AGILE

Automatic Generation of Instructions in Languages of Eastern Europe

---

Title ***Text Structuring Specification for the Final Prototype***

Authors Tony Hartley
Ivana Kruijff-Korbayová
Geert-Jan Kruijff
Danail Dochev
Ivan Hadjiiliev
Lena Sokolova

Deliverable *TEXS3*

Status *Final*

Availability *Public*

Date *January 27, 2000*

INCO COPERNICUS PL961104

Abstract:

This deliverable specifies various tasks for the Text Structuring Module for the final prototype. The coverage of the Text Structuring Module (developed in WP5.2) is extended both in depth and in breadth. For the full instructions in procedural style, the deliverable specifies additional stylistic variations, and more elaborate mechanisms for introducing and realising discourse relations. Furthermore, other new genres are introduced: summaries, basic step instructions, function descriptions, overviews, and tables of content. In order to generate these different genres, the user will not be required to provide new content descriptions (i.e., A-boxes): the main point is that text plans for all these different genres can be derived from the already existing A-boxes specifying procedural-style instructions. The generation of such "meta-texts" like overviews and function descriptions from multiple existing content specifications constitutes a rather novel effort in the field of NLG. The only other work with a comparable aim we are aware of is the generation of summaries from multiple input documents discussed in (Radev and McKeown 1998).

More information on AGILE is available on the project web page and from the project coordinators:

URL:	http://www.itri.brighton.ac.uk/projects/agile
email:	agile-coord@itri.bton.ac.uk
telephone:	+44-1273-642900
fax:	+44-1273-642908

Table of Content

Table of Figures

1. Introduction

1.1 Aims

The aim of the AGILE work package on text structuring (WP 5) is to develop a Text Structuring Module that is capable of planning texts of the form and style as found in the project domain, namely CADCAM software manuals. In the previous stages (WP 5.1 and WP5.2) we started by carrying out corpus studies. These corpus studies aimed at identifying the core aspects of text structure in instructions for procedural tasks ranging in complexity from simple (TEXS1) to elaborate (TEXS2). The core aspects we identified concerned:

- the rudimentary building blocks from which a structure for a text could be build (a *text plan*);
- the different ways in which content could be realised as clauses (*sentence planning*); and,
- the variations in styles that could be used to realise a text (bridging the gap between strategic generation and tactical generation).

Based on the specifications arising from corpus studies, a Text Structuring Module (TSM) has been developed. The current TSM is capable of generating text plans, together with the appropriate sentence plans, that cover the required variations in style and complexity as illustrated by the target texts given in (TEXS2), modulo phenomena not covered (or not coverable) by the grammars. The architecture of the TSM has been described in (TEXM1) and (for its current form) in (TEXM2).

So far, we have been considering full instructions in procedural style that describe how a single task can be achieved. An example of such a procedural task could be the construction of a particular CADCAM object like a "multiline". In this deliverable we extend our scope on text structuring in the following ways:

- Variations in the *style* in which elements of a text plan for a procedural text can be realised: for example, different styles of titles, running text versus lists enumerating steps, and the use of more complex tense and aspect.
- Variation in the *complexity* of the realisation of discourse relations and discourse markers. We will consider more types of discourse aggregation and syntactic aggregation (see below for more details). Furthermore, we will and introduce temporal discourse markers (like "after … now …").
- Variations in the *types* of texts that can be generated from a single content specification (A-box): for example, besides the procedural texts considered so far, non-procedural texts like overviews and descriptions. These different types of texts can be found in software manuals, in addition to the procedural text that usually makes up the body of a software manual.

Stylistic variation in the generation of elements of the text plan requires the proper imposition of realisation constraints on sentential generation, obtained through inclusion of these constraints in sentence plans.

Abstractly put, aggregation addresses the issue of combining two or more structures into a single linguistic structure which contributes to sentence structuring and construction (Reape & Mellish, 1999). In our case, aggregation deals with the combination of linguistic

structures that realise the content expressed by PROCEDUREs, as identified by the TASKs at the leaves of a text plan. We consider two different types of aggregation here, following the discussion by Reape and Mellish (1999). First of all, we extend the mechanisms for discourse aggregation, which is oriented at supporting rhetorical structure. In our case this means that we form a combined linguistic structure in which the rhetorical relation between the two components is made explicit (e.g. as can be expressed by "in order to"). Furthermore, we introduce more elaborate mechanisms for grouping of conceptually related content. This kind of grouping can be expressed syntactically through coordination or disjunction.

Finally, the generation of different genres concerns text planning proper, since each different type of text will require the generation of a text plan that is appropriate for that genre.

1.2 Overview

The discussion in the deliverable is structured as follows. Because the work in WP5.3 is based on the results of WP 5.2, we begin in section 1.3 by briefly discussing the current TSM. In that discussion we will primarily focus on how we can reuse, build forth on, the resources developed so far.

In Chapter 2 we discuss the texts to be covered in the final prototype. First of all, we distinguish a number of text types to be generated (Section 2.1), and then we discuss aspects of stylistic variation we are taking into account for the final prototype (Section 2.2). Finally, we sketch the correlations between them (Section 2.3). The subsequent chapters are concerned with detailed descriptions of individual text types, the stylistic variations available within the text type, and the formal specifications of the text structuring involved in generating texts of the respective types. Chapter 3 presents those aspects of full procedural instructions that were not fully covered in the intermediate prototype: in Section 3.1 we discuss aggregation, and in Section 3.2 we discuss the generation of explicit discourse markers. Chapter 4 concerns goal-oriented and step-oriented abbreviated procedural instructions. Chapter 5 concerns function-oriented descriptive texts, Chapter 0 concerns full and abstract overviews and Chapter 7 concerns tables of contents.

In addition to the main discussion there are two appendices. The first appendix describes the inquiries that are available in the current TSM. These inquiries form the basis for our formal specification, since these inquiries provide the necessary basic functionality for inspecting and accessing content in an A-box. The second appendix gives a set of target texts for the final prototype, against which we will be able to assess the further development of the TSM.

1.3 The Text Structuring Module

The AGILE Text Structuring Module (TSM, implemented for the intermediate prototype and described in TEXM2) consists of two parts.

One part concerns text planning, and essentially consists of a network of systems (region) implemented in KPML. A text plan is built by traversing the region, in a manner that is analogous to the way sentential structures are generated. While traversing the region, the structure of the A-box is examined and, depending on how content is configured and what content is present, a text plan is constructed.

The second part regards sentence planning. It is convenient to think of the text plan as a hierarchical structure, with individually identified pieces of content at the leaves of that text

plan structure. The sentence planner creates SPLs (Sentence Planning Language expressions) for these pieces of content, where an SPL may either realise just one leaf (leading mostly to a single-clause sentence) or aggregate several leaves, depending on the way these leaves are related in the text plan.

The text planner being a region implemented in KPML, its basic ingredients are systems, choosers, inquiries, and inquiry-implementations. The systems concentrate on inserting, or extending, the components of the text plan. For the instructional texts in procedural style, generated by the intermediate prototype, these basic components are the TASK-TITLE, TASKs, and INSTRUCTIONs. The TASK-TITLE corresponds to the topmost goal of the A-Box, a TASK corresponds to a PROCEDURE (and is identified with the PROCEDURE's GOAL), and an INSTRUCTION corresponds to a METHOD. PROCEDURE and METHOD are the so-called *configurational concepts* of the Domain Model. These concepts are used to structure the content in the A-Box. Consequently, each TASK can have a SIDE-EFFECT, and one or more TASK-INSTRUCTIONS. These are mapped from a PROCEDURE's SIDE-EFFECT and METHODS, respectively. Similarly, an INSTRUCTION can have a CONSTRAINT and/or a PRECONDITION, and must have a non-empty list of INSTRUCTION-TASKS. These come from a METHOD's CONSTRAINT, PRECONDITION, and SUBSTEPS respectively.

Most systems in the currently implemented region check in the A-box for either of the following two things. Either a system checks whether a particular configurational concept is present at the current locus of attention, and as such should be mapped to the appropriate text plan component which should then be inserted into the text plan. Or, it checks whether a list-structure like TASK-INSTRUCTIONS or INSTRUCTION-TASKS has a non-empty tail, and as such should be extended. Inserting or extending a text plan component is done by a system, which depends for its decision on a chooser. These choosers say either "yes do insert/extend" or "no don't do anything". In order to make these decisions, the choosers rely on inquiries, and these inquiries themselves are implemented as functions that look into the A-box. The current TSM has a number of general-purpose inquiries as well as more specific inquiries, all of which have been documented and can be reused in implementing more complex inquiries to underlie new systems and their choosers.

Most importantly, the inquiries can be used to examine parts of the A-box to any arbitrary level of detail. The TSM mostly queries the structure of the A-box (i.e. presence of values in slots of configurational concepts) but at times does descend to the level of content specified under a configurational concept. An example of this can be found in the sentence planner and the way constraints are ‘SPLized’ (expressed in SPLs)

In other words, the current TSM can be conceived of as providing a library of functions that can be readily used to query the A-box in order to assist in making decisions about building a text plan. The approach we want to propose for the final Text Structuring Specification deliverable (TEXS3) is based on the idea of reusing these functions.

1.4 Specifying the decisions for creating more complex text plans

As is hopefully clear from the previous section, and the way text structuring in general is done in AGILE (TEXM2), the structure of a text plan is an interpretation of the way content has been structured in the A-box that the text is to realise. Looking at the organisation of content in the part of the A-box under consideration, a system in the text planner's KPML region decides to extend the text plan in one way or the other.

This approach can be readily extended to creating the more complex variations that we envision for the final prototype. Like the text plans for the intermediate demonstrator, the resulting, more complex text plans will have to be built by interpreting the structure of an A-box. Fortunately, the necessary inquiry-implementations are available already in the current TSM. The way we propose to cover the creation of more complex text plans is to specify systems, choosers and their inquiries, in the formal way we adopted already in earlier deliverables (e.g., for WP6.2). For the specification of the inquiries, the important idea is that we can *reuse* the existing inquiries, either on their own or as building blocks for more complex inquiries and choosers.

2. Text types and stylistic variation in the final prototype

In the TEXS2 deliverable we discussed several variations for instructions in procedural style. For example, we considered textual styles like personal versus impersonal, various ways in which particular text elements like document titles could be realised, and whether additional information like side-effects should be realised explicitly.

Similarly, for the purpose of the final prototype, we focus on software documentation. However, as our corpus analyses revealed, different styles of conveying a given content can be employed, and different kinds of documentation focus on different chunks of the content, and present the content in different organisations. This means that the same content (set of A-boxes) can serve as input to generating various types of texts.

In this chapter, we describe the range of stylistic variation we take into account in the final prototype texts. We first introduce the text types supported in the final prototype, and then we discuss various ways of realising given content in general. The individual text types and the range of stylistic variation available with respect to each of them will be discussed in detail in subsequent chapter of this deliverable, along with examples and the necessary formal specifications.

2.1 Text types (genres)

Let us first overview the terminology we have been using in Agile. It has been introduced in more detail in the TEXS2 deliverable. There, we followed (Hartley and Paris 1996, H&P henceforth).

H&P propose that for the instructional texts as found in software manuals an obvious option is to explore the communicative purpose. The communicative purpose does not need to be constant throughout a manual, as H&P observe. They discern three different **functional sections** in their corpus of Macintosh manuals:

- Tutorial: concerning excercises for new users
- Series of step-by-step instructions for the major tasks
- Ready-reference summary of the commands

The functional sections as differentiated by communicative purposes according to H&P can be considered to constitute different **genres** in terms of Martin (1992). Genres are distinguished by their communicative purposes, and they control text structure and its realisation. According to H&P, genre is responsible for the selection of a text structure in terms of task elements.

H&P also suggest that it is useful to distinguish between two more specific communicative purposes in step-by-step sections, and we adopted this approach too. So, summarizing, in TEXS2 we used the following classification of text types (genres) found in step-by-step sections of the AutoCAD manual:

- **procedure** (we also refer to it as **instructive**), where the purpose is to enable the reader to perform a task
- **elaboration** (we also refer to it as **descriptive** or **informative**), where the purpose is to increase the reader's knowledge

We also said that the AutoCAD manual sections entitled "Further possibilities" and the introductory parts of sections would correspond to the informative text type, while those

pieces of the texts which we included in our WP3 corpus (see the CORP deliverable), and were marked as Procedure would be of the procedural text type.

While the initial demonstrator and the intermediate prototype aimed to generate procedural instructions, we now broaden the scope to other types of texts that can be found in software documentation. For the purpose of the final prototype, we are extending the coverage of text types (genres) in the following way:

1. Within **step-by-step**, we generate the following two text-types (sub-genres):

- **procedural type** (procedure, instructive):

  (a) **full** instructions in procedural style (procedural instructions)

  (b) **abbreviated** instructions in procedural style (procedural instructions):
  (i) summary instructions: goal-oriented = the why without the how
  (ii) basic steps: step-oriented = the how without the why, also commonly called "cheat-sheets", or informally characterised as robotic

- **descriptive type** (elaboration, informative):

  (a) **overviews** of the documented tasks:
  (i) full and
  (ii) abstract
  (these are like the texts in the introductory parts of sections in the AutoCAD manual, or like the texts found under "Further possibilities")

  (b) **explicit side-effects** might, if one wants, be considered as pieces of elaboration within procedural instructions (this is still in line with H&P)

2. Within **ready-reference summary of commands**, we generate the text-type (sub-genre) of **functional descriptions** (descriptions in function-oriented style): these are like the texts that can be found under "Related commands" in the AutoCAD manual). Given that we only try to convey content that we can distill out of the A-boxes, we only generate a particular kind of functional description, namely what commands do, or what selecting an interface object does.

3. As an extra genre, not discussed by H&P, we also generate a **Table of contents**.

Texts of all these types are to be generated from the same input, namely from collections of A-boxes. Let us now characterise each of the text types briefly (much more detailed discussions are provided in Chapters 3, 4, 5, 0, 7).

