *Preservation of cases from the proto-Indo-European language* (most Slavic languages have seven cases);
- *Difference between the perfect and imperfect verb aspect*;
- *Inflection of parts of speech* (agreeing in tense, inclination, person, number, gender, case, etc.).

The dominant language in the world of electronic communications is English, which belongs to the Germanic subgroup of the Indo-European languages. If we compare the features of English with the already presented features of the Slavic languages, we can state that:

- English has minimal inflection of parts of speech (inflection is mostly replaced with changes in the word order or with the use of other parts of speech, e.g. prepositions);
- Unlike the so-called *synthetic languages* (e.g. the Slavic languages) which use morpheme inflection to express different notions, English is an *analytic language,* in which individual language units are usually made up of a single morpheme.

It is not our intention to present the similarities and differences between individual languages in detail, but to emphasize that we can make use of the resemblances in the development of interpreters. The characteristics that are common to a certain group of languages can be introduced on an abstractly higher level than the actual characteristics of individual languages in this group. The abstraction made by separating the common characteristics from the specific ones can be connected with the dislayered architecture of interpreters.

6 INES –Implementation of the dislayered interpreter

We relied on this architectural approach when making an e-speranto interpreter INES (*INterpreter of E-Speranto*). For this purpose we used two programming languages with different programming paradigms. The programming language Java, in which the connection with the Internet[2] was made, employs the *object-oriented paradigm*. The core of the interpreter was written in the symbolic programming language Mathematica, which is well known for the *rule-based programming* and *symbolic pattern matching*. To simplify the initial development, we focused on the interpretation of simple sentences (i. e. sentences with only one verb). Despite this limitation, the interpreted languages are not deprived of their expressiveness, since it is possible to express almost anything with simple sentences.

Because e-speranto is a computer language, we can draw some parallels with the interpreters and translators of computer programming languages regarding the interpretation. Namely, the process of interpretation is similarly divided into several stages. We distinguish:

- lexical and syntactical analysis of the source code,
- generation of the intermediate code,
- code optimization, and
- compilation phase.

The lexical and syntactic analyses are performed every time during the composition of a document in e-speranto and are provided by the development environment. For this purpose the development environment based on the Eclipse platform was developed. The built-in XML editor performs the verification of the document conformity with the e-speranto grammar, syntax and vocabulary.

The other phases of development are realized in the INES interpreter. The conversion of the e-speranto document into the expressions of the Mathematica language is analogous to the generation of the intermediate code. The phase of code optimization in the classical translators corresponds to the adaptation of the intermediate representation structure to the form that is used in the process of interpretation in INES (compare Figures 2 and 5). The phase of optimization is important since it enables, to a certain degree, the independence of the tree structures in INES from the changing grammar and syntax of e-speranto, as the latter is still being developed.

The compilation phase is the main step of the interpretation in a selected natural language. As is the case with the majority of the interpreters of the interlingua, in INES this phase is also realized in two steps. In the first step, the replacement of e-speranto concepts and their attributes with the words in a target language (the so-called lexical transformation) takes place, while the structural transformation is performed in the second step. The interpretation in both steps is carried out with the modules that are arranged into three layers of abstraction. The dislayered architecture of INES with some distinctive procedures on individual layers is shown in Figure 4.

The first layer comprises modules that dictate the course of interpretation and are independent of the target language. The layer is only aware of the fact that a sentence in a natural language contains the elements that express an action or activity (i. e. the predicate) and the holder of this action or activity (i. e. the subject). This layer also contains the algorithms for movements in the tree structure. Among the various possible methods the top-bottom, left-right approach is

[2] *The interpretation of simple sentences can be tried out on e-speranto's Web page.*

implemented in INES. The individual subtrees are identified according to their type; their transformation is then performed by the lower layers.

