AGILE – A System for Multilingual Generation of Technical Instructions

Anthony Hartley[1], Donia Scott[1], John Bateman[2], Danail Dochev[3]

[1]Information Technology Research Institute – University of Brighton
[donia.scott, tony.hartley]@itri.brighton.ac.uk
[2]Faculty of linguistics and literature (FB10)– University of Bremen
bateman@uni-bremen.de
[3]Institute of Information Technologies – Bulgarian Academy of Sciences
dochev@iinf.bas.bg

Abstract

This paper presents a multilingual Natural Language Generation system that produces technical instruction texts in Bulgarian, Czech and Russian. It generates several types of texts, common for software manuals, in two styles. We illustrate the system's functionality with examples of its input and output behaviour. We discuss the criteria and procedures adopted for evaluating the system and summarise their results. The system embodies novel approaches to providing multilingual documentation, ranging from the re-use of a large-scale, broad coverage grammar of English in order to develop the lexico-grammatical resources necessary for the generation in the three target languages, through to the adoption of a 'knowledge editing' approach to specifying the desired content of the texts to be generated independently of the target languages in which those texts finally appear.

Keywords

Generation, multilinguality, styles, grammar re-use, evaluation

Introduction

The AGILE project (Automatic Generation of Instructions in Languages of Eastern Europe) was a three-year project (1998—2001) in which a prototype multilingual Natural Language Generation (NLG) system was developed for producing user manuals for CAD-CAM systems in Bulgarian, Czech and Russian. The project involved the Institute of Information Technologies—Bulgarian Academy of Sciences, Charles University in Prague, the Russian Research Institute for Artificial Intelligence, the University of the Saarland and the ITRI at the University of Brighton (coordinators). It built on the success of the previous DRAFTER system, developed by the ITRI with support from two commercial partners, Praetorius Ltd and Integral Solutions Ltd. (Scott & Evans, 1998). DRAFTER stands for "DRafting Assistant For TEchnical writers", and enables the production of draft software manuals in English and in French. Within AGILE, this approach was investigated further, considering a broader range of languages and more sophisticated textual possibilities.

The prototype system developed within AGILE allows users to specify the content of the instructions for carrying out tasks in the CAD-CAM domain. This content is then automatically expressed in each of the three target languages in parallel. The approach of multilingual NLG thus contrasts with the more conventional approach of (MT) in that the starting point is not a source text in one language, but a non-linguistic specification of the content to be expressed (Hartley and Paris, 1997). This specification may draw on one of the target languages, but need not: in our illustrations in the present paper, for example, the AGILE user-interface language was set to English in order to facilitate the presentation. One practical advantage of applying such a multilingual NLG system is therefore that end-users can directly 'author' draft first versions of texts in languages that they do not themselves speak.

The AGILE project also represented the first attempt ever at a comprehensive computational account of Bulgarian, Czech and Russian for the purpose of natural language generation. These grammars are implemented in the framework supported by the Komet-Penman MultiLingual (KPML: Bateman, 1997) environment, and may now be re-used and extended for generation in other domains.

Working with AGILE – an overview

In this section, we give a closer view of the functionality of the AGILE system by providing examples both of the kinds of texts produced and of the style of interaction required for providing the information for generation.

The output of the AGILE document generator

Here are examples—in English, for generality of illustration—of some of the different text types generated by AGILE in the personal style.

Full Instructions

To draw an arc

First start the *ARC* command using one of these methods:

Windows: From the *Arc* flyout on the *Draw* toolbar, choose *3 Points*.

DOS and UNIX: From the *Draw* menu choose *Arc*. Then choose *3 Points*.

Now specify three points of the arc.

1. Specify the start point (of the arc). First enter *endp*. Then select a line. The arc snaps to the endpoint of the line.

2. Specify the second point of the arc. First enter *poi*. Then select a point. The arc snaps to the point.

3. Specify the endpoint of the arc.

As is often the case for NLG systems, AGILE has flexible strategies for deciding whether to express a given 'package' of information in several sentences, as above, or in a single sentence, as below. Note that such variation is not normally considered a required functionality in the context of MT systems, but is a natural target within an NLG system because there the aim is always to produce the most appropriate text given some particular communicative goals.