Full instructions in procedural style are taken over from the intermediate prototype, and we try to provide more variation in the content realisation (see below). **Abbreviated instructions** are similar in style to the full instructions, but not all the content of the A-boxes is realised. They are procedural texts that are essentially abbreviations of the larger instructional texts considered in WP5.2. We consider two possible ways in which a the content of an A-box can be presented in an abbreviated way: **summaries (Goal-oriented abbreviated instructions)** focus on the higher level goals in the A-box, leaving out the steps to accomplished ("the why without the how"), and **basic steps (Steps-oriented abbreviated instructions)** focus on these steps themselves and leave out the higher level goals ("the how without all the why's"). Summary instructions thus convey only the more abstract parts of the content, while steps instructions convey the most concrete steps within procedures. The examples below illustrate these text styles on the basis of the text 2 from the intermediate prototype set of texts.

Summary

To draw a line and arc combination polyline

First draw a line segment.

Then draw an arc segment.

Then draw another line segment.

Finally, press *Return* to end the polyline.

Basic steps

To draw a line and arc combination polyline

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*.

Specify the start point of the line segment.

Specify the endpoint of the line segment.

Enter *a*.

Select *OK* in the *Arc mode confirmation* dialog box.

Specify the endpoint of the arc.

Enter *l*.

Select *OK* in the *Line mode confirmation* dialog box.

Enter the distance of the line in relation to the endpoint of the arc.

Enter the angle of the line in relation to the endpoint of the arc.

Press *Return*.

Descriptions in function-oriented style constitute the first type of three types of non-procedural texts we consider here. As our corpus analyses (CORP, TEXS2) revealed, descriptions in function-oriented style are very common in original software documentation in Czech, Russian and Bulgarian; in Czech, this text type is in fact prevalent. Descriptions convey the A-box content in a different way than the procedural instructions. Rather then providing step-by-step instructions, these texts tell the user what can be achieved by various command or actions, and what are the functions of particular interface objects. We consider two types of function descriptions, being descriptions of what Goal(s) a Method (user-action) can accomplish and descriptions of what Goal(s) a GUI object (for example, a command, menu item, or toolbar icon) achieves. The following examples illustrate the two styles of descriptions.

Action-based descriptions

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.

Object-based description

The *Add* button in the *Element Properties* dialog box adds an element.

The *OK* button in the *Element Properties* dialog box saves the style of the multiline element and exits the *Element Properties* dialog box.

Descriptive texts might be restricted to describing the content of a single A-box, though more interesting texts arise when a collection of A-boxes can be considered.

For overviews and TOCs this is no longer an option - for these text styles we will have to consider collections instead of individual A-boxes.

Overviews of the documented tasks convey the A-box content in a way similar to the function-oriented descriptions. They differ from them in concentrating on the user's tasks, and the ways they can be accomplished rather than on the functionality of the interface objects. Overviews often contain generalisations (abstractions) over the A-box contents. They also group the content realisations into logical clusters, either by the type of task performed by the user, or by the object(s) affected by the performed operations. Task overviews are an integral part of every software manual. They can be found in introductory chapters as well as in introductions to individual sections of the manual. An overview can present the content in the same order as it appears in the A-boxes, or in another order. The example below illustrates a simple overview.

The system enables you to create a multiline style, to specify the properties of a multiline, to draw a line and arc combination polyline, to draw an arc by specifying three points, and to define a boundary set in a complex drawing

Tables of contents are another necessary part of every piece of software documentation. Overviews appear as running text - TOCs present similar information, only then in a different layout. It is sensible to think of organising the TOC in different ways, e.g., by the sequence of procedural instructions as given in the manual (headings), by the interface objects (functional descriptions), or by the tasks (overviews). In this way, the TOC relates to the content conveyed in all the previous text types, but it present only a skeleton of the information, and it presents it in a different format. This is illustrated in the example of a simple TOC below.

1. Creating a multiline style	3
2. Specifying the properties of a multiline	5
3. Drawing a line and arc combination polyline	7
4. Drawing an arc by specifying three points	8
5. Defining a boundary set in a complex drawing	10

The generation of instructions in procedural style (either full or abbreviated) can be naturally performed and meaningfully demonstrated using a single A-box as the input. On the other hand, the remaining text styles, i.e. functional descriptions, overviews and tables of content, are more appropriately applied to larger inputs, most naturally thought of as collections (sets, lists) of A-boxes in which various tasks are modelled. It would of course be possible and meaningful to generate instructions in procedural style on the basis of an A-box collection. Most straightforwardly, this would yield a sequence of single A-box procedural texts. In order to support generation of any texts from a collection of A-boxes rather than a singe A-box, the functionality of both the IFACE and the CADCAM region needs to be extended. We need to load and traverse a list of A-boxes. (Chapters 6 and 7 discuss these issues in more detail.)

The generation of function descriptions, overviews and tables of content which we aim at for the Final Prototype amounts to generating "meta-texts" on the basis of multiple input content specifications. The only other work we are aware of which has a comparable goal is that of Radev and McKeown (1998). They discuss a methodology for generating summaries of news coming from multiple sources, with a focus on the domain of news on terrorism. As Radev and McKeown point out, the approach they present is fairly unique in that, bar a

few exceptions, most available summarizers are focus on the generation of a summary from a single article (source). It appears that this holds in general for generation of "meta-texts" like those we will generate in the Final Prototype, and our efforts can therefore be regarded as original in the light of the state-of-the-art in NLG.

2.2 Stylistic variations

For the purpose of the intermediate prototype, we allowed for different text styles to be used within a genre. These styles were chosen to differ in:

- lexicogrammatical features, i.e., personal vs. impersonal style
- distribution and lexical realisation of content, in particular explicitness of content, i.e., realisation with vs. without side-effects

The discussion of stylistic variation in the final prototype again concerns the above two factors, but we break them apart into more details. We discuss variation in:

- lexico-grammatical features (personal vs. impersonal style)
- realisation of deontic modality
- explicitness of content realisation
- linear organisation
- aggregation
- marking of sequences
- numbered list vs. running texts

In general, the choice of a particular style of realisation may be determined by the user's requirements, or by the complexity of the content being expressed. The variations we take into account for the final prototype concern the following aspects:

- lexico-grammatical features pertaining to personal vs. impersonal style (Voice, Mood, Person, Number)
- lexico-grammatical realisation of deontic modality (possibility, necessity)
- explicitness of content realisation: explicit vs. implicit side-effects, various degrees of full vs. abbreviated content expression
- linear organisation of the content realisation, in particular whether realisation of a Goal precedes the realisation of its accompanying Methods or the other way round
- aggregation of the realisation of content units into clause complexes, exploiting various relations, in particular RST-Means, RST-Purpose, RST-Sequence, Conjunction and Disjunction.
- explicit marking of sequences by discourse markers
- numbered lists vs. running texts

Stylistic variation applies *within* each text type (sub-genre). In each of the above mentioned text types, the realisation of a given content may vary, so the range of possible variation is not the same in all the text types. In the following sub-sections, we explain the variations in general, irrespective of text type. At the end of this section we provide a table

which correlates the stylistic variation and text type. Details of the possibilities within each text type are discussed in Chapters 3.1, 3.2, 4, 5, 0, 7.

2.2.1 Lexico-grammatical features of personal vs. impersonal style

This variation mainly concerns the choice of Voice and Mood. It is retained from the intermediate prototype, and has been discussed in detail in TEXS2. We have so far provided the following alternatives, common to Bulgarian, Czech and Russian:

- personal style: imperative mood, polite imperative (second person plural)
- impersonal style: indicative mood, medio-passive voice

There are additional possibilities, which we would like to cover in the final prototype. Personal style of procedural instructions in Czech very often employs indicative mood in active voice in first person plural:

Cz:
Abychom čáru ukončili, stiskneme Enter.
so-that-would-1pl line end-1pl press-ind-1pl Enter
In order to end the line we press Enter.

Also second person plural is possible, and is used in some manuals:

Cz:
Abyste čáru ukončili, stisknete Enter.
so-that-would-2pl line end-2pl press-ind-2pl Enter
In order to end the line you press Enter.

In Russian and Bulgarian, first person plural is not common to express instructions. It is a style of mathematical texts, including the speaker as Actor of a process. If used in instructions, it would indicate a change of style, for instance, it could be used to mark the author's proposal as in the following example:

Ru:
Приведем структуру слоев графического редактора.
Describe-1pl structure-acc layers-gen graphical editor-gen.
Let’s describe the structure of layers in the graphical editor.

Bu:
Да разгледаме структурата на мултилинията.
Let's consider-1pl structure-def of multiline-def.
Let's consider the multiline structure

Similar observations concerning first person plural hold for Bulgarian. It is sometimes found in recipes or in a lecturing style, but normally not in technical manuals.

The realisation in second person plural is possible in Russian and in Bulgarian only with an explicit modal element, where a choice from a set of alternatives is described:

Ru:
Вы можете выбрать один из следующих методов.
You can choose-inf one of following methods
You can choose one of the following methods.

Bu:
Може да изберете един от следните методи.
Can to choose-2pl one of following-def methods.
You can choose one of the following methods.

In impersonal style, Czech and Russian can employ infinitive constructions:

Cz:
Stisknout Enter, aby se čára ukončila.
Press-inf-1pl Enter so-that-would-1pl refl line end-3sg

Ru:
Нажмите Enter, чтобы завершить рисование линии.
Press-imper Enter in-order-to end drawing-acc line-gen.
Press Enter in order to end the line.

With respect to the realisation of headings in procedural texts, we have so far used only nominalisations. Another possible variation, which we have encountered in the Microsoft Word manuals in Czech, concerns the realisation of headings in interrogative mood. In questions, the various choices of Voice, Person and Number present in personal vs. impersonal style can be reflected:

Cz:

1. *Jak uložit soubor?*
   How save-inf file
   How to save a file?
2. *Jak se uloží soubor?*
   How refl save-3sg file
   How does a file save?
3. J*ak uložíme / uložíte soubor?*
   How save-ind-1pl / save-ind-2pl file?
   How do we/you save a file?

Bu:

4. *Как да запомните файла?*
   How to save-2pl file-def?
5. *Как се запомня файл?*
   How refl save-3sg file?

The use of interrogative mood in headings is also possible in Bulgarian. It reflects a rather informal style. It is not possible in Russian, except perhaps in a popular(ising) style.

For Russian, it is most appropriate to realise headings by partial nonfinite purpose-clauses, i.e. "*Чтобы* + inf" (*To* + inf). The motivation for preferring partial clauses is that very often an appropriate nominalization noun lexeme for a process is lacking.

2.2.2 Lexico-grammatical realisation of deontic modality

Another aspect of possible stylistic variation in the final prototype texts concerns the realisation of deontic modals. So far, we have not included any explicit modal elements in the generated texts. As an alternative, the possibility or necessity of certain processes can be made explicit. For example, a precondition can be realised with an explicit necessity modal:

Cz:

6. Personal style using 2nd person plural:
   Nejdříve musíte otevřít dialogový panel Multiline Styles
   First must-2pl open-inf dialogue box Multiline Styles
   First you must open the Multiline Styles dialogue box.
7. Impersonal style using reflexive passive:
   Nejdříve se musí otevřít dialogový panel Multiline Styles
   First refl must-2pl open-inf dialogue box Multiline Styles
   First the dialogue boc Multiline Styles needs to be open.
8. Style-independent realisation:
   Nejdříve je nutné otevřít dialogový panel Multiline Styles
   First is necessary open-inf dialogue box Multiline Styles
   It is first necessary to open the dialogue boc Multiline Styles.

Ru:

9. Personal style:
   Сначала Вы должны открыть диалоговое окно Multiline Styles
   First you-pl must-pl open-inf dialogue box-acc Multiline Styles
   First you must open the Multiline Styles dialogue box.

   Сначала Вам необходимо открыть диалоговое окно Multiline Styles
   First you-pl-dat necessary open-inf dialogue box-acc Multiline Styles
   First you must open the Multiline Styles dialogue box.
10. Impersonal style:
    Сначала необходимо открыть диалоговое окно Multiline Styles
    First necessary open-inf dialogue box-acc Multiline Styles
    First the dialogue box Multiline Styles needs to be open.

Bu:

11. Personal style:
    Отначало трябва да отворите диалоговия прозорец Multiline Styles.
    First must to open-2sg dialogue-def box Multiline Styles.
    First you must open the dialogue box Multiline Styles.
12. Impersonal style:
    Отначало трябва да се отвори диалоговия прозорец Multiline Styles.
    First must to refl open-3sg dialogue-def box Multiline Styles.
    First the dialogue box Multiline Styles must be opened.

While for example in French, it is possible to use something like *You necessarily open the Multiline Styles dialog box first*, such realisation is not natural in any of the three Slavic languages, and neither is it in English.

When explicit modals are used in a procedural text, the explicit modal element should not be repeated in every instruction, but rather, it should appear only in the first sentence. It seems natural that when a Goal is expressed, and then a list of steps, only the Goal would be realised with an explicit modal element.

As noted above, when a list of alternative way of achieving a Goal is presented, it is possible and natural to use an explicit possibility-modal. In 0, we repeat the Russian example shown as 0 above, and provide the corresponding Czech and Bulgarian versions:

13. Ru:
    Вы можете выбрать один из следующих способов.
    You can choose one of following methods
14. *Cz*:
    Můžete zvolit jeden z následujících způsobů.
    Can-ind-2pl choose-inf one-acc of following-gen methods-gen
15. *Bu*:
    Може да изберете един от следните методи.
    Can to choose-2pl one of following-def methods.

    You can choose one of the following methods.

Again, personal vs. impersonal variation in style is possible.

It should be noted that in the corpora we have analysed, explicit modals were not used in (enumerated lists of) step-by-step procedural instructions at all. We found them only in overviews (introductions to sections), and the functionally-oriented texts (descriptions). The following two examples illustrate the explicit expression of possibility in an overview-style and a description-style sentence, respectively:

Overview:

16. Cz:
    V programu AutoCAD můžete určovat vlastnosti čar.
    In program AutoCAD can-ind-2pl specify-inf properties lines-gen.
17. Bu:
    В програмата AutoCad може да определите характеристики на линия.
    In Program-def AutoCad may to specify-2pl properties of line.
18. Ru:
    В программе AutoCAD можно задать свойства линий.
    In program AutoCAD possible specify-inf properties-acc lines-gen.
    In the AutoCAD program you can specify line properties..