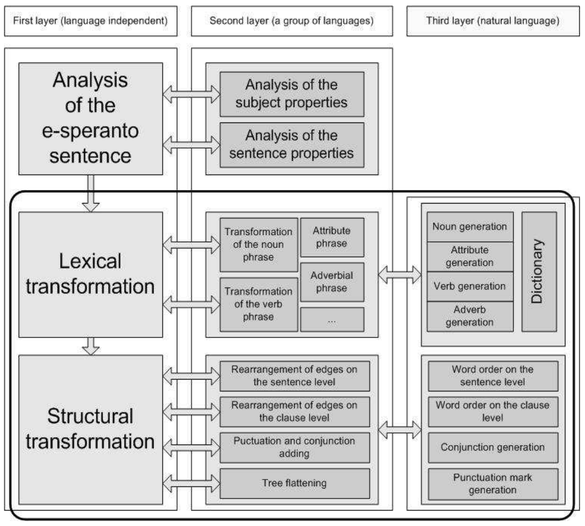
First layer (language independent)
Second layer (a group of languages)
Third layer (natural language)
Analysis of the e-speranto sentence
Analysis of the subject properties
Analysis of the sentence properties
Lexical transformation
Transformation of the noun phrase
Attribute phrase
Transformation of the verb phrase
Adverbial phrase
...
Noun generation
Attribute generation
Verb generation
Adverb generation
Dictionary
Structural transformation
Rearrangement of edges on the sentence level
Rearrangement of edges on the clause level
Puctuation and conjunction adding
Tree flattening
Word order on the sentence level
Word order on the clause level
Conjunction generation
Punctuation mark generation

Figure 4: Scheme presenting operation of INES. INES is an implementation of a dislayered interpreter with the architecture presented in the Figure 3. In INES, every phase of interpretation is divided into three layers. The procedures in the modules in the first layer perform language-independent operations that are common to rule-based machine translation. The procedures in the second layer are typical of a group of languages, while the ones in the third layer are language-specific.

The modules in the second layer are closer to the language families. These modules in general perform transformations of particular subtrees in accordance with their type. The type of a subtree is determined by the syntactic and/or semantic role the root element is performing according to the parent element in the tree representation. In general, a subtree of a certain type corresponds to a particular clause or its part in a sentence of a natural language. Figure 5 shows a subtree that corresponds to a predicate noun in a natural language.

The procedures with language-specific rules can be found in the third layer. These procedures map the parts of a tree structure to the elements of a natural language in a way that is specific to the language of interpretation. An example of such a transformation is the replacement of the e-speranto concepts and their attributes with the words of the target language or the rearrangement of the tree edges in accordance with the word order in which particular clauses appear in the target language. The access to word and phrase dictionaries is also implemented in this layer.

7 An example of interpretation in English and Slovene

Although e-speranto is primarily designed for the Slavic languages, it is not limited to this language group. It can also be used for interpreting in other languages with somewhat limited accuracy. Let us have a look at an example: the interpretation of an e-speranto record of the sentence "*E-speranto is a design of a computer language*." (Figure 2) in English, which can be classified in the Germanic language family, and the interpretation of the same sentence in Slovene, a representative of the Slavic family. As already stated, the two language groups differ, among other things, in the part of speech agreement. While the Slavic languages use inflections for denoting cases, the latter are indicated with word order or prepositions and only rarely with suffixes in Germanic languages.

$\boldsymbol{E_i}$ *– a set of all nodes that are subordinate to the i-th node*
$\boldsymbol{NT_i}$ *– a set of non-terminal nodes that are subordinate to the i-th node*
$\boldsymbol{T_i}$ *– a set of terminal nodes that are subordinate to the i-th node*
$\boldsymbol{L_i}$ *– a set of terminal nodes that represent concepts and are subordinate to the i-th node*
$\boldsymbol{A_i}$ *– a set of terminal nodes that represent the attributes of concepts and are subordinate to the i-th node*

$\boldsymbol{L_i \cap A_i = T_i}$, $\boldsymbol{T_i \cap NT_i = E_i}$

```
1.) For the current node create a new root node $N_i$.
2.) For the current node find sets $L_i$ and $A_i$.
3.) For every $l \in L_i$ pass the execution to a module on a lower layer with set $A_i$ as the parameter.
4.) Add a subtree which is a result of the transformation under 3.) to the node $N_i$.
5.) For the current node find the set $NT_i$.
6.) For every root element $m \in NT_i$ find the type of the subordinate subtree with m as the root element and pass the execution to a procedure that performs the transformation of such a subtree with $A_i$ functioning as the parameter.
7.) Add the results returned under 6.) to the node $N_i$.
8.) As a result of the transformation return a subtree with the root $N_i$.
```

Table 1: Example of the algorithm for the subtree transformation with a change of the layer of execution.