> …
> Now specify three points of the arc.
>
> 1. Specify the start point of the arc by entering *endp* and selecting a line so that the arc snaps to the endpoint of the line.
>
> …

Overviews

Overviews provide a summary of all the tasks that are documented in the manual. AGILE can produce texts in which they are more or less randomly ordered, as here:

> The system enables you to create a multiline style, to specify the properties of a multiline, to draw a line and arc combination polyline, and to draw an arc by specifying three points.

Alternatively, the tasks can also be grouped according to the objects the user can act upon—e.g. 'multiline'—or the actions the user can perform—e.g. 'draw'.

> The system enables you to create a multiline style, and to specify the properties of a multiline. You may also draw a line and arc combination polyline, and an arc by specifying three points.

Functional Descriptions

These describe the functionality of user interface commands such as buttons in toolbars or dialog boxes. The descriptions can be organised according to the command identity as here:

> The *Polyline* button on the *Polyline* flyout from the *Draw* toolbar (under Windows) starts the *PLINE* command.
>
> The *Polyline* button on the *Draw* menu (under DOS and UNIX) starts the *PLINE* command.

Or the descriptions can be expressed in terms of the action the user must perform.

> Selecting *Add* in the *Element Properties* dialog box adds an element.
>
> Choosing *OK* in the *Element Properties* dialog box saves the style of the multiline element and exits the *Element Properties* dialog box.

Again, which alternative is selected can be decided flexibly depending on the particular goals of the system when presenting the information provided for generation.

The input to the AGILE document generator

The person—usually a CAD-CAM system designer—who is providing the information to be expressed in the target instruction manual creates 'models' of individual user tasks. The task model is built up recursively in structures that mirror the GOAL + METHOD structure of software instructions. These are displayed to the user in the form of nested boxes (Figure 1). Clicking on a slot in a box brings up a menu of available fillers—actions or objects, depending on the nature of the slot. Currently, English, Bulgarian, Czech or Russian may be chosen as interface language: i.e., the language in which slots and their possible fillers are identified to the user.

In order to build a task model describing the procedure of *drawing a line by defining its start and end points*, for example, the GOAL is *draw* and the METHOD is *specify end points*. To specify this, the author will select *draw* from the choices offered as fillers for the goal shown (abbreviated) in Figure 1.

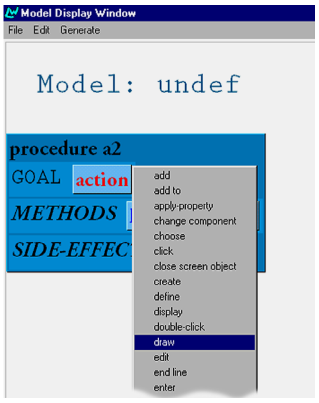


Figure 1. Specifying the GOAL action

Next, the author must specify what is to be drawn (Figure 2). Slots remaining that must be filled before a sufficiently complete specification has been input are signalled by a red label. The context-sensitive menu that appears will display only the names of those objects that are appropriate fillers for the specified goal—i.e., in the present case, those that can be drawn using the CAD-CAM application, e.g. *arc* and *polyline*, but not *UNIX* or *menu*.

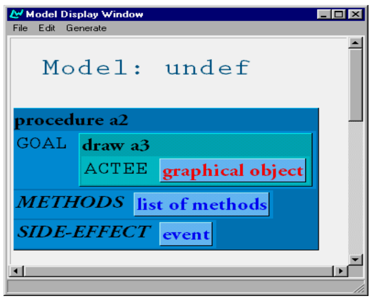


Figure 2. Specifying the GOAL object

Figure 3 shows the task model on the point of completion. The author has fully specified the GOAL and the first STEP in the METHOD—*defining the start* *point*. The action of the second step—*define*—has also been specified and all that remains is to select the object—*end point*.