Description:

19. Cz:
    Příkaz Multiline Styles umožňuje vytvářet styly čar.
    Command Multiline Styles enables create-inf styles-acc lines-gen
20. Bu:
    Командата Multiline Styles ви позволява да създадете стил на линия.
    Command-def Multiline Styles you enable-3sg to create-2pl style of line.
21. Ru:
    Команда Multiline Styles позволяет создавать стили линий
    Command Multiline Styles enables create-inf styles-acc lines-gen
    The Multiline Styles command enables you to create line styles.

2.2.3 Explicitness of content realisation

We have discussed the variation concerning explicit vs. implicit realisation of Side-effects in detail in TEXS2. We shall preserve this style variation in the final prototype.

In addition, we are going to support various degrees of full vs. abbreviated content expression. This is reflected especially in the text types of abbreviated procedural instructions, overviews and tables of contents. In all these cases, it is possible to vary the level at which explicit realisation stops. For a more elaborate discussion see the respective chapters.

2.2.4 Linear organisation

The A-box has a hierarchical structure, and it also contains lists of elements at the same level, e.g. alternative Methods within a Procedure, or sequence of Substeps within a Method-list.

Any list we are handling is ordered. While for alternative Methods, neither their order in the A-box nor the order in which they get realised really matters, the situation is different with lists of Substeps. We assume that the author who builds the A-box orders the Substeps in accordance to the order in which they need to be carried out by the user. Therefore, when the Substeps are realised as step-by-step instructions, their order should be obeyed. If their order is not reflected by the order in which they are realised, the temporal order should be indicated by explicit temporal discourse markers. We think that it is most natural for procedural texts to obey the ordering of Substeps as given in the A-boxes, and therefore we do so in the final prototype texts.

A different case is at hand with Goals and the (list of) Substeps within the corresponding Method(s). These are in a hierarchical relation in the A-box, and it is a matter of text planning to decide in what order their contents should be conveyed.

The issue of alternative ordering of Goal and Method, and therefore the corresponding Substeps, has not yet been addressed within the Agile project. There is a natural link to the issue of aggregation (next section) here, because the issue of Goal and Method(s) ordering only arises when the content is realised by a clause complex. When the content of the Goal is realised in a separate sentence, than it is natural to order this sentence before the realisation of the Method(s).

The A-box specifies the relation between a Goal and Method(s), and the corresponding Substeps, but it is a matter of text planning to chose a realisation of this relation by an RST-relation, i.e. either RST-Means or RST-Purpose (discussed in the next section). The current text planner supports one realisation only, either RST-Means, with Goal^Method, or RST-Purpose, with Method^Goal. The choice is made in the SPLIZE module.

With respect to linear ordering, all four possibilities are allowed in general:

- RST-Means : Goal^Method(s): *Specify the start point by entering endp*
- RST-Purpose: Method(s)^Goal: *Enter endp to specify the start point*
- RST-Means: Goal^Method(s): *By entering endp specify the start point*
- RST-Purpose: Method(s)^Goal: *To specify the start point enter endp*

2.2.5 Discourse aggregation and syntactic aggregation

There are various ways in which we can combine content, as expressed by PROCEDUREs in the A-box and identified by TASKs in the text plan. We already pointed out that we consider here discourse aggregation and syntactic aggregation. Given two PROCEDUREs specifying content, and the corresponding linguistic structures that would express that content, discourse aggregation concerns the combination of these two linguistic structures into a single linguistic structure that makes explicit the rhetorical relation between two individual linguistic structures, thus supporting the overall rhetorical structure of the text.

In the intermediate prototype we handled aggregation using either RST-MEANS or RST-PURPOSE. We did not provide any flexibility with respect to enabling various kinds of RST relations (also beyond MEANS or PURPOSE), nor could we handle syntactic aggregation like coordinating linguistic structures that correspond to subsequent PROCEDUREs in SUBSTEPS, or disjunction of alternative METHODs listed under a PROCEDURE's METHODS field. In the final prototype, we would like to support stylistic variation along these lines. One area to be covered is clause-complexity in realising a goal and its method(s) by RST-Means or RST-Purpose, as hinted at above. Here we provide an example using RST-Purpose.

22. Cz:
    Zvolte Element Properties, abyste přidali elementy ke stylu.
    Choose-imp-2pl Element Properties so-that-would-2pl add-2pl elements to style.
23. Bu:
    Изберете Element Properties, за да прибавите елементи към стила.
    Choose-2pl Element Properties in-order to add-2pl elements to style-def.
24. Ru:
    Выберите пункт Element Properties,
    Choose-imp-pl item-acc Element Properties,

    чтобы добавить элементы в стиль
    in-order-to add-inf elements-acc.in style-acc.

    Choose Element Properties to add elements to style.

Alternative Methods are naturally realised by Disjunction:

25. Cz:
    Zvolte Element Properties nebo stiskněte Ctrl-E.
    Choose-imp-2pl Element Properties or press-imp-2pl Ctrl-E.
26. Bu:
    Изберете Element Properties или натиснете Ctrl-E.
    Choose-2pl Element Properties or press-2pl Ctrl-E.
27. Ru:
    Выберите пункт Element Properties или нажмите Ctrl-E.
    Choose-imp-pl item-acc Element Properties or press-imp-pl Ctrl-E.

    Choose Element Properties or press Ctrl-E.

There are also of various ways of grouping content within an A-box, and using Conjunction. For example, Substeps within one Method can be conveniently conjoined:

28. Cz:
    Zadejte název souboru a zvolte jeho typ.
    Enter-imp-2pl name file-gen and choose-imp-2pl its type.
29. Bu:
    Въведете име на файл и изберете неговия тип.
    Enter-2pl name of file and choose-2pl its type.
30. Ru:
    Введите имя файла и выберите его тип
    Enter-imp-pl name-acc file-gen and choose-imp-pl its type.

    Enter the name of the file and choose its type.

Similar aggregation options as we have exemplified for instructions in procedural style are available for the realisation of descriptive texts and overviews. The following two examples illustrate aggregation by Conjunction in an overview-style and a description-style sentence, respectively:

Overview:

31. Cz:
    V programu AutoCAD můžete určovat vlastnosti čar
    In program AutoCAD can-ind-2pl specify-inf properties lines-gen

    a ukládat je jako styly.
    and save them as styles.
32. Bu:
    В програмата AutoCad можете да определите свойства на линия
    In program-def AutoCad may-2pl to specify-2pl properties of line

    и да ги запомните като стил.
    and to them save-2pl as style.

33. Ru:
В программе AutoCAD можно задать свойства линий и
In program AutoCAD possible specify-inf properties-acc lines-gen and

сохранить их в виде стилей
save-inf them in form styles-gen

In the AutoCAD program you can specify line properties and save them as styles.

Description:

34. Cz:
Příkaz Multiline Styles umožňuje vytvářet styly čar
Command Multiline Styles enables create-inf styles-acc lines-gen

a ukládat je jako styly.
and save them as styles.

35. Bu:
Командата Multiline Styles Ви позволява да създадете стилове на
Command-def Multiline Styles you enable-2pl to create-2pl styles of

линията и да ги запомните като стилове.
line-def and to them save-2pl as styles.

36. Ru:
Команда Multiline Styles позволяет создавать стили линий и
Command Multiline Styles enables create-inf styles-acc lines-gen and

сохранить их в виде стилей
save-inf them in form styles-gen

The Multiline Styles command enables you to create line styles and save them as styles.

Another case where aggregation can be applied in the realisation of the A-box content is the expression of Constraints. In the intermediate prototype system, Constraints are expressed in separate content-units, realised by a nominal group. A Constraint reflects a situation in which a particular Method can be used to accomplish a given Goal. Therefore, a Constraint could also be expressed either by a subordinate conditional clause modifying the main clause realising the Goal, or by an element within the Goal-clause.

2.2.6 Explicit marking of sequences by discourse markers

As we noted above, a sequence of Substeps within the same Method in an A-box is assumed to be ordered in accordance to the order of executing the Substeps to achieve the given Goal. We also noted that it is natural for the generated text to obey this order when the Substeps are realised. This is what the current text planner does, and also in the final prototype, the A-box ordering of Substeps will be obeyed in the realised texts.

In addition, is it possible to include explicit discourse markers reflecting the sequences of Substeps, e.g. *now, then, first, second*, *finally,* etc. (and their equivalents in Bulgarian, Czech and Russian). This may help the reader to orient herself, especially in the cases

where a hierarchical content is realised by a series of instructions. The explicit discourse markers thus reflect discourse structure (see Chapter 3.2 for examples and an elaborate discussion).

2.2.7 Numbered lists vs. running texts

Instructions in procedural style, but possibly also other text types, can express the content either in numbered lists, or in running text. Running text is typically used for overviews.

While numbering certainly is a matter of output formatting, it is more than just that. Issues of aggregation and explicit discourse markers are intertwined with the choice of numbered list vs. running text realisation.

So for example, whereas in a numbered list, explicit discourse markers signalling sequences would be redundant, such is needed to a greater extent in a running text. Running text often also exhibits more aggregation.

2.3 Correlations between text type and stylistic variation

The following table relates text types and stylistic variations. We indicate which parameters of style are fixed within a text type, and which stylistic variations are meaningful or useful.

Text Type	Personal vs. impersonal style	Explicit modality	Explicit side-effects	Ordering of Goal and Method(s)	Numbered list vs. Running text	Aggregation	Explicit discourse markers
Full procedural	Yes	yes	yes	Yes	yes	yes	yes
Goal summary	Yes	yes	not appl.	not appl	only num.	only conj.	not appl.
Basic steps	Yes	yes	yes	not appl.	only num.	conj and disj.	yes (for clustering)
Functional descriptions	Yes	yes	not appl.	yes	only running	yes	yes
Overviews	Yes	yes	not appl.	yes	only running	yes	yes
TOC	Not appl.	not appl.	not appl.	not appl	only num.	not appl.	not appl.

Figure 1: Correlations between text types and stylistic variations

The correlations captured in this table will be further discussed in the subsequent chapters. They will also be reflected in the formalisations, which will provide the necessary distinctions underlying the variety available in the content realisation.

3. Full procedural instructions

Clauses, sentences and also larger pieces of discourse are not isolated from one another. In a coherent discourse, there are relations between "discourse units". They go under various names, for instance *coherence relations* (Hobbs 1979), *discourse relations* (Asher 1993), further distinguished as *subject-matter* vs. *presentational* (RST, Mann and Thompson 1988), *informational (semantic)* vs. *intentional* (Moore and Pollack 1992).

Some of these relations (esp. the subject-matter or informational ones) reflect relations that hold between the objects the discourse is about, or between eventualities described in the discourse. The relations between various pieces of content as captured in an A-box can be classified in an abstract and general way as follows:

- SEQUENCE relation(s) among SUBSTEPS within a METHOD
- ALTERNATIVE relation(s) among METHODs within a PROCEDURE (if there are more than one)
- SUBORDINATION, or EMBEDDING of some piece of content with respect to another
- ENABLEMENT or PURPOSE relation(s) between the GOAL and the METHOD(s) within a PROCEDURE's METHODs (a special case of EMBEDDING)

Due to these inherent relations between the underlying pieces of content and/or due to the speaker's communicative GOALs, relations between discourse units are always present. However, sometimes they are signalled explicitly, and sometimes they are not. One can speculate that some contexts may be sufficiently specific, so that no explicit signalling is needed. It is also possible to find relations between pieces of content, which do not have a straightforward corresponding realisation, and it is for this reason that they are not explicitly signalled (Knott 1996).

On the other hand, there are linguistic means of signalling relations explicitly which can easily and efficiently convey more than just the type of relation at hand, i.e. a causal relation. This additional meaning gets in through presuppositions associated with certain discourse connectives (Lagerwerf 1998, Stone and Webber 1998, Webber et al. 1999a, Webber et al. 1999b). However interesting this topic is, also from the NLG perspective, it appears outside the AGILE scope at the moment. We concentrate on the assertion-part of the meaning of discourse connectives, leaving the presupposition-part aside.

Explicit signals of relations between discourse units can take various forms. In written procedural instructions, which is what we focus on in this chapter, the following means are available, and exploited:

- linguistic expressions going under the names of *discourse markers* (Schiffrin 1987), *discourse connectives* (Webber, *ibid*), *clausal connectives* (Knott and Mellish 1996), *discourse cues* (Grosz and Sidner 1986), *cue phrases* (Knott 1996), and *clue phrases* (Cohen 1987)
- list numbering
- layout, i.e. mostly line-breaks between discourse units and indentation

These means naturally interplay. In the procedural instructions in our AutoCAD corpus, we have encountered the following ways of exploiting the means of realisation with respect to the above mentioned relations between pieces of content in the A-box:

- SEQUENCE relation among SUBSTEPS within a METHOD is primarily reflected by the ordering of sentences and clauses that realise the SUBSTEPS. In addition, the realisations of the SUBSTEPS are organised into lists with a common layout. The lists are mostly numbered. Sometimes there are explicit discourse markers.
- ALTERNATIVE relation among METHODs within a PROCEDURE is most often encountered with the alternative ways of performing a task depending on the used operating system. The alternatives can be distinguished by a Constraint. In this case, a particular layout is used in the AutoCAD manual: the Constraints are realised by nominal groups and typeset in bold face; the groups of instructions pertaining to the same Constraints are realised as running text. On the other hand, for METHODs not distinguished by a Constraint, the realisations are usually aggregated by means of Disjunction.
- EMBEDDING of some piece of content with respect to another (for instance, SUBSTEPS of a PROCEDURE's METHODs embedded within a SUBSTEPS within a higher PROCEDURE's METHODS), is very often not reflected in the realisation in the AutoCAD manual. This means that SUBSTEPS are realised in a continuing numbered list irrespective of the (presumed) underlying hierarchical structure of the task as a whole. In our opinion, this nivelisation of the underlying hierarchical structure decreases clarity of the instruction text.
  There are a few cases in the manual where the hierarchical structure is reflected by layout (indentation of a sub-list) and/or by explicit discourse markers (e.g., *Now save the style* in IMD-Text4).
- ENAMBLEMENT or PURPOSE relation between the GOAL and a METHOD within a PROCEDURE is in some cases reflected by aggregation by means of a clause complex involving RST-Means or RST-Purpose. This is mostly when the METHOD is rather simple with respect to the SUBSTEPS it requires. When the METHOD is complex, the GOAL and the METHOD are realised by separate sentences. It is then very often the case that the EMBEDDING is disregarded in the realisation exactly as described above. So, the GOAL and the SUBSTEPS of the corresponding METHOD end up in one and the same numbered list. Again, our opinion is that this nivelisation makes understanding the instructions more difficult.