Figure 5 shows a subtree in the form of a tree structure of the Mathematica programming language. The subtree corresponds to the noun phrase within the predicate of the analysed sentence. The mapping of the subtree to a structure which is closer to the language of interpretation is carried out in the middle layer (Figure 4) on the basis of the algorithm presented in Table 1. The transformation depends on the type of the subtree. The interpretation of the concepts within the tree structure, together with their attributes (e.g. number, deep case, etc), is performed by a module on the lowest level ("the layer of languages" in the Figure 4).

Figure 6 shows the result of the transformation performed by a module designed specifically for English. The deep case, marked with *ofGenitive* in the original representation tree, is expressed with the adding of a branch with the preposition *of* to a node where the source tree contained a branch with the attribute of the deep case.

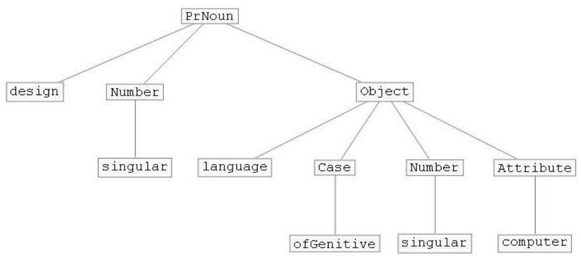

Figure 5: Noun phrase “*design of a computer language*” prior to the transformation in the form of the tree structure of the symbolic programming language Mathematica.

Interpretation of this structure in Slovene is performed in a completely different manner, since the deep cases in e-speranto are linked with the selection of suffixes that are added to the word roots (Figure 7). Selection of the suffix depends on the grammatical case that corresponds to the deep case in e-speranto (in Slovenian in this case to the genitive). Moreover, selection of the suffix also depends on the grammatical gender of the Slovene equivalent of the e-speranto concept language (in Slovenian *jezik*) and its number. The suffix *–a* is thus added to the root word *language* (*jezik*) and the suffix *–ega* is added to the root word *računalnišk* (*računalniški* = *computer*). The case must be assigned to all e-speranto attributes that are derived from the level where the deep case is specified. If, for instance, the attribute *formal* were also subordinate to the attribute *computer*, the interpretation of the deep case *ofGenitive* would be necessary also on this level (*zasnova formaln-ega računalnišk-ega jezik-a, design of a formal computer language*).

A module that supports the aforementioned mechanism of composing language units can also be used for other Slavic languages by changing only the content of the layer of languages.

In the presented example, interpretation in Slovene (and other Slavic languages) differs significantly from that in English. The difference lies both in the language rules and also in the procedures on the basis of which the rules are applied. Of course, we could create a generalized procedure that would perform the same tasks on the basis of different rules for both Slovene and English. The usability of such a method for languages that do not belong to the Indo-European language group (e.g. Japanese or Chinese) is questionable. The development of general algorithms for processing the language structures is a demanding task which requires cooperation of a large group of people from different fields. The process is much easier if the developmental group uses the already developed modules on higher levels of abstraction and merely adds the content that is specific to their language of interpretation.

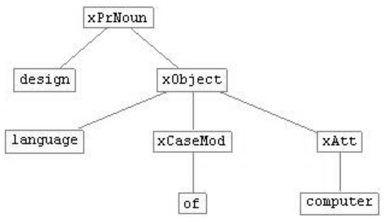

Figure 6: Tree after transformation with a module specific to English. The deep case in e-speranto is expressed with a preposition in English.

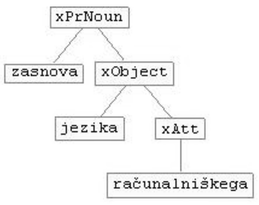

Figure 7: Tree after transformation with a module for the Slavic languages, specifically with the Slovenian grammar rules. The deep case in e-speranto is expressed with suffixes that are added to the root words.

8 Conclusion

In this paper we introduced the idea of a modular design of interpreters of the interlingua with a layered architecture on different layers of abstraction. The levels of abstraction refer to the degree of abstractness of the language structures that enter modules on a certain level and are processed by the procedures that are implemented in the modules. The potential advantage of the proposed approach is especially reduction of the development cost of the interpreters in related languages, since the development of modules that cover the common characteristics of languages is required only once.