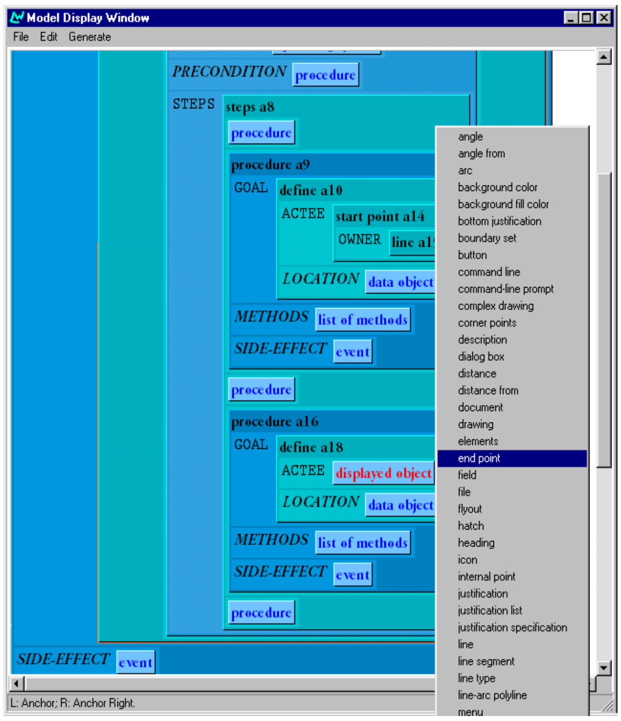
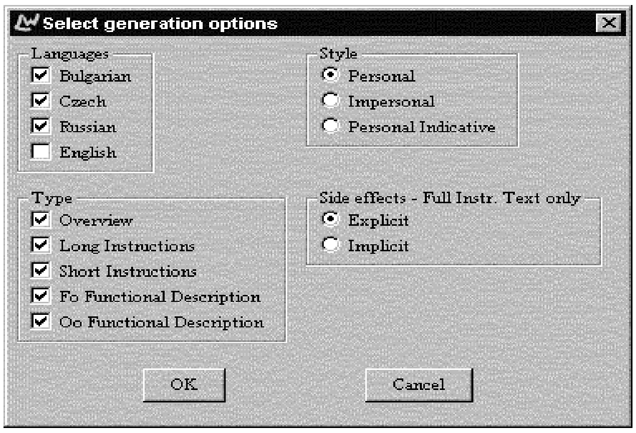


Figure 3. Choosing a value to complete the model

Selection of text type and style

Once the task model has been completed and saved, the author can choose in which languages and styles to have its content expressed (Figure 4). The 'Overview' option applies only when the author chooses to generate the documentation for several tasks at the same time, e.g. to produce a section or chapter of a manual. In these conditions, the table of contents is generated automatically.


Select generation options

Languages
- [✓] Bulgarian
- [✓] Czech
- [✓] Russian
- [ ] English

Style
- [● Personal]
- ○ Impersonal
- ○ Personal Indicative

Type
- [✓] Overview
- [✓] Long Instructions
- [✓] Short Instructions
- [✓] Fo Functional Description
- [✓] Oo Functional Description

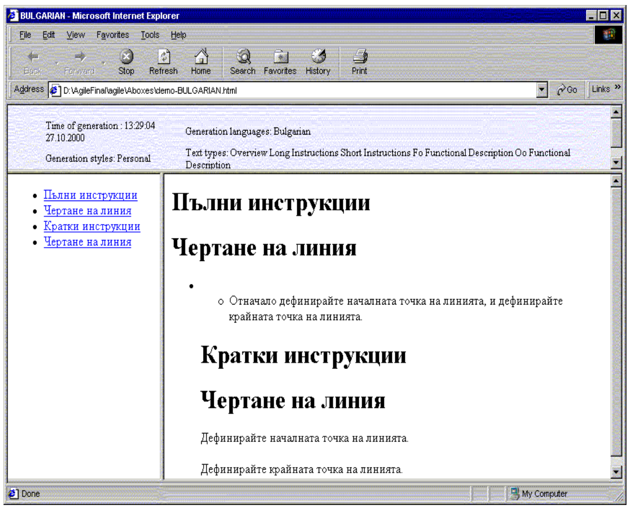
Side effects - Full Instr. Text only
- [● Explicit]
- ○ Implicit