This summary shows that the interplay between the kinds of relations between pieces of content and the means of their realisation is quite complex. We have encountered some regularity, for instance the use of numbered lists for a SEQUENCE of SUBSTEPS, and the use of aggregation or indentation to reflect EMBEDDING. It appears natural to use a vertical list layout for alternative METHODs distinguished by the operating system. Also, the top-level SUBSTEPS are naturally realised in a vertical list. These appear to be typical for the given genre, that is, for procedural instructions.

We have also encountered cases where relations between pieces of content are disregarded (not reflected) in the text. This may be a matter of stylistic choice within the genre. An example of this is the neglect for hierarchical structure of the task. In our opinion, such realisations are uncooperative towards the reader, because they obscure the inherent structure of the task.

In this chapter, we focus on the realisations of the SEQUENCE, ALTERNATIVE and EMBEDDING relations in procedural instructions. First in Section 3.1 we discuss aggregation as a realisation of ALTERNATIVE and EMBEDDING relations. Then, in Section 3.2 we are

concerned with the realisation of the SEQUENCE relation by means other than straightforward ordering of discourse units.

3.1 Aggregation in full procedural instructions

In this section, we will discuss the explicit inclusion of discourse relations into the text plan. We begin by outlining the approach we take in section 0. There, we begin with explaining how the approach provides a generalisation of the way we handled discourse relations in text structuring for the intermediate prototype. The more general character of the approach provides a transparent way of how discourse relations are handled, for one making it easier to be extended to a broader coverage of possible text structures, and for another, facilitating a more direct comparison to existing theories on discourse structure.

Thereafter we discuss in section 3.1.2, two discourse relations that are naturally prevailing in the instructional texts we are dealing with, namely *conjunction* as a realisation of the SEQUENCE relation and *disjunction* as a realisation of the ALTERNATIVE relation. These two relations reflect two fundamental structures in our A-boxes, namely a METHOD's SUBSTEPS as a sequence of steps to be taken, and a PROCEDURE's METHODS as alternative ways of obtaining the PROCEDURE's GOAL.

In section 3.1.3 we discuss the introduction of (more abstract) discourse relations that reflect EMBEDDING, into the text plan. These are RST-MEANS or RST-PURPOSE. These discourse relations can be used to aggregate content, possibly in combination with conjunction, giving rise to more complex sentences. An interesting stylistic variation we consider here is the relative ordering of a discourse (RST) relation's satellite and nucleus. As previous corpus study already showed, the preference of one order over the other for example can be related to the (assumed) level of expertise of the intended readership. We describe how this can be accommodated in the approach.

Up to this point, we have been discussing relations between PROCEDURES, METHODS, or between a single PROCEDURE and one or more METHODS. In section 3.1.4 we discuss different levels of aggregation (or, to be more precise, granularity) including other kinds of configurational concepts as well. The example we consider is the inclusion of a CONSTRAINT in the sentential structure, instead of regarding it as a "label", as illustrated by the following:

If you are using **Windows,** choose Multiline Styles from the Object Properties toolbar. **Windows:** Choose Multiline Styles from the Object Properties toolbar.

3.1.1 The approach

In the TSM for the intermediate prototype (TEXM2) we also used discourse relations, but these were not explicitly included in the text plan. Rather, they were derived when interpreting the text plan for the sake of sentence planning. Although a valid and functioning approach, it seems more economic and more natural to infer discourse relations already at the stage of constructing the text plan, and include the discourse relations explicitly.

The reason for explicit inclusion being more economic is that the A-box content necessary to act upon for inferring discourse relations is already available (and actively inspected) when constructing the text plan. Explicit inclusion means simply taking that

extra step further, rather than deferring the decision for using a particular discourse relation to a later stage where we would then have to inspect the same content again. Furthermore, the text plans that we obtain this way are more closely related to existing theories on discourse structure (e.g. (Polanyi 1988), (Gardent 1994), (Webber 1998), (Lagerwerf 1998)).

Explicitly including discourse relations into the text plan should be understood in the following way. The TSM for the intermediate prototype constructs text plans by interpreting the structure of the A-box, as determined by the use of configurational concepts. The interpretation was based on an almost isomorphic mapping between configurational concepts and the constituents of a text plan, called text structure elements (again, cf. (TEXM2)). A simple example is the text plan in Figure 2.

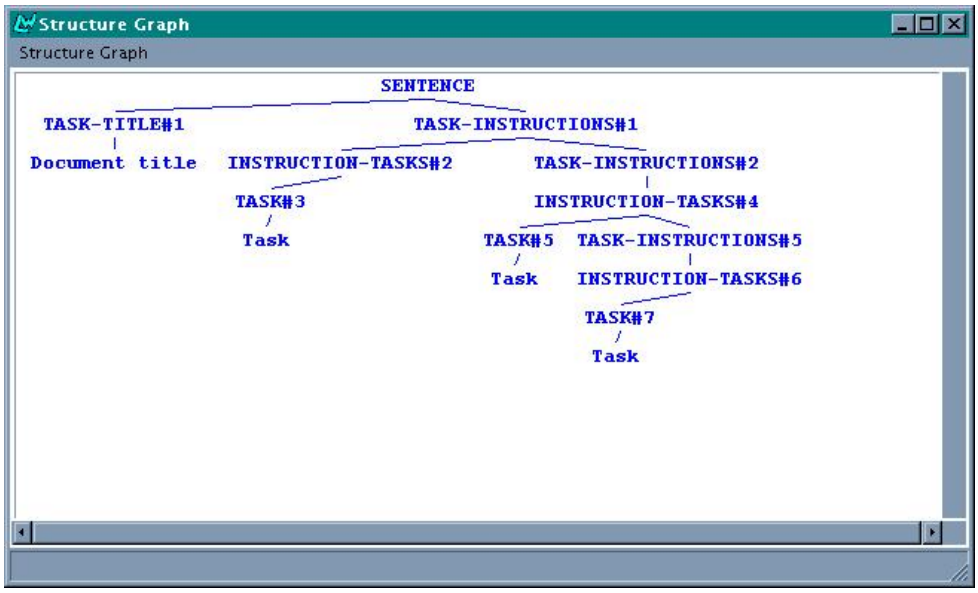


Figure 2: Intermediate prototype text plan

Instead of having multiple nodes like TASK-INSTRUCTIONS (corresponding to a PROCEDURE's METHODS) and INSTRUCTION-TASKS (corresponding to a METHOD's SUBSTEPS), we propose here to designate most of these nodes using more informative names indicating discourse relations.

The consequence is not just that the resulting text plan looks nicer. The decisions for including a particular discourse relation can now be determined by systems in a KPML region like the CADCAM-INSTRUCTIONS region (TEXM2). Therefore, extending the coverage becomes very similar to developing lexico-grammars - we are now, in a more concrete sense, building "text grammars". Furthermore, when planning sentences to realise the text plan, the sentence planner now only has to inspect the text plan to effectuate aggregation in a sentence plan rather than infer a possible discourse relation.

3.1.2 Conjunction and disjunction

Conjunction and disjunction are relations that follow naturally from two configurational concepts in the AGILE Domain Model. Namely, a PROCEDURE has a METHODS field (of type METHOD-LIST) that lists the alternative ways of obtaining the PROCEDURE's GOAL. This usually can be translated into disjunction, particularly when we are aiming at generating running text:

To save a document
Select the *Save* command from the *File* menu
Click on the *Save* icon in the toolbar

To save a document, select the *Save* command from the *File* menu or click on the *Save* icon in the toolbar.

Similarly, a METHOD that specifies one way to achieve a PROCEDURE's GOAL, has a SUBSTEPS field (of type PROCEDURE-LIST) that gives the PROCEDUREs or "steps" that have to be taken. In other words, the SUBSTEPS field specifies a sequence, which we conceive of as a form of conjunction.

To save a document under a different name
1. Select the *Save As* command from the *File* menu.
2. Enter the file name in the *File name* box.
3. Click the OK button.

To save a document under a different name, select the *Save* command from the *File* menu, enter the file name in the *File name* box, and click the OK button.

Let us first consider conjunction. Naturally we should discuss the conditions under which TASKs (being the text structure elements corresponding to PROCEDUREs) could indeed be considered proper to be conjoined (being eventually realised using a coordinated structure), and when they should be kept separate. By its definition in the Domain Model, a PROCEDURE has to specify a GOAL, and may specify a SIDE-EFFECT and/or METHODS. In the following example, the first step has a side-effect and the second step has alternatives:

To save a document under a different name
1. Select the *Save As* command from the *File* menu. The *Save As …* dialogue box appears.
2. Provide a file name by entering the file name in the *File name* box or by selecting a file name from the *Directory list*.
3. Click the OK button.

In this case we do not want to conjoin the steps, for this would sound very unnatural. Instead, we should simply state them separately:

To save a document under a different name, select the *Save As* command from the *File* menu. The *Save As …* dialogue box appears. Provide a file name by entering the file name in the *File name* box or by selecting a file name from the *Directory list*. Click the OK button.

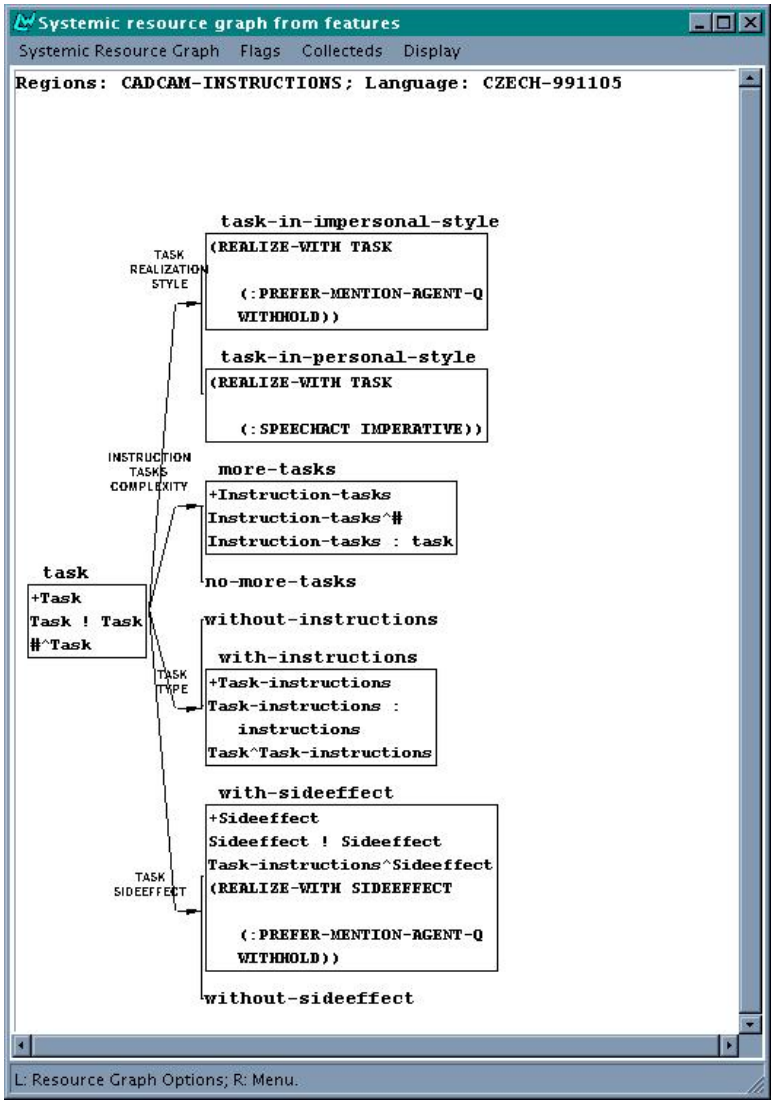


Figure 3: Systemic region graph focussing on TASK

Hence, we suggest here to introduce a conjunction into the text plan when we have two TASKs that are *simplex*, i.e. which do not specify side-effects nor METHODs. When we construct the part of a text plan corresponding to a METHOD's SUBSTEPS, then above each step (a TASK) we have a node that is either labelled "CONJOINED-INSTRUCTION-TASKS" to indicate conjunction or "SEPARATE-INSTRUCTION-TASKS" to signify that the TASK under that node should not be conjoined to the preceding TASK(s).[1]

[1] INSTRUCTION-TASKS is a text structure element that corresponds to the type of structure that a METHOD's SUBSTEPS field contains, namely a PROCEDURE-LIST. A PROCEDURE-LIST implements a list structure in a traditional way, by having a head that is of type PROCEDURE, and a tail that is of type PROCEDURE-LIST. For that reason the text plan contains a whole branch of nonterminal nodes called *-INSTRUCTION-TASKS rather than just one *-INSTRUCTION-TASKS with all its elements (TASKs) as its immediate daughters. We make convenient use here of having several *-INSTRUCTION-TASKS nodes (one for each TASK) to express conjoinability of individual TASKs.

