# Multilinguality in a Text Generation System For Three Slavic Languages


## Abstract

The paper describes a multilingual textgeneration system for Bulgarian, Czech and Russian in the domain of CAD/CAM software instructions. Starting from a language-independent semantic representation, the system drafs continuous text as it is typically found in instructional texts in the three languages. The core modules for strategic and tactical generation are implemented using the KPML platform for resource development and generation. The prominent characteristic of the approach to multilinguality implemented here is that it makes maximal use of the commonalities between languages while also accounting for the differences between them.


## 1 Introduction

This paper describes how multilingual generation of Bulgarian, Czech and Russian instructional texts is achieved in the X project. Starting from language-independent semantic representations specified by a user, the system drafts continuous texts such as found in software user-manuals. Individual modules of the system are multilingual in that they enable both *integration* of resources defining commonalities between languages, and *integrity* in that at the same time the ensemble of language-specific resources is preserved (cf. (Bateman *et al.*1999)).

We consider the following assumptions as most essential for a multilingual text generation system.

Firstly, 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 a knowledge representation pertaining to the application domain.

Secondly, 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, some language-sensitive decisions about the realization of the specified content have to be made throughout the generation process.

Thirdly, 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.

These assumptions characterize the notion of multilinguality we have been trying to employ throughout the development of our system. A prototype system for automatic drafting of CAD/CAM software documentation has been built that implements this notion of multilinguility. The application scenario of the system is as follows.

With the help of a GUI, the user provides language-independent task models. These task models specify the contents of the texts that are to be generated.

Subsequently, the continuous instructional texts realizing the specified content in Bulgarian, Czech and Russian are generated as follows. First, the text planner assigns parts of the task model to text elements and arranges them in a hierarchical fashion - a text plan. Then, the sentence planner organizes the content of the text

*To draw 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"

elements into sentence-sized chunks and creates the corresponding sentence planning language (SPL) formulae. These SPL formulae then serve as input to the tactical generator that generate the linguistic realizations corresponding to these SPLs - the text. This architecture mirrors the reference architecture for generation as discussed in (Reiter & Dale1997).

The texts we are aiming to generate are instructions of the kind one can frequently find in software user-manuals. In the current stage of the project, we concentrate on procedural texts. These are step-by-step descriptions how to perform domain tasks using the given software tools. We chose as target texts routine passages from the AutoCAD user-manual. A simplified version of one such procedural text is given in Figure 1.

As explained above, the modules are pipelined so that a continuous text is generated realizing the intended meaning of the input semantic representation. The core modules for strategic and tactical generation are implemented using the Komet-Penman Multilingual system (KPML,(Bateman1997)). The development method we have employed has the following main characteristics:

**Re-use and adaptation of available resources.** The Penman Upper Model of KPML (UM, (Bateman *et al.*1991)) was adopted as the basis for our linguistic semantics. The more restricted domain model (DM) is defined as a specialization of UM concepts, relevant to the CAD/CAM-domain. The DM was inspired by the DM 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). The lexical resources and morphological modules for Bulgarian, Czech and Russian available to us were re-used as modules interfaced with KPML (e.g. (Hajič and Hladká1997) for Czech). Because no grammars suitable for generation in Bulgarian, Czech and Russian existed, a grammar for English (NIGEL, (Mann and Matthiessen1985)) was re-used to build them.

**Combination of two methods of resources development**, enabling 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, on which KPML is based. The other method is instance-oriented, being guided by a register analysis.

**Cross-linguistic resource-sharing.** A cross-linguistic approach to linguistic specifications and implementation is taken by maximizing resource sharing, i.e. taking into account similarities and differences among the treated languages

The paper is structured as follows. In Section 2 we describe the semantic representations used as input to the generation process. In Section 3 we explain our approach to strategic generation, and we describe how the requirements of particular languages are taken into account in this process. In particular, we distinguish between text planning and sentence planning. In Section 4 we present the tactical generators. We make reference to English as a lingua franca since the grammars for Bulgarian, Czech and Russian have been built up on the basis of the English NIGEL grammar. For each language, there is a grammar, lexicon and a morphology module. In Section 5 we conclude with a summary.

## 2 Language-independent Content Specifications

Content is specified in terms of *Assertion-boxes* or A-boxes for short. These A-boxes are considered to be entirely neutral as for the language that will be used to express the A-box's content.

An A-box specifies content by *instantiating concepts* from the DM or UM, and putting these

**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

concepts into 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 we distinguish. These concepts cover the constructions found in the instructional texts we are dealing with.

Configurational concepts are devoid of actual content. The content is provided by instantiations of (DM) 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 can then be taken as input for a text planning process.

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 & Dale1997)). At each stage there is an increase in sensitivity to, or dependency on, the target language in which the text 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 in the individual language grammars, and the text planning resources are represented in the same fashion. For the text type and languages of concern here, however, variation across languages at the text planning stage is minimal. The result of text planning is a text plan, with fragments of propositional content distributed over the 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.