OK Cancel

Figure 4. Selecting languages, text types and styles

Display of the output

The documents generated include HTML markup that is appropriate for the text structure of the texts produced and can accordingly be displayed in standard browsers (Figure 5). A separate window is opened for each output language. The pane on the left contains the list of contents, represented by hyperlinks to the corresponding sections of the manual displayed in the pane on the right. The documents can be saved and edited further as required.


BULGARIAN - Microsoft Internet Explorer

Address D:\AgileFinal\agile\Aboxes\demo-BULGARIAN.html

Time of generation : 13:29:04
27.10.2000

Generation styles: Personal

Generation languages: Bulgarian

Text types: Overview Long Instructions Short Instructions Fo Functional Description Oo Functional Description

- Пълни инструкции
- Чертане на линия
- Кратки инструкции
- Чертане на линия

Пълни инструкции

Чертане на линия

-
  - Отначало дефинирайте началната точка на линията, и дефинирайте крайната точка на линията.

Кратки инструкции

Чертане на линия

Дефинирайте началната точка на линията.

Дефинирайте крайната точка на линията.

Done

My Computer

Figure 5. Generated text in Bulgarian.

Planning texts of different types and styles

Our analysis of the corpora of instructional texts for each language (see below) revealed marked differences between languages when expressing the same content. One of the advantages of multilingual NLG over MT is that the style of the output text can be made fully appropriate to the target language, with no interference from structures more appropriate to some other (source) language. In the AGILE project, we aimed to produce texts that are sensitive to the stylistic requirements of not only the output language, but also the various sections of the manual. For example, the full instructions section can be generated in either a personal or impersonal style. In the former, the reader (software user) is addressed directly; in the latter, the tone is more formal.

A text planner, or Text Structuring Module (TSM), is responsible for constructing a text plan that matches the genre and style features selected by the 'author' to the information that she has specified via the graphical interface; the plan structures the information accordingly. A sentence planner interprets the text plan to create plans for sentences. By generating a sequence of sentence plans, and having a lexico-grammar generate each one as it comes, AGILE produces the entire text that expresses the content specified by the user.

The TSM's approach to discourse structuring combines elements of Halliday's Systemic Functional Grammar (SFG: Halliday, 1985), Mann and Thompson's Rhetorical Structure Theory (RST: Mann & Thompson, (1988), and the Prague Functional Generative Description (FGD) (Sgall et al. 1986). Importantly for the three Slavic languages represented in AGILE, the TSM is able to appropriately manipulate textual (information) structure which can then be expressed through contextually appropriate word order. The synthesis of the three approaches resulted in a design that adds value over and above the contributions made by each individual approach.

Developing unified grammars for Bulgarian, Czech and Russian

We developed new computational grammars for each of the languages targeted. The approach was to re-use a large-scale, broad coverage English grammar (the Nigel grammar) in order to construct grammars of a similar scale for Bulgarian, Czech and Russian. Whereas the production of new language resources on the basis of existing grammars has been carried out previously (cf. Rayner *et al.,* 2000), new within AGILE was the fact that the languages addressed are not closely related typologically to the source language. The theoretical motivation for this approach was originally set out in Bateman et al. (1991), where a particular form of *functional typology* was proposed as an effective means of re-using grammatical (and other) descriptions across a range of languages broader than that allowed by structural typological approaches. This also supported a novel grammar development strategy within AGILE in which the individual partners each worked independently on inherently multilingual 'core areas' which were subsequently combined within a single multilingual grammatical resource.