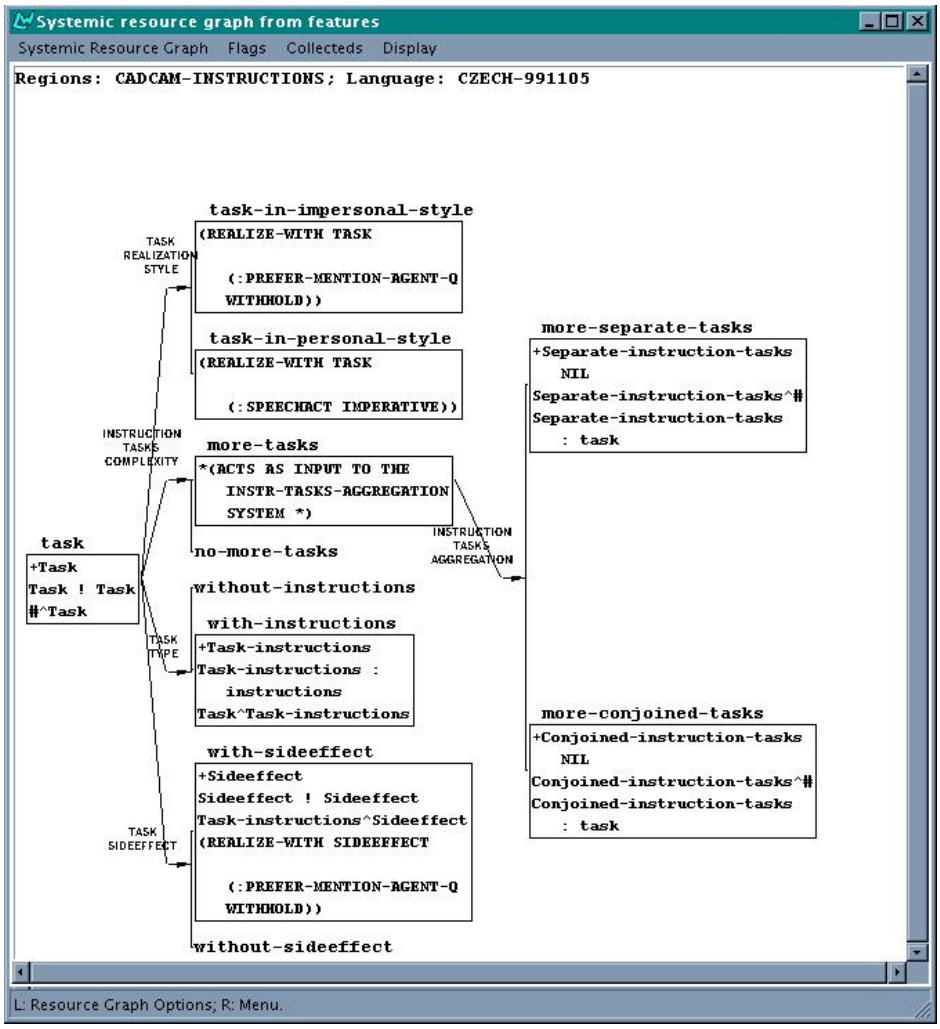


Figure 4: INSTRUCTION-TASKS-AGGREGATION system

Formally, we can specify this as follows. To begin with, we are dealing here with TASKS and their planning as members of an INSTRUCTION-TASKS list. The systems in the CADCAM-INSTRUCTIONS region that are responsible for that planning are given in Figure 3.

Instead of immediately introducing an INSTRUCTION-TASKS node in the system INSTRUCTION-TASK-COMPLEXITY we can decide to introduce only the abstract feature *More-Tasks* and let that feature be an entry condition to another system INSTRUCTION-TASKS-AGGREGATION where we decide about conjoinability.

The decision for introducing a node SEPARATE-INSTRUCTION-TASKS or CONJOINED-INSTRUCTIONS-TASKS depends whether the TASK that we will be introducing under this node will be simplex or not, as we discussed above.

System:

```
More-tasks →
(more-conjoined-tasks
      insert CONJOINED-INSTRUCTION-TASKS
      preselect CONJOINED-INSTRUCTION-TASKS TASK
      orderatend CONJOINED-INSTRUCTION-TASKS
)
(more-separated-tasks
      insert SEPARATE-INSTRUCTION-TASKS
      preselect SEPARATE-INSTRUCTION-TASKS TASK
      orderatend SEPARATE-INSTRUCTION-TASKS
)
```

Chooser:

```
Choose more-conjoined-tasks if conjoinable
else choose more-separated-tasks
```

The chooser itself relays the question about the conjoinability of the TASK to be inserted to an inquiry that looks in the A-box whether the PROCEDURE, to which the TASK is to correspond, specifies a SIDE-EFFECT and/or METHODS. If the PROCEDURE has neither a SIDE-EFFECT nor METHODS then we consider the PROCEDURE (and the TASK) simplex:

Inquiry:

```
Consider root to be the PROCEDURE to be considered next
if (get-abox-slot dm::side-effect)
      or (get-abox-slot dm::METHODs)
then return NIL   ;; not simplex
else return TRUE  ;; simplex, no side-effect nor METHODs
```

By considering each PROCEDURE (each TASK) on its own, rather than the entire list at once, we can obtain a heterogeneous structure in which conjunction and separation are interleaved. In the next example, the side-effect separates the first step from the final two steps. The latter are again conjoined:

> To save a document under a different name, select the *Save As* command from the *File* menu. The *Save As …* dialogue box appears. Provide a file name by entering the file name in the *File name* box and click the OK button.

We can take a similar approach for disjunction. As we pointed out above, the possibility for disjunction arises when a PROCEDURE specifies several METHODs as members of its METHODS' METHOD-LIST. Each of these METHODs present an alternative way of obtaining the PROCEDURE's GOAL. However, like in the case of conjunction, we should be more specific about when disjunction would indeed be natural, and when we should keep things separated.

By its definition in the Domain Model a METHOD must have a non-empty SUBSTEPS field, and may specify a CONSTRAINT and/or a PRECONDITION. A CONSTRAINT is of type OPERATING-SYSTEM, like "Windows" in the examples above. A

PRECONDITION is a PROCEDURE that should be executed before the METHOD's SUBSTEPS can be carried out.

Again, we propose a simplicity measure. We should only insert a disjunction (realised later as an "or") between METHODs if each METHOD is simplex, i.e. it does not specify a CONSTRAINT nor a PRECONDITION. However, a simplicity measure in the case of disjunction is a bit more complex than it was for conjunction. The uncomplicated situation is that the *entire* list consists of METHODs none of which specify a CONSTRAINT or PRECONDITION. In that case we can straightforwardly allow for disjunction.

However, the situation becomes more complicated in case no METHODs in the list specify a PRECONDITION but all METHODs *do* specify a CONSTRAINT. In that case we should keep separated METHODs that have different CONSTRAINTs, but we can put a disjunction between METHODs that follow upon one another and which have identical CONSTRAINTs. For example, the simple text below could be specified using one PROCEDURE and three METHODs. Two of these METHODs would specify alternative ways of saving a document under a different name when using Windows, whereas the third METHOD would specify the step needed to achieve that same GOAL under Dos.

To save a document under a different name:
Windows: Select *Save As …* from the *File* menu or click the *Save As...* icon on the toolbar
Dos: Select *Save As...* from the *File* menu.

We provide the following formal specification for deciding to include disjunction into the text plan. To begin with, what we are dealing with METHODs (and their SUBSTEPS) that are specified as members of a METHOD-LIST. In text planning a METHOD corresponds to an INSTRUCTION, and the METHOD-LIST being the value of the PROCEDURE's METHODS field corresponds to TASK-INSTRUCTIONS in a way similar to the INSTRUCTION-TASKS construction above. The relevant systems are shown in Figure 5.

Again, instead of immediately introducing a TASK-INSTRUCTIONS node in the system TASK-INSTRUCTIONS-COMPLEXITY we introduce only the abstract feature *More-Instructions* and let that feature be an entry condition to an other system TASK-INSTRUCTIONS-AGGREGATION where we decide about disjoinability.

```
System:
    More-Instructions →
    (more-disjoined-instructions
          insert DISJOINED-TASK-INSTRUCTIONS
          preselect DISJOINED-TASK-INSTRUCTIONS INSTRUCTIONS
          orderatend DISJOINED-TASK-INSTRUCTIONS
    )
    (more-separate-instructions
          insert SEPARATE-TASK-INSTRUCTIONS
          preselect SEPARATE-TASK-INSTRUCTIONS INSTRUCTIONS
          orderatend SEPARATE-TASK-INSTRUCTIONS
    )
Chooser:
    Choose more-disjoined-instructions if disjoinable
    else choose more-separated-instructions
```

Inquiry:

```
      ;; if we are at the beginning of a new METHOD-LIST then
;; current-constraint must be reset to nil.

Consider the root to be the METHOD to be considered next
if the entire list contains METHODS that are simplex
then
      return TRUE ;; can insert disjunction
else
      if (equal (get-abox-slot DM::constraint)
            (current-constraint))
      then  return TRUE
      else  current-constraint = (get-abox-slot DM::constraint)
            return NIL
```

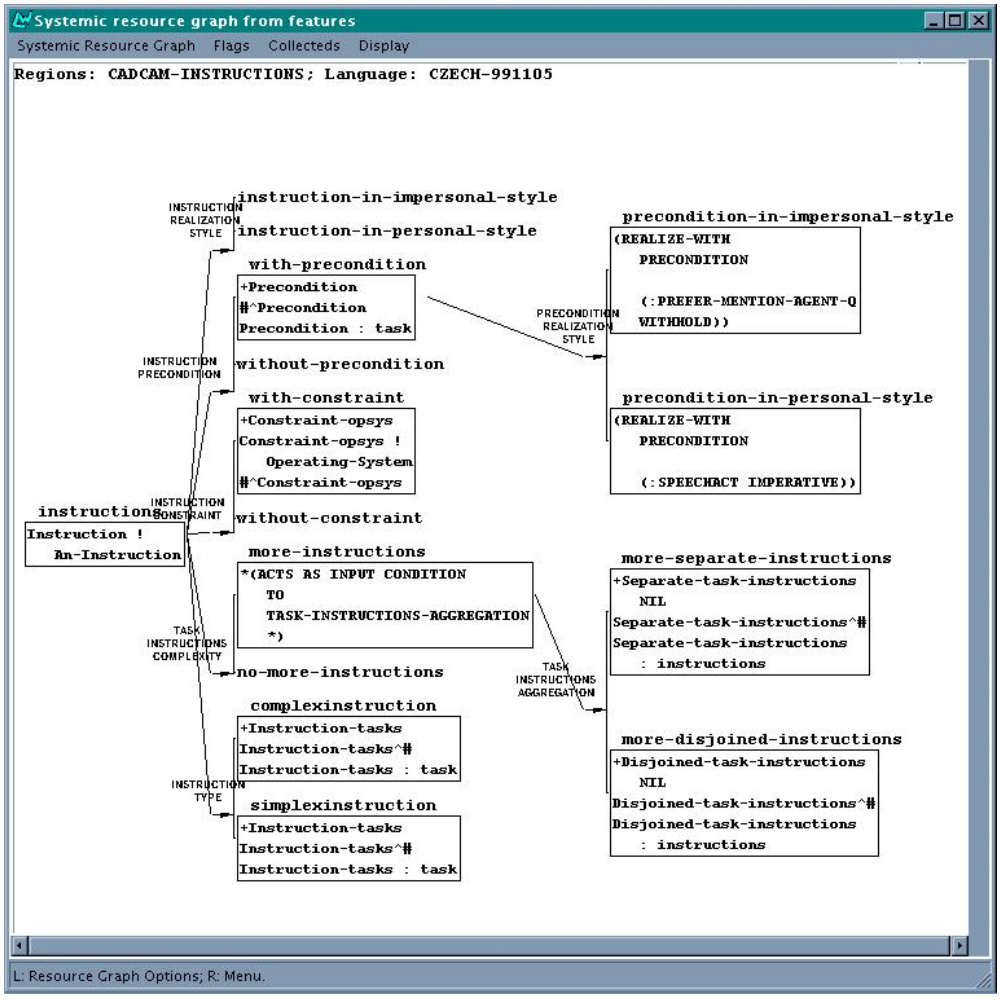


Figure 5: Systemic region focusing on INSTRUCTION

Because disjunction and conjunction act on different text structure elements, namely TASK-INSTRUCTIONS respectively INSTRUCTION-TASKS, we can obtain conjunctions within disjunctions (since each METHOD defined SUBSTEPS that introduce one or more *-INSTRUCTION-TASKS nodes). For example:

To save a document:
Windows: Open the *File* menu and select the *Save* command or click the *Save* icon on the toolbar.

Concluding this section, we should make some final remarks. To begin with, the inquiries underlying our decisions whether to introduce conjunction or disjunction are oblivious to any restrictions on the number of elements that could be combined this way. If we want to impose such a limit, we could do so by for example using a counter keeping track of the elements already conjoined or disjoined.

Furthermore, when planning sentences, the sentence planner should inspect the text plan to determine whether a sentence should be planned including a coordinated structure.

Finally, because we extend the systems dealing with the complexity of lists rather than INSTRUCTION-TYPE or TASK-TYPE, the first node in the text plan corresponding to list structure will never indicate disjunction or conjunction.

This last point is important for the discussion in the next section. There we consider the introduction of nodes into the text plan which indicate discourse relations like RST-MEANS or RST-PURPOSE. The approach builds forth on our discussion above --in particular, the first node corresponding to a METHOD-LIST (i.e., the first TASK-INSTRUCTIONS node) will come to signify whether a discourse relation should be used later in aggregation, and if so which relation.

3.1.3 Aggregation using discourse relations

Conjunction and disjunction hold between text structure elements that are of the same type, namely TASKs respectively INSTRUCTIONs. The discourse relations RST-PURPOSE and RST-MEANS are between text structure elements that are of *different* types. Both discourse relations can be used, under certain restrictions, to couple a TASK to an INSTRUCTION.

3.1.3.1 Discourse relations in the text plan

In the text plan, a TASK is related to an INSTRUCTION by means of a TASK-INSTRUCTIONS node, as shown in Figure 2 repeated here as Figure 6.

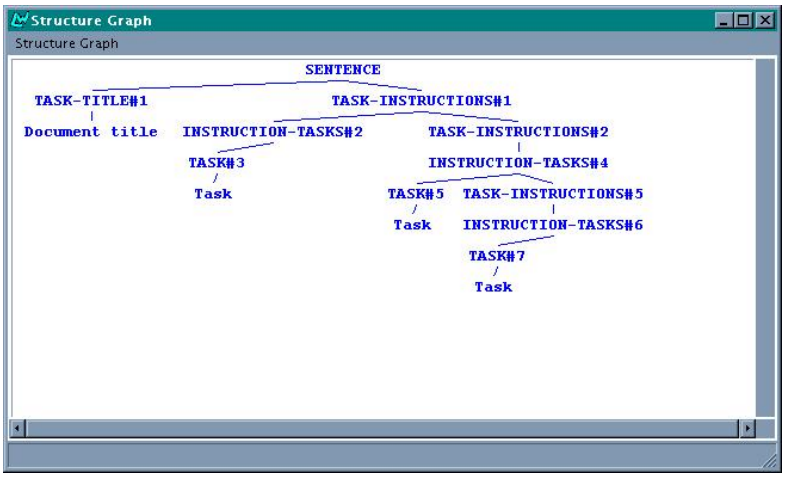


Figure 6: TASK, TASK-INSTRUCTIONS, INSTRUCTION-TASKS

The TASK-INSTRUCTIONS node corresponds to the METHOD-LIST that fills the METHODS slot of the PROCEDURE to which the TASK corresponds. We will use this node to signify explictly the discourse relation that is to hold between the TASK and an INSTRUCTION.[2] We already saw in Figure 3 (repeated here as Figure 7) that the first TASK-INSTRUCTIONS would get inserted in the TASK-TYPE system, depending on whether the underlying PROCEDURE has a filled METHODS field or not.