The idea of the abstractly layered interpreter was practically implemented on the interpretation of e-speranto in the Slovenian and English language. These two interpreters comprise a part of the modules that is common and a part specific to an individual language. Although the languages belong to different language subgroups, we were able to reuse about half of the modules or the programming code.

The INES framework is currently in the process of being extended and upgraded. As at the time being, its functionality enables merely processing of simple language structures, it is not to be expected that with a further upgrading of its functionality the factor of reuse in interpretation in two languages that do not share many similarities will remain on this level. We expect better results with interpretation of languages that are similar both in syntax and grammar.

Our future work will be towards two fields. First, we want to perform a more thorough research in the optimal number of layers in the layered architecture of the interpreters and determine the content that needs to be placed into individual layers so that the factor of reuse is optimally high. Secondly, we intend to test the idea on a large number of Slavic languages and to assess its advantages in terms of the cost of the interpreter development and the quality of interpretation. The reference language will still be English. The issue of interfaces will be given some attention, too. An effort will be taken to assure conformity of data structures that are passed among the layers, particularly with more complex interpreters that combine a large number of modules.

9 References

[1] R. G. Gordon Jr., (ed.), *Ethnologue: Languages of the World*, 15th edition, Dallas, Tex: SIL International, 2005

[2] Presis, Amebis, http://presis.amebis.si

[3] Systran, http://www.systransoft.com

[4] Promt, http://www.promt.ru/ru/index.php

[5] Google Translate, http://translate.google.com

[6] W. Hutchins, H. Somers, *An Introduction to Machine Translation*, Academic Press, New York, 1992

[7] K. Schubert, The Architecture of DLT – interlingual or double-dialect, *New Directions in Machine Translation*, Floris Publications, Holland, 1988

[8] E. Nyberg, T. Mitamura, The KANT system: Fast, accurate, high-quality translation in practical domains, *COLING*, 1992

[9] H. Uchida, et al., Universal Networking Language: A gift for a millenium. The United Nations University, Tokyo, Japan, 1999

[10] I. Bugoslavsky, Some controversial Issues of UNL: Linguistic Aspects, *Universal Network Language: Advances in Theory and Applications*, Research on Computer Science, 2005

[11] C. Boitet, GETA's methodology and its current developments, *PACLING'97*, Meisei University, Ohme, Japan, Proc. 23-57, September 1997

[12] G. Sérasset, C. Boitet, On UNL as the future "html of the linguistic content" & reuse of existing NLP components in UNL-related applications with the example of a UNL-French deconverter, *COLING*, August 2000

[13] E. Blanc, From the UNL hypergraph to GETA's multilevel tree, *MT2000: machine translation and multilingual applications in the new millennium*, University of Exeter, British Computer Society, November 2000

[14] S. Omerović, G. Jakus, T. Filimonova, S. Tomažič, Zapis večjezičnih besedil v e-sperantu, *Elektrotehniški vestnik*, Vol. 74, No. 3, 2007

[15] S. Tomažič, Multilingual Web with E-speranto, *The IPSI BgD Transactions on Internet Research*, IPSI Bgd Internet Research Society, July 2007 Vol. 3 No. 2

[16] F. Amerio, G. Bonvecchiato, G. C. Fighiera, *Esperanto: Data and Facts*, 2nd edition, FEI - Milan, 2002

Grega Jakus graduated from the Faculty of Electrical Engineering of the University of Ljubljana, Slovenia, in 2007. He is currently employed as a junior researcher in the Laboratory of Communication Devices at the same faculty. His research focuses on machine translation algorithms.

Sanida Omerović graduated from the Faculty of Electrical Engineering of the University of Belgrade, Serbia, in 2005. Currently she is a postgraduate student at the Faculty of Electrical Engineering of the University of Ljubljana. Her research focuses on knowledge presentation systems.

Tatjana Filimonova received her Ph.D. in 2004 from the Philological Faculty, MGU, Russia. She works as a researcher in the Laboratory of Communication Devices at the Faculty of Electrical Engineering. Her research includes computer linguistics, lexicology and lexicography.

Sašo Tomažič is a Full Professor at the Faculty of Electrical Engineering of the University of Ljubljana. He is the Head of the Laboratory of Communication Devices and of the Chair of Telecommunications. His work includes research in the field of signal processing, security in telecommunications, electronic commerce and information systems.