Multilinguality in a Text Generation System For Three Slavic Languages

Geert-Jan Kruijff[a], **Elke Teich**[b], **John Bateman**[c], **Ivana Kruijff-Korbayová**[a], Hana Skoumalová[a], Serge Sharoff[d], Lena Sokolova[d], Tony Hartley[e], Kamenka Staykova[f], Jiří Hana[a]

[a]Charles University, Prague; [b]University of the Saarland; [c]University of Bremen; [d]RRIAI, Moscow; [e]University of Brighton; [f]IIT, BAS, Sofia

http://www.itri.brighton.ac.uk/projects/agile/

Abstract

This paper describes a multilingual text generation system in the domain of CAD/CAM software instructions for Bulgarian, Czech and Russian. Starting from a language-independent semantic representation, the system drafts natural, continuous text as typically found in software manuals. The core modules for strategic and tactical generation are implemented using the KPML platform for linguistic resource development and generation. Prominent characteristics of the approach implemented are a treatment of multilinguality that makes maximal use of the commonalities between languages while also accounting for their differences and a common representational strategy for both text planning and sentence generation.

1 Introduction

This paper describes the Agile system[1] for the multilingual generation of instructional texts as found in software user-manuals in Bulgarian, Czech and Russian. The current prototype focuses on the automatic drafting of CAD/CAM software documentation; routine passages as found in the AutoCAD user-manual have been taken as target texts. The application scenario of the Agile system is as follows. First, a user constructs, with the help of a GUI, language-independent task models that specify the contents of the documentation to be generated. The user additionally specifies the language (currently Bulgarian, Czech or Russian) and the register of the text to be generated. The Agile system then produces continuous instructional texts realizing the specified content and conforming to the style of software user-manuals. The texts produced are intended to serve as drafts for final revision; this 'drafting' scenario is therefore analogous to that first explored within the Drafter project. Within the Agile project, however, we have explored a more thoroughly multilingual architecture, making substantial use of existing linguistic resources and components.

The design of the Agile system overall rests on the following three assumptions.

First, the input of the system should be specified irrespective of any particular output language. This means that the user must be able to express the content that she wants the texts to convey, irrespective of what natural language(s) she masters and in what language(s) the output text should be realized. Such language-independent content specification can take the form of some knowledge representation pertaining to the application domain.

Second, the texts generated as the output of the system should be well-formulated with respect to the expectations of native speakers of each particular language covered by the system. Since differences among languages may appear at any level, language-sensitive decisions about the realization of the specified content must be possible throughout the generation process.

And third, the notion of multilinguality employed in the system should be recursive, in the sense that the modules responsible for the generation should themselves be multilingual. The text generation tasks which are common to the languages under consideration should be performed only once. Ideally, there should be one process of generation yielding output in multiple languages rather than a sequence of monolingual processes. This view of 'intrinsic multilinguality' builds on the approach set out in Bateman et al. (1999). Each module of the system is fully multilingual in that it simul-

[1]EU Inco-Copernicus project PL961004: 'Automatic Generation of Instructional Texts in the Languages of Eastern Europe'

taneously enables both *integration* of linguistic resources, defining commonalities between languages, and resource *integrity*, in that the individuality of each of the language-specific resources of a multilingual ensemble is always preserved.

We consider these assumptions and the view of multilinguality entailed by them to be crucial for the design of effective multilingual text generation systems. The results so far achieved by the Agile system support this and also offer a solid experiential basis for the development of further multilingual generation systems.

The overall operation of the Agile system is as follows. After the user has specified some intended text content (described in Section 2) via the Agile GUI, the system proceeds to generate the texts required. To do this, a text planner (Section 3) first assigns parts of the task model to text elements and arranges them in a hierarchical fashion–a text plan. Then, a sentence planner organizes the content of the text elements into sentence-sized chunks and creates the corresponding input for the tactical generator, expressed in standard sentence planning language (SPL) formulae. Finally, the tactical generator generates the linguistic realizations corresponding to these SPLs–the text (Section 4). In the stage of the project reported here, we concentrated particularly on procedural texts. These offer step-by-step descriptions of how to perform domain tasks using the given software tools. A simplified version of one such procedural text is given (for English) in Figure 1. This architecture mirrors the reference architecture for generation discussed in Reiter & Dale (1997). The modules of the system are pipelined so that a continuous text is generated realizing the intended meaning of the input semantic representation without backtracking or revision.