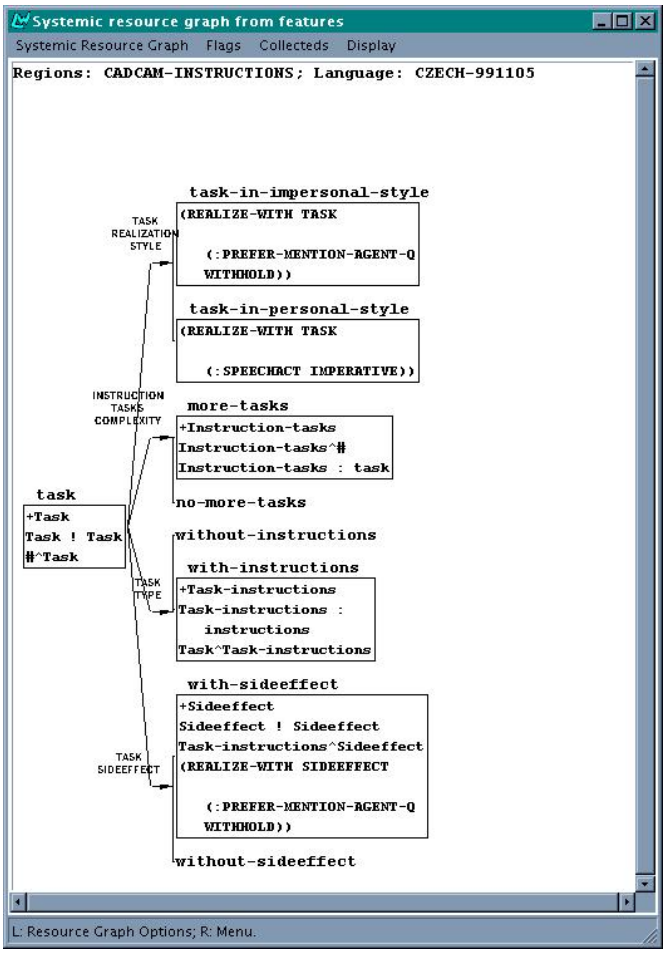


Figure 7: Systemic region graph focusing on TASK

Instead of directly inserting a TASK-INSTRUCTIONS node in the TASK-TYPE system, we propose to apply a similar construction here as we employed above, namely deferring the decision for insertion of a particular type of TASK-INSTRUCTIONS node to a different system to which the feature *with-instructions* acts as entry condition.

Hence, the TASK-TYPE system should first of all be altered into the following:

[2] More specifically, the TASK-INSTRUCTIONS node that connects TASK to its INSTRUCTION(s) corresponds to the first METHOD in the METHOD-LIST. As we discussed in the previous section, subsequent METHODs in that list will be placed under nodes that signify either disjunction or separation. The very first TASK-INSTRUCTIONS node obviously does not signify any such relation.

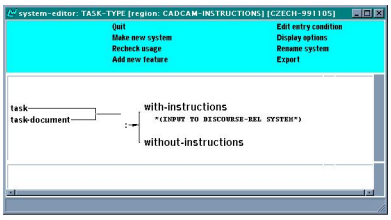


Figure 8: Modified TASK-TYPE system

Subsequently, we specify the system TASK-INSTRUCTION-DISCOURSE-RELATIONS where we decide whether a TASK and its INSTRUCTION(s) should be explicitly related through a discourse relation, and if so, which relation. For each discourse relation we want to be able to distinguish, there is a choice in the system. Each discourse relation gets signified in the text plan by a node bearing its name, for example TI-RST-MANNER or TI-RST-PURPOSE (with "TI" indicating aggregation of a TASK and its INSTRUCTION(s)).

The decision to aggregate, and using what relation, depends on a number of factors: for example, what type of full text we are going to generate (full text or using enumeration), whether the TASK has one or more INSTRUCTIONs, what readership the text is intended for.

For example, consider the following two text fragments. One fragment is formulated using enumeration, the other fragment is running text.

Enumeration:

Add elements to the style.

1. Choose Element Properties. The *Element Properties* dialogue box appears.
2. Enter the offset of the multiline element in the *Element Properties* dialogue box and select Add to add the element.

Running text:

Choose Element Properties to add elements to the style. The *Element Properties* dialogue box appears. Enter the offset of the multiline element in the *Element Properties* dialogue box and select Add to add the element.

In the case of running text there is no problem to aggregate the overall GOAL with the first step to achieve that GOAL. If we would do the same in the case of enumeration, this would look unnatural.

The system can be formally specified as follows:

```
System: [TASK-INSTRUCTIONS-DISCOURSE-RELATIONS]
    with-instructions →
    (ti-separate
          insert TASK-INSTRUCTIONS
          preselect TASK-INSTRUCTIONS INSTRUCTIONS
          orderatend TASK-INSTRUCTIONS
    )
    (ti-means
          insert TI-RST-MEANS
          preselect TI-RST-MEANS INSTRUCTIONS
          orderatend TI-RST-MEANS
```

```
)
(ti-purpose
      insert TI-RST-PURPOSE
      preselect TI-RST-PURPOSE INSTRUCTIONS
      orderatend TI-RST-PURPOSE
)
```

Because the discourse relations are indicated apart from (specifically, above) conjunction and disjunction, we are able to plan an aggregation between a TASK and a conjunction of the SUBSTEPS that the TASK's INSTRUCTION specifies. This could for example lead to the following running text:

To save a document, open the *File* menu and choose the *Save* command or click the *Save* icon in the toolbar.

Once we arrive to sentence planning, aggregation of clauses (realising bits of content) into more complex sentences should be done by the sentence planner inspecting the text plan for information on what to do. Of interest in the context of this deliverable is the *order* in which a discourse relation's nucleus (TASK) and satellite (one or more INSTRUCTIONs) should be planned to occur in a sentence. We turn to this issue in the next section.

3.1.3.2 Relative ordering of nucleus and satellite in discourse relations

Each discourse relation, like RST-MEANS or RST-PURPOSE, has a nucleus and a satellite. The nucleus is said to govern the satellite, with the satellite being dependent on the nucleus in the sense as specified by the particular discourse relation involved. Intuitively, one might say that the relation explains the way a satellite contributes to the meaning of the overall construction.

Prototypically, in an RST-MEANS relation the nucleus precedes the satellite, whereas in RST-PURPOSE the situation is the other way round with the satellite preceeding the nucleus. And yet, in the case of these two relations the nucleus and satellite can theoretically occur in any possible order with the whole construction being -in principle- linguistically correct.

For example, consider the following texts. In the first text, the ordering METHOD^GOAL is used together with RST-Purpose, in the second text, the ordering GOAL^METHODs is used together with RST-Means.

1. Choose Element Properties to add elements to the style.[3]
2. In the Element Properties dialog box, enter the offset of the multiline element.
3. You need to choose Add to add the element.
4. Repeat these steps to define another element.[4]
5. Choose OK to save the style of the multiline element and to exit the Element Properties dialog box.

1. Add elements to the style by choosing Element Properties

[3] The choose action doesn't automatically cause elements to be added. Rather, it enables elements to be added by achieving a precondition, that of opening the Element Properties dialog box.

[4] Clearly, this is a 'conditional' or 'optional' GOAL. To paraphrase: *If you wish to define another element, repeat these steps*. The current domain model does not allow for such a possibility.

2. In the Element Properties dialog box, it is necessary to enter the offset of the multiline element.
3. Add the element by choosing Add.
6. Define another element by repeating these steps .[5]
7. Save the style of the multiline element and to exit the Element Properties dialog box by choosing OK.

As a matter of fact, the proper ordering of nucleus and satellite in the sentence may depend on a number of factors that can lead to overriding the prototypical order. We choose not to represent any ordering constraints within the text plan itself, but rather, delegate this issue to the sentence planner. When constructing sentence plans that involve aggregation using RST-MEANS or RST-PURPOSE, the sentence planner should take into account at least the following factors:

- **Heavy versus light constituents**: Grammatically, the preference is to place heavy constituents at the end. In our case, a heavy constituent would be an INSTRUCTION that is a conjunction of TASKS, acting as satellite.
- **Consistency**: Textually, there is a preference to be consist in the ordering of nucleus/satellite across sentences, at least within one and the same textual segment.
- **Readership**: At least in the absence of heavy constituents, the level of expertise of the intended readership can be used to impose a preferred order (TEXS2). A novice appears to prefer the order INSTRUCTION (satellite) - TASK (nucleus), whereas experts seem to favour the order TASK (nucleus) - INSTRUCTIONS (satellite).

The issue of relative ordering of GOAL and METHOD in instructional texts was previously addressed by Delin, Hartley and Scott (1996). Within a corpus of instructions in English and French, they undertook a contrastive analysis of expressions within the same sentence of the semantic relations of GENERATION and ENABLEMENT. These relations hold between pairs of actions in a task. Generation holds between two actions α and β if the performance of α automatically brings about the occurrence of β. Enablement holds between α and β if the execution of α brings about a set of conditions that are necessary, but not necessarily sufficient for the subsequent performance of β; in other words, α is a precondition for β.

In the expression of generation, there was a tendency in both languages to place the user GOAL (the generated action) before the METHOD (the generating action).

In the expression of enablement, the preferred ordering of the actions in the text respected their temporal sequence in the world—the 'iconic' ordering. However, there were many occurrences of non-iconic ordering, which appeared to be chosen for one of two reasons:

- To present to the user the purpose of performing the GOAL action, and to indicate to the user that the goal named in the purpose expression (*to DO, in order to DO*) is either optional or contrastive (cf. Van der Linden and Martin (1995)), or that it has global scope over a range of actions to be related subsequently (cf. Thompson (1985));
  or
- To prevent the user from making a mistake by performing a task prematurely (*before DOing*).

[5] Clearly, this is a 'conditional' or 'optional' GOAL. To paraphrase: *If you wish to define another element, repeat these steps*. The current domain model does not allow for such a possibility.

The authors observed in French a combination of ordering and phrasing that seemed to be unacceptable in English, in which the enabled action was presented first as a command and the enabling action second by means of an *apres avoir* phrase. They were not able to suggest why this ordering would be acceptable in French and acknowledged that more data was required in order to tell if its appearance reflected a general tendency. They also acknowledged that constraints other than local and global purpose, contrast and optionality were likely to affect ordering. Considerations of textual well-formedness such as continuity of reference, closeness of anaphoric links, etc. also exert an influence.

The relative GOAL and METHOD ordering thus remains an open issue and a challenge for text and sentence planning in AGILE. Given that our corpus analysis (the CORP deliverable of AGILE WP 3) did not distinguish between enablement and generation, we do not have the kind of data that Delin, Hartley and Scott (1996) discuss. It therefore remains to be seen whether can perform some additional corpus analysis in order to substantiate a more general control mechanism for the GOAL and METHOD ordering with the AGILE system.

3.1.4 Granularity

Under the general denomination of aggregation we can also share granularity. In our case, we can consider the integration of a METHOD's CONSTRAINT directly into the sentence that realises the METHOD's SUBSTEPS, as in the following examples:

In **Windows,** choose *Multiline style* from the *Object Properties* toolbar.
In **DOS or UNIX,** choose *Multiline style* from the *Data* menu.

Using **Windows,** you should choose *Multiline style* from the Object Properties toolbar.
Using **DOS or UNIX,** you should choose *Multiline style* from the Data menu.

If you are using **Windows,** you should choose *Multiline style* from the Object Properties toolbar.
If you are using **DOS or UNIX,** you should choose *Multiline style* from the Data menu.

Here, we consider a CONSTRAINT to identify a specific context of application, and as such can be realised as a location (as in the first pair of sentences), as an instrument (as in the second pair of sentences), or by a conditional construction (as in the third pair of sentences). All three realisations are consistent with the intuitions behind the CONSTRAINT concept as it is formulated in the Domain Model.

In the text plan we can already plan the inclusion of the CONSTRAINT, like in the examples above. We can do so by differentiating different types of CONSTRAINT-nodes in the text plan, in a similar vein as we did above. Here, we will use SEPARATE-CONSTRAINT, LOCATION-CONSTRAINT, MEANS-CONSTRAINT and CONDITION-CONSTRAINT. The reason for making the differentiation already during text planning rather than deferring to the stage of sentence planning, is that the decision for one type over the other depends on

- the overall type of text we are planning, i.e. running text or using enumeration; and,
- what style the text is to be generated in.

In case the text is to be realised in imperative voice using enumeration, then the only natural choice appears to be to use SEPARATE-CONSTRAINT, which keeps the CONSTRAINT separate from the sentence realising the METHOD's SUBSTEPS. The

reason being that there is no explicit mention of the hearer (ruling out "*If you are using **Windows**, choose* ..."). Hence, in this case the text planner should insert SEPARATE-CONSTRAINT into the text plan.

On the other hand, if we are to generate a running text, we can use LOCATION-CONSTRAINT if the imperative voice is used, or either MEANS-CONSTRAINT or CONDITION-CONSTRAINT, depending on the grammatical constructions available in a given language, if active voice is used. For example, the MEANS-CONSTRAINT realisation is not available in Czech, but is possible in Russian.