Of particular interest here is the way concepts get eventually realized. Naturally, languages need not realize concepts as single words. For example, in Czech the concept *Menu* gets realized as "menu" but the interface modality *Dialogbox* is realized as a multiword expression "dialogový okno" (whose components may undergo declension). We tackle this problem as follows. The sentence planner creates SPL formulas that specify sentences, realizing a part of the content of the A-box, as identified in the text plan. For concepts that need to be realized as multiword expressions, the sentence planner inserts the SPL code specifying that complex expression into the SPL specification of the whole clause. 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 (cf. §4.3 below). More extensive differences between languages are handled by conditionalizing the text planner resources further.

## 3 Strategic Generation: From Content Specifications to Sentence Plans

In this section we discuss how we can set up a structure for a text that is to realize a given A-box, by looking at how the concepts in the A-box are organized in order to specify a particular content. We begin by making a distinction between *text structure elements* being the elements from which a (task-oriented) text is built up, and *text templates*, which condition the way text structure elements are to be linguistically realized. We focus on the relation between concepts on the one hand, and text structure elements on the other hand. We are specifically interested in the configurational concepts that are used to configure the content specified in an A-box, because we want to have a close connection between how the content can be defined in an A-box and how that content is to be spelled

**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 (obligatory),
(ii) Side-Effect (optional).

Figure 3: Text Structure Elements

| | | |
|---|---|---|
| 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

out in a text using a particular organization.

### 3.1 Structuring and Styling

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

Our aim is to create a structure for a text that is to realize the content specified by an A-box. In Figure 4 the text structure elements (TSEs, cf. Figure 3) are related to the DM's configurational concepts (cf. Figure 2) so as to enable us to build a text structure following how content has been structured in an A-box.

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

### 3.2 Text Planning & Sentence Planning

The major component of the text planner is formed by the systemic network for text structuring. We have constructed a region that defines an additional level of linguistic resources for the level of *genre*. The region enables the composition of text structures in a way that is very similar to the way the lexico-grammar builds up grammatical structures. In fact, by using KPML to implement the means for text structuring,the interaction between global level text generation (strategic generation) and lexico-grammatical expression (tactical generation) is facilitated[1].

The region implements the viewpoint that text templates and text structure elements are essentially orthogonal ideas. Therefore it consists of two parts. One part deals with interpreting the A-box in terms of text structure elements. By traversing the network that the systems of this part make up, we obtain a text structure for the A-box conforming to the definitions we provided above. The other part of the region imposes constraints on the realisation of the text structure elements that are being introduced by traversing the other part of the region. Naturally it will depend on our choice of a particular text template (style) which constraints will be imposed. These choices are made through interaction between the user and the system.

The text plan we derive can be thought of as a hierarchical structure (built by TSEs) with bits of A-box content at its leaves. The sentence planner takes the text plan as input, and creates SPL formulas to realize the content identified by the text plan's leaves. The SPLs that the sentence planner outputs can group one or more leaves together (aggregation), depending on decisions taken by the text planner concerning dis-

[1] Thus, we overcome the notorious problem known as the *generation gap* in which a text planning module lacks control over the fine-grained distinctions that are available in the grammar. In our case, both text planning and sentence planning are integrated into one and the same system (organized in a stratificational manner).

course relations. Furthermore, constraints on realization that were introduced by the text-planner now get included into the SPLs.

From a multilingual perspective, of interest is the way we deal with language-independent concepts that get realized (in a specific language) by means of a complex expression. The solution implemented here is to associate bits of SPL code to each concept, relativized to a particular language, and include the proper bit of SPL code into an SPL whenever we need to realize a concept. The resulting SPL formulas thus represent the language-specific semantics of the sentences to be generated.

## 4 Tactical generation: From Sentence Plans to Sentences

The resources used at this stage in generation, i.e., the Bulgarian, Cezch and Russian grammars and lexicons, have been built up by a method called *resource sharing* (Bateman *et al.*1991), which makes maximal use of the commonalities between languages according to a small set of parameters which are given by the underlying linguistic model, Systemic Functional Linguistics (SFL; (Halliday1985)).

The crucial characteristic of this model of multilingual representation is that it allows for the representation of *both* commonalities *and* differences between languages, covering the observable contrastive-linguistic phenomena—even among typologically rather distant languages. For the theoretical basis of this model see (Teich1995); more details about the application in multilingual generation can be found in (Bateman *et al.*1999).

In the remainder of this section we exemplify the concept of cross-linguistic resource sharing and re-use implemented in the present approach: Section 4.1 illustrates some of the contrastive-linguistic phenomena that are covered by this model with examples from English, Bulgarian, Czech and Russian. Section 4.2 shows the organization of the lexicons and briefly describes lexical choice.

### 4.1 Semantic and grammatical cross-linguistic variation

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

**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 underlies 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*). The 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 on object (not quite reaching or 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 can thus diverge (see an example in Figure 5).[3]

**Example 2: English, Bulgarian and**

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

[3] Note that here, to illustrate the cross-linguistic contrast, the SPLs are written with Upper Model concepts rather than domain model concepts (as in Figure 6).