Several important properties have characterized the method of development leading to the Agile system. These are to a large extent responsible for the effectiveness of the system. These include:

Re-use and adaptation of available resources. We have re-used substantial bodies of existing linguistic resources at all levels relevant for the system; this played a crucial role in achieving the sophisticated generation capabilities now displayed by the system in each of its languages of expertise—prior to the project there were no substantial automatic generation systems for any of the languages covered. The core modules for strategic and tactical generation were all implemented using the Komet-Penman Multilingual system (KPML: cf. Bateman et al., 1999)—a Common Lisp based grammar development environment. In addition, we adopted the Penman Upper Model as used within Penman/KPML as the basis for our linguistic semantics; a more restricted domain model (DM) relevant to the CAD/CAM-domain was defined as a specialization of the UM concepts. The DM was inspired by the domain model of the Drafter project, but presents a generalization of the latter in that it allows for embedding tasks and instructions to any arbitrary recursive depth (i.e., more complex text plans). Already existing lexical resources and morphological modules available to the project were re-used for Bulgarian, Czech and Russian: the Czech and Bulgarian components were modules written in C (e.g., Hajič & Hladká, 1997, for Czech) that were interfaced with KPML using a standard set of API-methods (cf. Bateman & Sharoff, 1998). Finally, because no grammars suitable for generation in Bulgarian, Czech and Russian existed, a grammar for English (NIGEL: Mann & Matthiessen, 1985) was re-used to build them; for the theoretical basis of this technique see Teich (1995).

To draw a polyline
First start the PLINE command using one of these methods:
Windows From the Polyline flyout on the Draw toolbar, choose Polyline.
DOS and UNIX From the Draw menu, choose Polyline.

1. Specify the start point of the polyline.
2. Specify the next point of the polyline.
3. Press Return to end the polyline.

Figure 1: Example "To draw a polyline"

Combination of two methods of resources development. Two methods were combined to enable us to develop basic general-language grammars and sublanguage grammars for CAD/CAM instructional texts at the same time. One method is the system-oriented one aimed at building a computational resource with a view of the whole language system: this is a method strongly supported by the KPML development environment. The other method is instance-oriented, and is guided by a detailed register analysis. The latter method was particularly important given the Agile goal of being able to generate texts belonging to rather diverse text types— e.g., impersonal *vs.* personal; procedural, functional descriptions, overviews etc.

Cross-linguistic resource-sharing. A cross-linguistic approach to linguistic specifications and implementation was taken by maximizing resource sharing, i.e. taking into account similarities and differences among the treated languages so that development tasks have been distributed across different languages and re-used wherever possible.

2 Language-independent Content Specifications

The content constructed by a user via the Agile GUI is specified in terms of *Assertion-boxes* or A-boxes. These A-boxes are considered to be entirely neutral with respect to the language that will be used to express the A-box's content. Thus individual A-boxes can be used for generating multiple languages. A-boxes specify content by *instantiating concepts* from the DM or UM, and placing these concepts in relation to one another by means of *configurational concepts*. The configurational concepts define admissible ways in which content can be structured. Figure 2 gives the configurational concepts distinguished within Agile.

Procedure A procedure has three slots:
(i) GOAL (obligatory,filled by a USER-ACTION),
(ii) METHODS (optional, filled by a METHOD-LIST),
(iii) SIDE-EFFECT (optional, filled by a USER-EVENT).

Method A method has three slots:
(i) CONSTRAINT (optional, filled by an OPERATING-SYSTEM),
(ii) PRECONDITION (optional, filled by a PROCEDURE),
(iii) SUBSTEPS (obligatory, filled by a PROCEDURE-LIST).