Upon encountering a constraint under one of these nodes in the text plan, the sentence planner should act as follows:

- SEPARATE-CONSTRAINT: A separate sentence-plan should be created, realising the CONSTRAINT. This is the way it is done already in the TSM for the intermediate prototype. The corresponding SPL contains the following statement:
  ```
  (P / object :NAME gui-windows)
  ```
- LOCATION-CONSTRAINT becomes a Spatial-Locating Circumstance within the clause realising the TASK. It is reflected by the following bit of SPL code:
  ```
  :spatial-locating(P / object :NAME gui-windows)
  ```
- MEANS-CONSTRAINT is realised by the Range of RST-Means, where the Domain corresponds to the TASK. The following bit of SPL code reflects this:
  ```
  (RST / RST-Means Manner
     :Domain …
     :Range(Range / dispositive-material-action
       :lex use :ACTEE (P / object :NAME gui-windows)))
  ```

- CONDITION-CONSTRAINT is realised by the Domain of RST-Logical-Condition, where the Range corresponds to the TASK. The following bit of SPL code reflects this:

```
(RST / RST-Logical-Condition
   :Domain(Range / dispositive-material-action
     :lex use :ACTEE (P / object :NAME gui-windows)
   :Range))
```

3.2 Explicit discourse markers for sequencing

In this section, we focus on the realisations of the SEQUENCE relation in procedural instructions. We are concerned with other means reflecting SEQUENCE than the straightforward ordering of discourse units. What we are especially concerned with is the realisation of content in lists vs. in running text and the accompanying layout decisions, the use of explicit signals for SEQUENCE relations, and the decision whether to use numbering vs. linguistic sequence markers. By *linguistic sequence markers* we mean expressions like first, second, third, now, then, finally, lastly (and their Bulgarian, Czech and Russian counterparts).

We first provide a discussion, which aims to substantiate the need for explicit signalling of SEQUENCE in procedural instructions. We also discuss alternative ways of realising relations between pieces of content, using the means distinguished above. Finally, we provide a formalisation of the approach proposed here.

3.2.1 The need for explicit sequence markers in AGILE texts

A list of SUBSTEPS in an A-box is assumed to be ordered in accordance with the order of executing the SUBSTEPS to achieve a given GOAL. It is natural for instructions in procedural style to obey this order when the SUBSTEPS are realised. This is what the current text planner does, and also in the final prototype, the A-box ordering of SUBSTEPS will be obeyed in the realised texts.

When the order of instructions accords with the temporal order of carrying out these instructions, there does not seem much need for explicitly indicating the sequential order. Indeed, it is not difficult to interpret the following two short texts:

To save a document under a different name

1. Select the *Save As* command from the *File* menu.
2. Enter the file name in the *File name* box.
3. Click the OK button.

To save a document under a different name, select the *Save* command from the *File* menu, enter the file name in the *File name* box, and click the OK button.

The first of these two texts demonstrates a numbered list realisation of step-by-step procedural instructions. In this case, the overt numbering and layout together make the sequencing so clear that additional linguistic sequence markers would be inappropriate, as the following example demonstrates.

To save a document under a different name
1. First select the *Save As* command from the *File* menu.
2. Now enter the file name in the *File name* box.
3. Finally, click the OK button.

On the other hand, inserting explicit sequence markers into the running-text realisation of the same step-by-step procedural instructions shown above, is much less superfluous. In fact, a reader can find it useful to orient herself:

To save a document under a different name, first select the *Save* command from the *File* menu. Then enter the file name in the *File name* box. Finally click the OK button.

vs.

To save a document under a different name, first select the *Save* command from the *File* menu. Then enter the file name in the *File name* box, and finally, click the OK button.

vs.

To save a document under a different name, first select the *Save* command from the *File* menu, then enter the file name in the *File name* box, and finally click the OK button.

The running text seems more natural, and certainly is easier to process when sequences are indicated explicitly by discourse markers. The discourse markers can be linguistic ones, as above, or can again be numbers, resulting in "horizontal" list. Henceforth, we only discuss the explicit marking of sequences in running texts by linguistic markers.

The more complex semi-running text below illustrates this even better. Two versions are shown: one with numbering, the other with only linguistic sequence markers. We have used abundant explicit sequence markers, which we underlined to make them easier to notice. If none of the markers were present, the text would be quite unnatural and we believe also difficult to process.

To create a multiline style

First open the Multiline Styles dialog box.
Windows Choose Multiline Style from the Object Properties toolbar or the Data menu.
DOS or UNIX Choose Multiline Style from the Data menu.

Now choose Element Properties to add elements to the style. The Element Properties dialog box appears. First enter the offset of the multiline element in the Element Properties dialog box. Then select Add to add the element. To define another element, repeat these steps.

Finally choose OK to save the style of the multiline element and to exit the Element Properties dialog box.

To create a multiline style

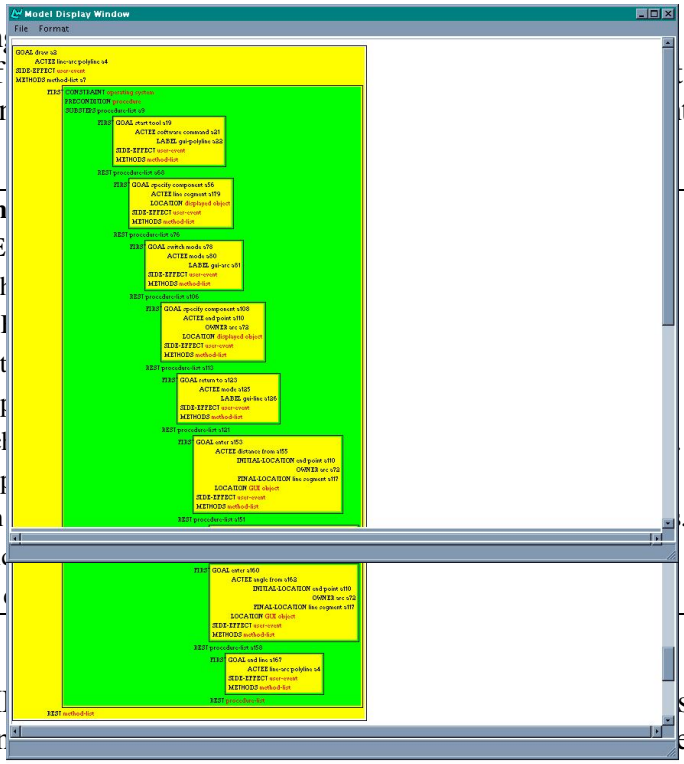
1. Open the Multiline Styles dialog box.
Windows Choose Multiline Style from the Object Properties toolbar or the Data menu.
DOS or UNIX Choose Multiline Style from the Data menu.

2. Choose Element Properties to add elements to the style. The Element Properties dialog box appears. First enter the offset of the multiline element in the Element Properties dialog box. Then select Add to add the element. To define another element, repeat these steps.

3. Choose OK to save the style of the multiline element and to exit the Element Properties dialog box.

So far it may seem that explicit marking of sequences would be appropriate for running-text instructions, and rather superfluous for numbered-list instructions. It is indeed the case

that the numberin demonstrated in the preceding pair of tional text get more complex, simply ir t:



To draw a line an
1. Start the PLINE
Windows: From th
DOS and UNIX: I
2. Specify the start
3. Specify the endp
4. Enter **a** to switch . Select OK.
5. Specify the endp
6. Enter **l** to return . Select OK.
7. Enter the distanc
8. Press Return to

In this text (IM ser is given full step-by-step instruction e there is a structure present in the perf t entirely obscures it. Namely, the task of drawing a line and arc combination polyline consists of three main sub-tasks, namely drawing a line, then drawing an adjacent arc and then drawing another line.

This structure should also be reflected in the A-box specification of the text content. Let us consider two example A-boxes shown in Figure 9 and Figure 10, which both model a simplified content based on IMD-Text2. The content is simpler than that in the above text, for the sake of brevity. The A-box in Figure 9 does not involve a hierarchical task structure, as it contains simply a list of PROCEDUREs with no METHODs.

The A-box in Figure 9, in which the hierarchical structure of the task is not reflected, and its straightforward clause-by-clause realisation:

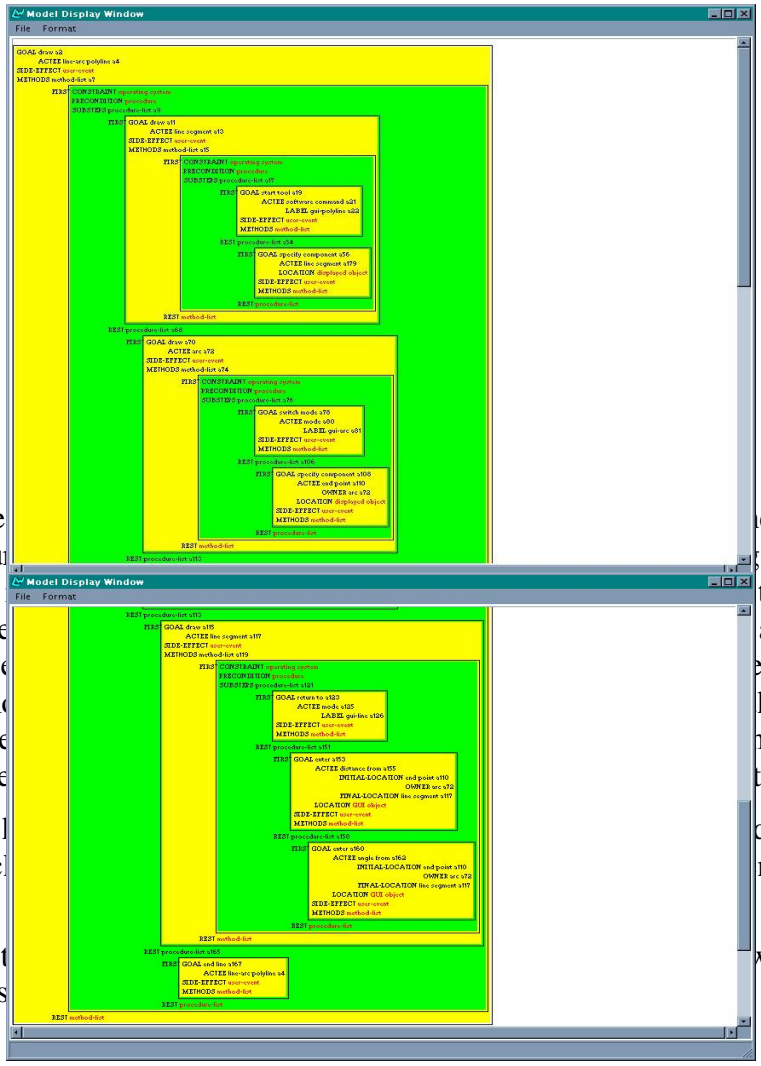
To draw a line and arc combination polyline
1. Start the PLINE command.
2. Specify a line segment.
4. Switch to the Arc mode.
5. Specify the endpoint of the arc.
6. Return to Line mode.
7. Enter the distance of the line from the endpoint of the arc.
8. End the line and arc combination polyline.

Figure 9: A-box not reflecting hierarchical task structure

Now compare the A-box in Figure 10, which models the hierarchical task structure by containing the PROCEDUREs with GOALs corresponding to "draw a line segment" and "draw an arc segment", and by containing METHODs for achieving these GOALs.

The A-box in Figure 10 would be straightforwardly realised clause-by-clause as follows:

To draw a line and arc combination polyline
1. Draw a line segment.
2. Start the PLINE command.
3. Specify the start point of the line segment.
4. Specify the endpoint of the line segment.
5. Draw an arc segment.
6. Switch to Arc mode.
7. Specify the endpoint of the arc.
8. Draw a line segment.
9. Return to Line mode.
10. Enter the distance of the line from the endpoint of the arc.
11. End the polyline.



If the A-box re ether or not to reflect this structu text planning. We believe that to the user is necessary, or at le about drawing more complex obje es consisting of various combinati he instructions are explicit with re made easier for the user. Therefore tructured.

Clearly, if the l de perspicuous for the reader, suc nt. We need to exploit layout and

The next quest wing examples illustrate some pos

To draw a line and arc combination polyline

1. Draw a line segment.

   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.

   Then specify the start point of the line segment and the endpoint of the line segment.

2. Draw an arc segment.

   First switch to Arc mode by entering **a**. The Arc mode confirmation dialog box appears. Select OK.
   Then specify the endpoint of the arc.

3. Draw another line segment.

   First return to Line mode by entering **l**. The Line mode confirmation dialog box appears. Select OK.
   Then enter the distance and angle of the line in relation to the endpoint of the arc.

4. Press Return to end the polyline.

In this example, we have used a numbered list for the top-level GOALs, and use explicit sequence markers for the lower-level GOALs, along with aggregation:

Another possibility is to use explicit sequence markers for the top-level GOALs within the list of SUBSTEPS, and a numbered list of the lower-level SUBSTEPS (the numbering re-sets within each sub-list). Such a version is illustrated below:

To draw a line and arc combination polyline

First draw a line segment.

1. 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.
2. Specify the start point of the line segment.
3. Specify the endpoint of the line segment.

Then draw an arc segment.

1. Enter **a** to switch to Arc mode. The Arc mode confirmation dialog box appears.
2. Select OK.
3. Specify the endpoint of the arc.

Then draw another line segment.

1. Enter **l** to return to Line mode. The Line mode confirmation dialog box appears.
2. Select OK.
3. Enter the distance and angle of the line in relation to the endpoint of the arc.

Finally, press Return to end the polyline.

It seems, for the text of this complexity, that both exemplified ways of realising the SEQUENCEs and EMBEDDINGs are sensible.

Assuming the hierarchical structure of a task is reflected in the hierarchical structure of an A-box, it is straightforward to generate a text with the corresponding hierarchical structure. This is in fact more straightforward than generating a text in which the hierarchical structure is neglected. In order to achieve the latter, some parts of the A-box content need to be skipped, i.e. not explicitly realised. On the other hand, when we want the

hierarchical structure of the task to be reflected in accordance to the A-box, we only have to ensure that all A-box GOALs are realised, and we have to hierarchically structure the text.

Another example, which demonstrates the usefulness of providing such explicit hierarchical structuring in addition to explicit marking of SEQUENCEs, is the following text based on IMD-Text4:

To specify the properties of a multiline and save the style

From the Data menu, choose Multiline Style. The Multiline Style dialog box appears.

First specify the properties of the multiline.

1. Choose Multiline Properties. The Multiline Properties dialog box appears.
2. Select Display Joints to display a line at the vertices of the multiline.
3. Under Caps, select a line or an arc for the startpoint of the multiline. Then select a line or an arc for the endpoint of the multiline. Lastly, enter an angle.
4. Under Fill, select On to display a background color.
5. Choose Color. Then select the background fill color from the Select Color dialog box.
6. Choose OK to return to the Multiline Styles dialog box. The Multiline Properties dialog box disappears.