The priorities for grammar development were set by a corpus-based contrastive analysis of (non-translated) instructional texts in the target languages; this resulted first in the creation of sub-language grammars for the domain. However, a primary goal of the AGILE project was to develop *re-usable* lexico-grammatical resources that are suitable for multilingual generation in Bulgarian, Czech and Russian in other domains as well. For this we needed a framework that would be accessible to the partners in terms of the linguistic concepts used, adaptable to the project languages, and interfaceable with other components of the complete application. It was these considerations that led us to adopt the KPML tactical generator and development environment (Komet-Penman Multilingual: Bateman, 1997). As well as showing some commonalties with the Eastern European tradition of functional linguistics, KPML is especially geared towards the development of multilingual grammars and implements the notion of functional typology mentioned above in order to offer various ways of sharing the computational description of an existing grammar with new languages that are added to the system.

The three target languages of AGILE naturally share many features by virtue of their common Slavic origins, but it is also striking that, when viewed functionally, they additionally share very many features with the original source English grammar. All four grammars (the English and the newly developed Bulgarian, Czech and Russian) then exhibit considerable overlap without committing to identity where it would be inappropriate. If we consider, for example, the equivalents generated for the sentence *Enter the 'Draw' command and click on the 'OK' button to start the program,* we find that the overall structure is very similar in the corresponding Bulgarian, Czech and Russian sentences but there are also significant differences: whereas Russian and Bulgarian both use a dependent clause to express the 'purpose' element, Czech uses instead a prepositional phrase. Moreover, there are finer differences between the Russian and Bulgarian: the Russian adopts a non-finite clause construction in the dependent clause, whereas the corresponding Bulgarian clause is finite. An extract from corresponding grammatical structures as generated is given in Figure 6.

More generally, the unified grammars handle similarities and differences across their languages such as:

- free word order in target languages, governed by the same principles, although differing slightly in surface realization;
- same basic choices of aspect in target languages with slight variation in textual instantiations;
- specific agreement phenomena at clause and nominal group level;
- some specifics of spatio-temporal prepositional phrases in Slavic languages (distinguishing two types of locations according to the number of dimensions instead of three in English; realization in Bulgarian by choice of preposition, and in Czech and Russian also by case);
- possibility for subject dropping in Czech and Bulgarian declarative clauses etc.

All of these areas represent traditional problems for MT approaches. The multilingual generation account places their description within the expected variation found across the grammatical systems of distinct languages. The effectiveness of this procedure, both for grammar description and for distributed development, has provided considerable further support of the effectiveness of the functional typological approach to resource development as supported by KPML.

Evaluating the AGILE system

Compared with Natural Language Understanding, rather little work has been done on evaluation in Natural Language Generation, particularly of end-to-end systems that go all the way from content specification to text generation. Here our previous experience of evaluating DRAFTER was helpful. For AGILE, we designed an evaluation scenario which addressed: the *usability* of the integrated system for creating and editing text specification models; and two dimensions of text quality—the *grammaticality* of the output texts and their *acceptability* as a first draft of a user manual.

The results of our evaluation showed that, with training, users are able to write documentation for the CAD/CAM domain in their own language with the aid of AGILE and that the quality of output texts is sufficiently good for their inclusion into drafts of high-quality manuals. This was true for all three localised versions and for all the subjects tested.

Usability

The evaluators were, at each site, IT specialists rather than authors or linguists. The evaluation was preceded by training in the underlying concepts and in the use of the system. The training was supported by a Conceptual Tutorial, introducing basic concepts of authoring documents in AGILE, and a Training Manual, defining