Method-List A METHOD-LIST is a list of METHOD's.

Procedure-List A PROCEDURE-LIST is a list of PROCEDURE's.

Figure 2: Configurational concepts

Configurational concepts are devoid of actual content. The content is provided by instantiations of concepts that represent various user actions, interface events, and interface modalities and functions. Taken together, these instantiations provide the basic propositional content for instructional texts and are taken as input for the text planning process.

3 Strategic Generation: From Content Specifications to Sentence Plans

To realize an A-box as a text, we go through successive stages of text planning, sentence planning, and lexico-grammatical generation (cf. also Reiter & Dale, 1997). At each stage there is an increase in sensitivity to, or dependency on, the target language in which output will be generated. Although the text planner itself is language-independent, the text planning resources may differ from language to language as much as is required. This is exactly analogous to the situation we find within the individual language grammars as represented within KPML: we therefore represent the text planning resources in the same fashion. For the text type and languages of concern here, however, variation across languages at the text planning stage turned out to be minimal.

The organization of an A-box is used to guide the text planning process. Here, we draw a distinction between *text structure elements* (TSEs) as the elements from which a (task-oriented) text is built up, and *text templates*, which condition the way TSEs are to be realized linguistically. We focus on the relation between concepts on the one hand, and TSEs on the other. We are specifically interested in the configurational concepts that are used to configure the content specified in an A-box because we want to maintain a close connection between how the content can be defined in an A-box and how that content is to be spelled out in text.

3.1 Structuring and Styling

A text structure element is a predefined component that needs to be filled by one or more specific parts of the user's content definition. Using the reader-oriented terminology common in technical authoring guides, we distinguish a small (recursively defined) set of text TSEs; these are listed in Figure 3.

Task-Document A TASK-DOCUMENT has two slots:
(i) TASK-TITLE (obligatory),
(ii) TASK-INSTRUCTIONS (obligatory), being a list of at least one INSTRUCTION.

Instruction An INSTRUCTION has three slots:
(i) TASKS (obligatory), being a list of at least one TASK,
(ii) CONSTRAINT (optional),
(iii) PRECONDITION (optional).

Task A TASK has two slots:
(i) INSTRUCTIONS (optional),
(ii) SIDE-EFFECT (optional).

Figure 3: Text Structure Elements (TSEs)

The TSEs are placed in correspondence with the configurational concepts of the DM (cf. Figure 2); this enables us to build a text structure that follows the structuring of the content in an A-box (cf. Figure 4).

Orthogonal to the notion of text structure element is the notion of text template. Whereas TSEs capture *what* needs to be realized, the text template captures *how* that content is to be realized. Thus, a template defines a style for expressing the content. As we discuss below, we define text templates in terms of constraints on the realization of specific (individual) TSEs. For example, whereas in Bulgarian and Czech headings (to which the TASK-TITLE element corresponds: cf. Figure 4) are usually realized as nominal groups, in the Russian AutoCAD manual headings are realized as nonfinite purpose clauses as they are in English.

3.2 Text Planning & Sentence Planning

The major component of the text planner is formed by a systemic network for text structuring; this network, called the text structuring region, defines an additional level of linguistic resources for the level of *genre*. This region constructs text structures in a way that is very similar to the way in which the systemic networks of the grammars of the tactical generator build up grammatical structures. In fact, by using KPML to implement this means for text structuring, the interaction between global level text generation (strategic generation) and lexico-grammatical expression (tactical generation) is greatly facilitated. Moreover, this approach has the advantage that constraints on output realization can be easily accumulated and propagated: for example, the text planner can impose constraints on the output lexicogrammatical realization of particular text plan elements, such as the realization of text headings by a nominalization in Czech and Bulgarian or by an infinite purpose clause in Russian. This is one contribution to overcoming the notorious *generation gap* problem caused when a text planning module lacks control over the fine-grained distinctions that are available in a grammar. In our case, both text planning and sentence planning are integrated into one and the same system and are distinguished by stratification.