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 (1)

a. Shared partial SPL for English, Bulgarian, Czech and Russian

```
(example
 :name DO-Text1
 :logicalform
 (s / DM::draw
      :actee (d / DM::polyline)))
```

b. Complete SPL for English and Russian

```
(example
 :name DO-Text1-En-Ru
 :logicalform
 (s / DM::draw
      :PROPOSAL-Q PROPOSAL
      :actee (d / DM::polyline)))
```

c. Complete SPL for Bulgarian and Czech

```
(example
 :name DO-Text1-Bu-Cz
 :logicalform
 (s / DM::draw
      :EXIST-SPEECHACT-Q NOSPEECHACT
      :actee (d / DM::polyline)))
```

Figure 6: SPL expressions (2)

**Czech headers in CAD/CAM texts**. For an example of cross-linguistic commonality at the level of semantic input and divergence at the level of grammar, consider sentences (1)–(4) below, which function as headers of the descriptions of particular actions that can be performed with the given software.

(1) En: To draw a polyline

(2) Bu: Чертаене на полилиниа
Drawing-NOMINAL of polyline-INDEF

(3) Cz: Nakreslení křivky
drawing-NOMINAL line-GEN

(4) Ru: Чтобы нарисовать полилинию
in-order draw-INF polyline-ACC

The major differences between (1), (2), (3) and (4) that have to be accounted for are the following:

**Different ranks**: (1) and (2) are clauses (nonfinite), (3) and (4) are nominal groups (nominalizations).

**Different syntactic realizations**: (2) and (3) differ in that in Bulgarian, which does not have case, the relation between the syntactic head (*chertaene*) and the modifier (*polilinia*) is expressed by a preposition (*na*), whereas in Czech, which has case, this relation is expressed by genitive case *(křivky)*.

The only difference showing in the input SPL expressions is the first. Due to the basic semantic commonality among (1)-(4), the input SPL expressions are largely the same (see a. in Figure 6). However, the English (1) and the Russian (2) are nonfinite clauses (see the complete SPL in b. in Figure 6), the Bulgarian (2) and the Czech (3) are nominalizations (see the complete SPL in c. in Figure 6).

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 *na* ('of') is inserted, and for Czech, the nominal group realizing the postmodifier is attributed genitive case.[4]

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

```
(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 languages are picked up from the lexicon via the domain model. A domain model 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 domain 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 other purposes. With the help of the lexicon, the inflectional properties collected for a particular lexical item are translated into a format suitable for external morphological modules and this module is called. The result of the external module, the inflected form, is passed back to the KPML system.

## 5 Conclusions

The paper discussed the architecture of a system for generating instructional texts in Bulgarian, Czech and Russian. It was pointed out how the architecture, consisting of strategic generation (text and sentence planning) and tactical generation (lexicogrammatical generation) is a *multilingual* architecture by facilitating both integration of resources common to languages and preserving integrity of language-specific resources.

Strategic generation and tactical generation have been implemented in KPML, a platform for the development and management of multilingual resources. The implementation of the general architecture is based on standard generation techniques (c.f. e.g. (Reiter & Dale1997)). The major achievement over such architectures is that in the present approach, problems to do with the generation gap are avoided because text planning and sentence planning are integrated in one and the same system.

## References

John A. Bateman, Christian M.I.M. Matthiessen, Keizo Nanri, and Licheng Zeng. The re-use of linguistic resources across languages in multilingual generation components. In *Proceedings of IJCAI'91, Sydney, Australia*, volume 2, pages 966 – 971. Morgan Kaufmann Publishers, 1991.

John A. Bateman, Enabling technology for multilingual natural language generation: the KPML development environment. *Journal of Natural Language Engineering*, 3 (1):15–55. 1997

John A. Bateman, Christian M.I.M. Matthiessen, and Licheng Zeng. Multilingual natural language generation for multilingual software: a functional linguistic approach. *Applied Artificial Intelligence*, 13(6): 607-639, 1999

Jan Hajič and Barbora Hladká. Probabilistic and Rule-Based Tagger of an Inflective Language - a Comparison. In *Proceedings of ANLP'97*, pp. 111-118, Washington, USA, 1997.

Michael A.K. Halliday. *An Introduction to Functional Grammar*. Edward Arnold, London, 1985.

Robert T. Kasper. A flexible interface for linking applications to PENMAN's sentence generator. In *Proceedings of the DARPA Workshop on Speech and Natural Language*. 1989

William Mann and Christian M.I.M. Matthiessen. A demonstration of the Nigel text generation computer program. In James D. Benson and William S. Greaves, editors, *Systemic Perspectives on Discourse, vol 1*. Ablex, Norwood, N.J., 1985.

Ehud Reiter and Robert Dale. Building Applied Natural-Language Generation Systems. *Journal of Natural Language Engineering*, 3 (1):57-87, 1997.

Elke Teich. Towards a methodology for the construction of multilingual resources for multilingual text generation. In *Proceedings of the IJCAI'95 workshop on multilingual generation*, pages 136–148, 1995. Montréal, Canada.