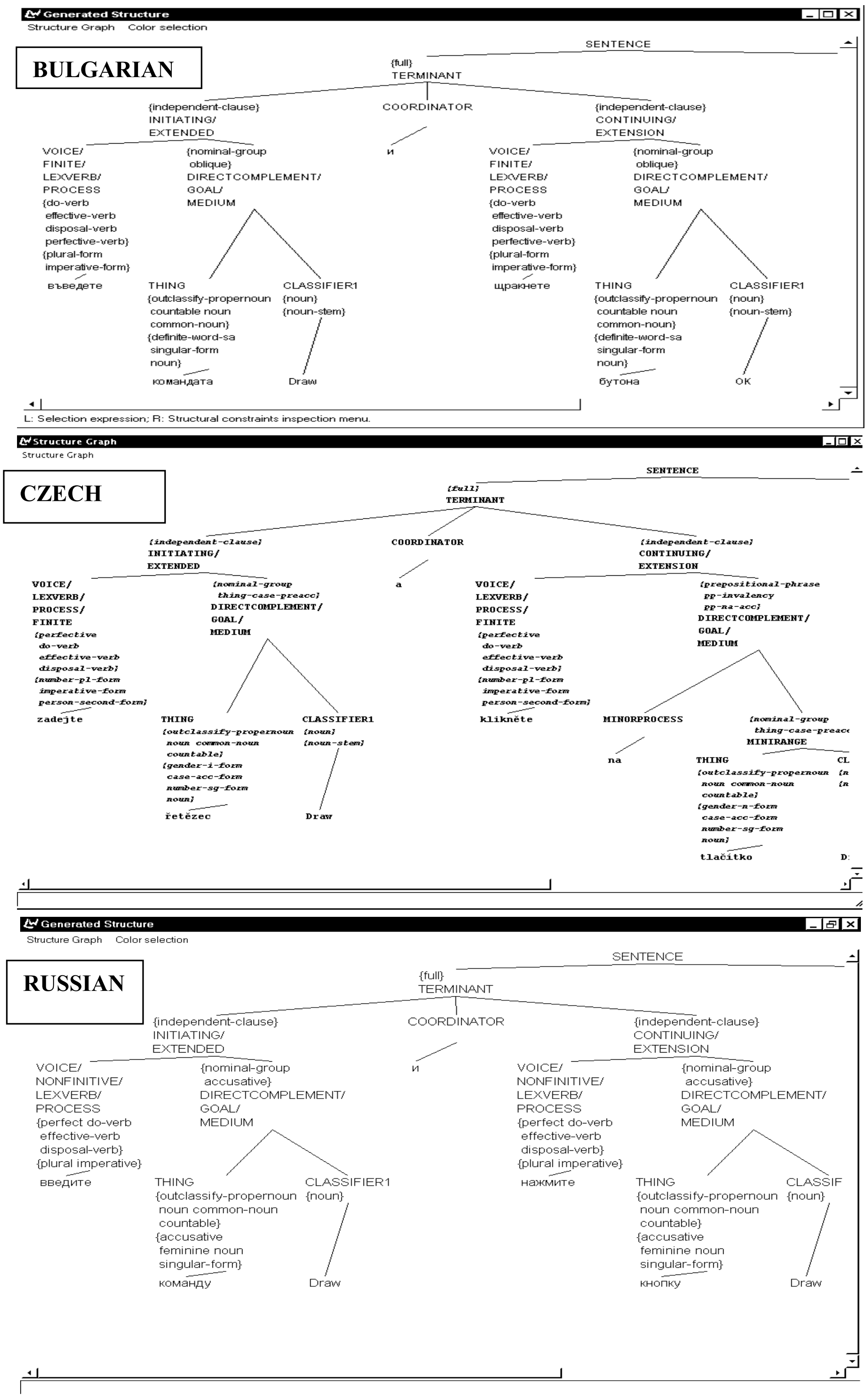


Figure 6: Extract of generated grammatical structure for Bulgarian, Czech and Russian within the AGILE domain.

methods for specification of a fragment of a manual in near to real authoring conditions. The testing session comprised five exercises with a time limit. The evaluators edited and created both simple and complex models for individual CAD-CAM tasks, and composed sets of models for related tasks. Partially built models were sent from one site to another for completion.

The knowledge editing interface was judged to be rather clumsy however. Detailed analysis of the task models produced by the evaluators showed that the most of them were correctly structured. But in some cases evaluators had wrongly created multiple instances of a concept instead of multiple pointers to a single instance.