TASK-TITLE	↔	GOAL of topmost PROCEDURE
TASK-INSTRUCTIONS	↔	METHODS of PROCEDURE
SIDE-EFFECT	↔	SIDE-EFFECT of PROCEDURE
TASK	↔	GOAL of PROCEDURE
CONSTRAINT	↔	CONSTRAINT of METHOD
PRECONDITION	↔	PRECONDITION of METHOD
INSTRUCTION-TASKS	↔	SUBSTEPS of a METHOD
INSTRUCTION	↔	METHOD

Figure 4: Mapping TSEs and configurational concepts defined in the DM

Following on from the orthogonality of text templates and text structure elements, the text structuring region consists of two parts. One part deals with interpreting the A-box in terms of TSEs: traversing the network of this part of the region produces a text structure for the A-box conforming to the definitions above. The second part of the region imposes constraints on the realization of the TSEs introduced by the first part. Diverse constraints can be imposed depending on the user's choice of style, e.g., personal (featuring ppredominantly imperatives) *vs.* impersonal (featuring indicatives).

The result of text planning is a text plan. This can be thought of as a hierarchical structure (built by TSEs) with bits of A-box content at its leaves together with additional constraints imposed by the text planning process: e.g., that the Title segment of the document should not be realized as a full clause but rather as a nominal phrase or a purposive dependent clause. The text plan may also include constraints on preferred layout of the document elements: this information is passed on via HTML annotations. The sentence planner then takes this text plan as input, and creates SPL formulae to express the content identified by the text plan's leaves. The resulting SPLs can also group one or more leaves together (aggregation) depending on decisions taken by the text planner concerning discourse relations. Furthermore, constraints on realization that were introduced by the text-planner are also included into the SPLs at this stage.

Of particular interest multilingually is the way concepts may require different kinds of realizations in different languages. For example, languages need not of course realize concepts as single words: in Czech the concept *Menu* gets realized as "menu" but the interface modality *Dialogbox* is realized as a multiword expression "dialogové okno" (whose components—i.e., an adjective and a nominal head—may undergo various grammatical operations independently). The Agile system sentence planner handles such cases by inserting SPL forms corresponding to the literal semantics of the complex expressions required; these are then expressed via the tactical generator in the usual way. The resulting SPL formulas thus represent the language-specific semantics of the sentences to be generated. Otherwise, if a concept maps to a single word, the sentence planner leaves the further specification of how the concept should be realized to the lexico-grammar and its concept-to-word mappings. More extensive differences between languages are handled by conditionalizing the text and sentence planner resources further according to language.

4 Tactical Generation: From Sentence Plans to Sentences

The tactical generation component that constructs sentences (and other grammatical units) from the SPL formulae specified in the text plan relies on linguistic resources for Bulgarian, Czech and Russian. The necessary grammars and lexicons have been constructed employing the methods described in Section 1. As noted there, the crucial characteristic of this model of multilingual representation is that it allows for the representation of *both* commonalities *and* differences between languages, as required to cover the observable contrastive-linguistic phenomena. This can be applied even among typologically rather distant languages.

We first illustrate this with respect to some of the contrastive-linguistic phenomena that are covered by this model employing examples from English, Bulgarian, Czech and Russian. We then show the organization of the lexicons and briefly describe lexical choice.

4.1 Semantic and grammatical cross-linguistic variation

One of the tenets of our model of cross-linguistic variation is that languages have a rather high degree of similarity semantically and tend to differ syntactically. We can thus expect to have identical SPL expressions for Bulgarian, Czech and Russian in many cases, although these may be realized by diverging syntactic structures. However, we also allow for the case in which there is no commonality at this level and even the SPL expressions diverge.[2] Example 1 illustrates the latter case (high semantic divergence, plus grammatical divergence), and example 2 the former (semantic commonality, plus grammatical divergence).