Acceptability

The evaluators were native speakers of the language they judged, and experienced in writing and/or translating software documentation. Following methods used to evaluate machine translation systems, they were asked to rate the quality of the output on a four-point scale—Excellent, Good, Poor, Terrible. They also rated the Full Instructions relative to human-authored reference texts.

The texts generated by AGILE in all three languages were judged to be of comparable quality to similar texts found in good commercial manuals. Functional Descriptions, Full Instructions and Quick References were judged Good to Excellent, while Overviews, were rated Poor to Good.

Grammaticality

For each of the three languages, we obtained judgments from two native speakers trained in the linguistic description of their own language. In order to keep the content constant across the three languages, the six judges evaluated texts originated from the same composite task model. Their evaluations covered all of the running-text types, using 16 error categories. For Bulgarian and Russian, these text types were made available in two stylistic variants: Personal and Impersonal. Since Czech has two ways of expressing personal style, the Czech judges evaluated three variants: Personal Indicative, Personal Explicit and Impersonal Explicit.

Almost no grammatical errors were identified by the judges, other than errors classified as ones of word order, a well-known difficulty in Eastern European languages. Even then, some were thought to be stylistic rather than syntactic.

Coverage

We employed a method of grammar development that was both *instance-oriented* and *system-oriented*. Instance-oriented means basing development on a corpus of texts from the target sublanguage. System-oriented means building the computational grammar with a view to the language system as a whole so as to encourage re-use. We therefore assessed the extent to which the multilingual resources developed within the project cover those grammatical constructions found at four increasingly general levels (Figure 7): the texts extracted from CAD/CAM manuals that served as a ‘target’ for the AGILE prototype; software manuals in general; other instructions; and general language. In the figure, grey-shading suggests additionally the range of coverage; the target texts are completely covered, software manuals in general less completely, and so on.

Although there are obviously still significant gaps in coverage for a general grammar (negation, for example), the resources developed within AGILE still present a substantial first-approximation to a general generation capability that can simplify the construction of further systems in both related and new domains.

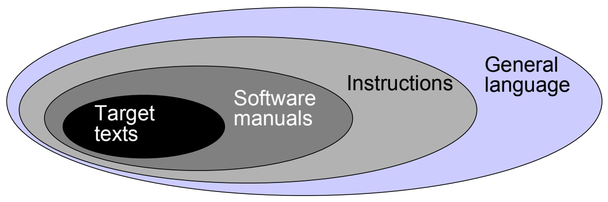


Figure 7: Coverage of lexico-grammatical resources

Finally, we should note that the use (and accordingly the relevance of providing an implementation) of certain constructions was also sometimes impeded by the inability to represent their semantics in the Domain Model, which for AGILE is an ontology of concepts from the CAD-CAM domain. As always in NLG, improvements in the adopted knowledge representation can be expected to improve the range and quality of the texts that may be produced.

Acknowledgements

The work described in the paper has been supported by EC INCO-COPERNICUS project PL961104 AGILE “Automatic generation of Instructions in Languages of Eastern Europe”. The authors express their gratitude to all the participants in the AGILE project, upon whose work this paper reports.

References

Bateman, J.A., Matthiessen, C.M.I.M., Nanri, K. and Zeng, L. (1991). The re-use of linguistic resources across languages in multilingual generation components. In: *IJCAI‘91,* pp. 966—971.

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

Halliday, M.A.K. (1985). An Introduction to Functional Grammar. Edward Arnold, London.

Hartley, A. and Paris, C. (1997) Multilingual document production. *Machine Translation* 12(1-2), 109-129.

Mann, W.C. and Thompson, S.A. (1988). Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text* 8, 243--281.

Scott, D. and Evans, R. (1998). Multilingual Document Management without Translation. *ELSNEWS: The newsletter of the European Network in Language and Speech* 7, 2—3.

Rayner, M., Carter, D., Bouillon, P., Digilakis, V. and Wirén, M. (2000). The spoken language translator. Cambridge University Press.

Sgall, P., Hajičová, E. and Panevová, J. (1986). The meaning of the sentence in its pragmatic aspects. Reidel.