Example 1: English and Russian spatial PPs. The major lexico-grammatical difference between English and Russian prepositional phrases is that the relation expressed by the PP is realized by the choice of the preposition in English, whereas in Russian, it is in addition realized by case-government. In the area of spatial PPs, the choice of a particular preposition in English corresponds to a distinction in the dimensionality of the object that realizes the range of the relation expressed by the PP. For both PPs expressing a location and PPs expressing movement, English distinguishes between three-dimensional objects (*in, into*), one-or-two-dimensional objects (*on, onto*) and zero-dimensional objects (*at, to*).

In Russian, in contrast, zero-or-three dimensional objects (preposition: *v*) are opposed to one-or-two-dimensional objects (preposition: *na*). A further difference between the expression of static location vs. movement is expressed by case selection: *na*/*v*+locative case expresses static location, *v*/*na*+accusative case expresses movement (entering or reaching an object) and the preposition *k*+dative case expresses movement towards an object (not quite reaching or

[2]This distinguishes our approach from interlingua-based systems, which typically require a common semantic (or conceptual) input.

entering it). In the converse relation, motion away from an object, *s* is selected for movement from within an object, and *ot* for movement away from the vicinity of an object. Here, both prepositions govern genitive case. The dimensionality of the object is only relevant for the distinction between *v/na* and *s/ot*, but not for *k*. Since the conceptualizations of spatial relations are different across English and Russian, the input SPL expressions diverge as shown in Figure 5); rather than using domain model concepts, these SPL expressions restrict themselves to Upper Model concepts in order to highlight the cross-linguistic contrast. This example illustrates well how it is often necessary to 'semanticize' events differently in different languages in order to achieve the most natural results. Note that Czech is here very similar to Russian.

a. SPL Russian

```
(example
 :name DO-Text1-Ru
 :targetform "Pomestite fragment v bufer."
 :logicalform
 (s / dispositive-material-action
      :lex pomestitj
      :speech-act-id command
      :actee (a / object :lex fragment)
      :destination (d / THREE-D-OBJECT
                         :lex bufer)))
```

b. SPL for English

```
(example
 :name DO-Text1-En
 :targetform "Put the selection on the clipboard."
 :logicalform
 (s / dispositive-material-action
      :lex put
      :speech-act-id command
      :actee (a / object :lex selection)
      :destination (d / ONE-OR-TWO-D-OBJECT
                         :lex clipboard)))
```

Figure 5: SPL expressions

Example 2: English, Bulgarian and Czech headers in CAD/CAM texts. Grammatical units (1)–(4) below show an example of cross-linguistic commonality at the level of semantic input and divergence at the level of grammar. These units all function as self-sufficient Task-titles for the descriptions of particular actions that can be performed with the given software.

(1) En: To draw a polyline

(2)	Bu:	Чертаене на	полилиния
		Drawing- of	polyline-INDEF
		NOMINAL	
(3)	Cz:	Kreslení	křivky
		drawing-NOMINAL	line-GEN
(4)	Ru:	Чтобы нарисовать	полилинию
		in-order draw-INF	polyline-ACC

There are two major differences across (1)–(4) that need to be accounted for: (i) they exhibit **divergent grammatical ranks** in that (1) and (4) are clauses (nonfinite), while (2) and (3) are nominal groups (nominalizations); and (ii) they show **divergent syntactic realizations**: (2) and (3) differ in that in Bulgarian, which does not have case, the relation between the syntactic head Чертаене (*chertaene*) and the modifier полилиния (*polilinia*) is expressed by a preposition на (*na*), whereas in Czech, which has case, this relation is expressed by genitive case (*křivky*).

Despite these differences, only the first divergence has any consequences for the SPL expressions required; the basic semantic commonality among (1)–(4) is preserved. This is shown in Figure 6 by means of the standard linguistic *conditionalization* provided by KPML for all levels of linguistic description. The conditionalization shows that both the English (1) and the Russian (4) are nonfinite clauses while the Bulgarian (2) and the Czech (3) are nominalizations. These SPL expressions also show the use of domain concepts as produced by the text planner rather than upper model concepts as in the SPLs in Figure 5.

```
(example
 :name DO-Text1
 :logicalform
 (s / DM::draw
      :en :ru   :PROPOSAL-Q & PROPOSAL
      :bu :cz   :EXIST-SPEECHACT-Q & NOSPEECHACT
      :actee (d / DM::polyline)))
```

Figure 6: Multilingual SPL expression for the header examples

The second difference is handled by the generation grammars internally. Here, Bulgarian and Czech share the basic functional-grammatical description of postmodifiers for nominalizations (Figure 7). The difference in structure only shows in syntagmatic realization and is separate from the functional description: For Bulgarian, the postmodifier marker на (*na*: 'of') is inserted, and for Czech, the nominal group realizing the Postmodifier is attributed genitive case.[3]

```
(gate
 :name MEDIUM-QUALIFIER
 :inputs processual-mediated
 :outputs
  ((1.0 medium-qualifier
   (:bu :cz  preselect Medium nominal-group)
   (:cz  preselect Medium noun-gen-case)
   (:bu insert Mediumqualifiermarker)
   (:bu lexify Mediumqualifiermarker na)))
 :region QUALIFICATION)
```

Figure 7: Shared system for Bulgarian and Czech

4.2 Lexical choice and lexicons

The lexical items for each language are selected from the lexicon via the domain model. A DM concept is annotated with one or more lexical items from each language. If there is more than one item per language, the choice is constrained by features imposed by the grammar.

For example the concept `DM::draw` is annotated with two lexical items which are the imperfective and perfective forms of the verb *draw* in Czech, Bulgarian and Russian. If the grammar selects imperfective aspect, the first is chosen; if the grammar selects perfective aspect, the second is chosen. This mechanism is used also for the choice between a verb and its nominalization, among others. With the help of the lexicon, the inflectional properties collected for a particular lexical item during generation are translated into a format suitable for external morphological modules, which are then called. The result of the external module, the inflected form, is passed back to the KPML system and inserted into the grammatical structure.

5 Evaluation and Conclusions

A first round of evaluation has been carried out on the Agile prototype. This directly assessed the ability of users to control multilingual generation in the three languages, as well as the design and robustness of the system components. Groups of users were given a brief training period and then asked to construct A-boxes expressing given content. Texts were cross-generated: i.e., the languages were varied across the A-boxes independently of the native languages of the subjects who created them. Errors were then classified and recommendations for the next and final Agile prototype collected. The generated texts were then evaluated by expert technical authors. They were generally judged to be of a broadly similar quality to the texts originating from manuals, and both kinds of texts received similar criticism. The main source of criticism and errors was the design of the GUI which is now being improved for the final prototype. The overall design of the system has therefore shown itself to offer an effective approach for multilingual generation. We are now extending the system to cover a broader range of text types as well as the further grammatical and semantic variation required by the evaluators as well as by the additional text types.

[3]This description is also valid for Russian, which has a nominal group structure similar to Czech. The Bulgarian one is more like English.

References

Bateman, J. A., Matthiessen, C. M. I. M., & Zeng, L. (1999). Multilingual natural language generation for multilingual software: a functional linguistic approach. *Applied Artificial Intelligence, 13*(6), 607–639.

Bateman, J. A. & Sharoff, S. (1998). Multilingual grammars and multilingual lexicons for multilingual text generation. In *Multilinguality in the lexicon II*, ECAI'98 Workshop 13, (pp. 1–8).

Hajič, J. & Hladká, B. (1997). Probabilistic and rule-based tagger of an inflective language – a comparison. In *Proceedings of ANLP'97*, (pp. 111–118).

Mann, W. C. & Matthiessen, C. M. I. M. (1985). Demonstration of the Nigel text generation computer program. In J. D. Benson & W. S. Greaves (Eds.), *Systemic Perspectives on Discourse, Volume 1* (pp. 50–83). Ablex.

Reiter, E. & Dale, R. (1997). Building applied natural language generation systems. *Journal of Natural Language Engineering, 3*, 57–87.

Teich, E. (1995). Towards a methodology for the construction of multilingual resources for multilingual text generation. In Proceedings of the IJCAI'95 workshop on multilingual generation, (pp. 136–148).