Now save the style.

1. Under Name, enter the name of the style.
2. Under Description, enter a description.
3. Select Add to add the style to the drawing.
4. Select Save to save the style to a file.
5. Choose OK and close the dialog box.[6]

Here, we have used explicit sequence markers only for those top-level GOALs within the list of SUBSTEPS, which themselves contain further SUBSTEPS. These lower-level SUBSTEPS are again realised by numbered lists (again we re-set the numbering within each sub-list, esp. because each sub-list is itself quite long), and we use running text within these SUBSTEPS (cf. 1, 3, 5 and 6 under "first"). Note that we use explicit sequence markers within the "embedded" running texts as well (cf. 3 and 5 under "first").

Before we conclude this discussion of various combinations of layout, explicit marking of SEQUENCEs and EMBEDDING and running text vs. numbered lists, let us point out once more that it is indeed necessary to combine these aspects. Consider the following example, containing four versions of the same procedural instructions:

To create a multiline style, first open the Multiline Styles dialog box. If you are using **Windows,** choose Multiline Style from the Object Properties toolbar or the Data menu. If you are using **DOS or UNIX**, choose Multiline Style from the Data menu. Now choose Element Properties to add elements to the style. The Element Properties dialog box appears. First enter the offset of the multiline element in the Element Properties dialog box. Then select Add to add the element. To define another element, repeat these steps. Finally, choose OK to save the style of the multiline element and to exit the Element Properties dialog box.

VS.

[6] The closure is an automatic consequence of choosing OK. A less ambiguous wording would be *Choose OK to close the dialog box.* An unambiguous wording would be *Close the dialog box by choosing OK.*

To create a multiline style

1. Open the Multiline Styles dialog box.
   Windows Choose Multiline Style from the Object Properties toolbar or the Data menu.
   DOS or UNIX Choose Multiline Style from the Data menu.
2. Choose Element Properties to add elements to the style.
   The Element Properties dialog box appears.
3. Enter the offset of the multiline element in the Element Properties dialog box.
4. Select Add to add the element.
5. To define another element, repeat these steps.
6. Choose OK to save the style of the multiline element and to exit the Element Properties dialog box.

vs.

To create a multiline style

First open the Multiline Styles dialog box.
   Windows Choose Multiline Style from the Object Properties toolbar or the Data menu.
   DOS or UNIX Choose Multiline Style from the Data menu.

Now add elements to the style.
   1. Choose Element Properties. The Element Properties dialog box appears.
   2 Enter the offset of the multiline element in the Element Properties dialog box.
   3 Select Add to add the element.
   4. To define another element, repeat these steps.

Finally, choose OK to save the style of the multiline element and to exit the Element Properties dialog box.

vs.

To create a multiline style

1. Open the Multiline Styles dialog box.
   Windows Choose Multiline Style from the Object Properties toolbar or the Data menu.
   DOS or UNIX Choose Multiline Style from the Data menu.
2. Add elements to the style. First choose Element Properties. The Element Properties dialog box appears. Then enter the offset of the multiline element in the Element Properties dialog box. Finally select Add to add the element. To define another element, repeat these steps.
3. Choose OK to save the style of the multiline element and to exit the Element Properties dialog box.

The first, fully running text is obviously a bad one, as it blurs everything together.

The second, fully numbered list, in our opinion, obscures the hierarchical structure of the task. As a consequence of this, the reader does not know which steps she should repeat to define another element. Clearly, she does not have to start from step 1. But where the iteration begins she does not know, because no sequence of steps stands out from the rest (we believe it is steps 2-4 that need to be repeated).

The third and fourth versions exercise, we think, a suitable combination of layout, explicit marking of SEQUENCEs and EMBEDDING and running text vs. numbered lists.

3.2.2 Formalisation

In the preceding section, we have made a number of observations concerning layout, explicit marking of SEQUENCEs and EMBEDDING and running text vs. numbered lists. Following upon the discussion, we present here a specification of the functionality that is needed in the TSM in order to handle explicit marking of SEQUENCE, ALTERNATIVE and EMBEDDING relations in texts, which realise the related pieces in separate sentences.

Our main conclusion was that a combination of layout, running text vs. numbered lists, and explicit marking of relations as realisation means within a procedural text is desirable, rather than a fixed use of one way of realising content relations. This means that the choice of means to realise a particular piece of content depends not only on the content per se, but also on the realisation of other pieces.

We also observed that the texts appear more natural when their structure somehow explicitly reflects the hierarchical structure of the content (if/as captured in the A-box). When we want the hierarchical structure of the task to be reflected in accordance to the A-box, we have to ensure the following:

- that all A-box GOALs are realised (this is nothing new for the TSM)
- that a suitable combination of layout, running text vs. numbered lists, and explicit marking of relations are utilised

The formalisation splits into two branches. The first concerns Tasks realised by separate sentences. The second concerns Task realised by clauses within a complex sentence, due to aggregation.

3.2.2.1 Explicit sequence markers with separate tasks

When a separate task is being inserted, which happens in the system INSTRUCTION-TASKS-AGGREGATION, it has to be decided whether to use running text or a list.

```
System: SEPARATE-INSTRUCTION-TASKS-SEQUENCE:
    More-Separate-Tasks →
          [Running-Tasks]
          [List-Tasks]
```

The choice between list and running text is to a great degree determined by the genre of step-by-step instructions in procedural style. As we saw above, an entirely running text is out of question. At least the top-level of instructions (i.e. the realisations of the top-level GOALs in a PROCEDURE-lists under METHODs of the GOAL at the root of the A-box, see Figure 11), are thus realised as a list. As for the lower-levels, we saw above that a numbered list of instructions throughout the whole text is bad. Therefore, some decisions on the basis of the complexity of the content being expressed need to be made.

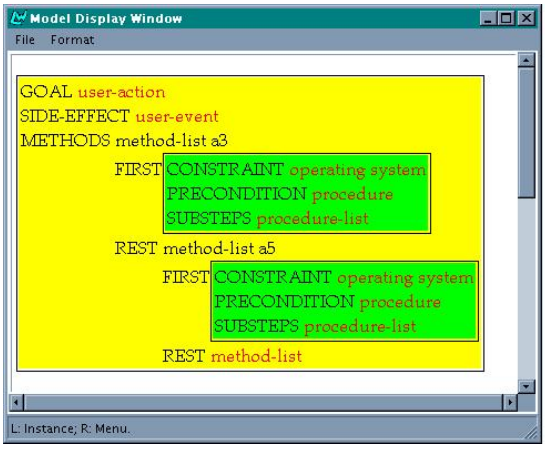


Figure 11: Top-level SUBSTEPS in A-box

We propose to realise simple content at lower-than-top levels as running text, and any complex content as a list. Thus, if the lower-level SUBSTEPS do not include alternative METHODs with Constraints, or further embedded complex SUBSTEPS, we realise them as a running text. These conditions are similar to those on aggregation (cf. section 3.1). The chooser below reflects these considerations:

```
Chooser: SEPARATE-INSTRUCTION-TASKS-SEQUENCE-CHOOSER:
    If at the top-level in A-box,
    Then choose List-Tasks
    Else
          If at any lower level in A-box,
          and the SUBSTEPS are simplex
          and have no Constraints,
          Then choose Running-Tasks
          Else choose List-Tasks
```

In addition, it is suitable to include explicit linguistic sequence markers in the lower-level running text.

Let us now turn to the decision whether to mark a SEQUENCE relation explicitly or not. This decision can also be made when a separate task is being inserted. Moreover, we noted above that there is some interplay between the choice of list vs. running text and explicit SEQUENCE marking. The top-level of instructions always uses explicit markers, for the lower levels we make a more elaborate decision. So, we have two systems making this decision, specified as follows:

```
System: SEPARATE-LIST-INSTRUCTION-TASKS-SEQUENCE-MARKING:
    More-Separate-Tasks AND List-Tasks →
          [Explicit-Marking-Tasks]
          [No-Marking-Tasks]

System: SEPARATE-RUNNING-INSTRUCTION-TASKS-SEQUENCE-MARKING:
    More-Separate-Tasks AND Running-Tasks →
          [Explicit-Marking-Tasks]
          [No-Marking-Tasks]
```

It appears typical for the genre of step-by-step instructions in procedural style that in the top-level instructions explicit marking of SEQUENCEs is employed. At the second level of instructions, it seems sensible to use explicit sequence marking if there are at least two SUBSTEPS. At any lower level of embedding, we propose to use **no** marking of SEQUENCEs in lists, and explicit marking of SEQUENCEs in running texts, where it again seems sensible to use explicit sequence marking if there are at least two SUBSTEPS and. These criteria are captured in the choosers specified below.

```
Chooser: SEPARATE-LIST-INSTRUCTION-TASKS-SEQUENCE-MARKING-CHOOSER:
    If at the top-level in A-box,
    Then choose Explicit-Marking-Tasks
    Else
          If at the second-level in A-box,
          Then
                If there are more than 1 SUBSTEPS
                Then choose Explicit-Marking-Tasks
                Else choose No-Marking-Tasks
          Else  choose No-Marking-Tasks

Chooser: SEPARATE-RUNNING-INSTRUCTION-TASKS-SEQUENCE-MARKING-CHOOSER:
    If there are more than 1 SUBSTEPS
    Then choose Explicit-Marking-Tasks
    Else choose No-Marking-Tasks
```

Note that the top-level is always realised by a list. Any further embedded level can be a list or a running text (if it contains only simplex SUBSTEPS with no alternative METHODs distinguished by Constraints).

As for the realisation of explicit SEQUENCE marking, we have the following observations:

- In running text, linguistic markers rather than numbering are used.
- In a list, either numbering or linguistic markers can be used. We observed that when explicitly reflecting SEQUENCEs, it suffices to use either numbering or linguistic markers, i.e. there is no need to use explicit linguistic markers together with numbering.

These observations lead to the following systems:

```
System: RUNNING-INSTRUCTION-TASKS-SEQUENCE:
    Running-Tasks AND Explicit-Marking-Tasks →
          [Linguistic-Markers-Tasks]

System: LIST-INSTRUCTION-TASKS-SEQUENCE:
    List-Tasks and Explicit-Marking-Tasks →
          [Number-Markers-Tasks]
          [Linguistic-Markers-Tasks]
```

The decision whether to use a numbered list or a list with linguistic markers involves a certain degree of freedom in balancing out the means when combining them to realise a more complex content. This concerns in particular the content of the top-level SUBSTEPS, especially when there are further lower levels of SUBSTEPS.

We propose to consider it a matter of stylistic choice, whether the top-level SUBSTEPS are realised as a numbered list or using linguistic sequence markers. If we want to make this stylistic decision automatically, as to relieve the user of this burden, it is possible to choose between numbering and linguistic markers for the top level depending on how many top-level SUBSTEPS there are. If there are few (say, up to three), linguistic markers seem a natural choice. If there are more, it appears better to use numbering.

The realisation of the second-level SUBSTEPS can then take a complementary form, in order to avoid repetition. So, if the top-level uses linguistic markers, the second-level can use numbered list. If the top-level uses numbered list, the second-level can use linguistic markers.

We said above that any further lower levels of SUBSTEPS (i.e., the third embedded level or more) would be realised as running text, and therefore without explicit markers, unless they themselves include alternative METHODs with Constraints, and/or further complex SUBSTEPS. If they do include alternative METHODs with Constraints, and/or further complex SUBSTEPS, they would be realised by a list. In this case, we do not include any explicit SEQUENCE markers. We only use indentation (which is to be inserted by the sentence planner).

```
Chooser: LIST-INSTRUCTION-TASKS-SEQUENCE-CHOOSER:
    If at the top-level in A-box,
    Then
          If there are at most three top-level PROCEDUREs,
          Then choose Linguistic-Markers-Tasks
          Else choose Number-Markers-Tasks
    Else
          If at the second-level in A-box,
          Then
                If the top-level used Linguistic-Markers-Tasks,
                Then choose Number-Markers-Tasks
                Else choose Linguistic-Markers-Tasks
          Else
                if the SUBSTEPS are simplex
                and have no METHODs with Constraints
                Then choose Linguistic-Markers-Tasks
                Else choose Number-Markers-Tasks
```

Obviously, the decisions concerning layout, explicit marking of relations and running text vs. numbered lists need to be closely coupled with the decisions concerning aggregation. We will return to this point in the implementation phase (TEXM3).

Last but not least, we should point out that the exact realisation of the SEQUENCE marker depends on how many-th element of a list is being marked.

For numbered lists this does not seem to be an issue for the text planner, because numbering of list elements is to be done through HTML annotation of the output (cf. (TEXM2) and (TEXM1) for examples). The HTML tags are then to be interpreted by a suitable viewer, which takes care of the list numbering.

The linguistic markers, on the other hand, need to be realised by the lexico-grammar. The SPL thus needs to contain sufficient information to enable the grammar to realise the linguistic marker appropriate for a given element in a sequence. We propose to use the following linguistic markers depending on how many elements there are in a sequence:

- **English:**
  2 elements: *First, Now*
  3 and more elements: *First*, (*Then*)[+7], *Finally*

[7] (Then)+ means one or more occurrences of 'then'.

- **Bulgarian:**
  2 elements: *Първо, Сега*
  3 and more elements: *Първо,* (*После*)$^{+}$, *Накрая*
- **Czech:**
  2 elements: *Nejdříve, Nyní*
  3 and more elements: *Nejdříve,* (*Potom*)$^{+}$, *Nakonec*
- **Russian:**
  2 elements: *Сначала, Затем* (*Теперь* is also possible, but the context of its usage is restricted to the here-and-now interaction style)
  3 and more elements: *Сначала , затем, наконец*

Below we show an example SPL for the sentence "*Then choose the color of the element*" in Czech:

```
(EXAMPLE
 :NAME S10-IMP-simplified
 :SET-NAME   Cz-D1-T1-IMP
 :TARGETFORM   "Potom zvolte barvu elementu."
 :LOGICALFORM
    (Act / DM::CHOOSE
        :CONJUNCTIVE then
        :SPEECHACT Imperative
        :ACTEE
            (Obj1 / DM::COLOR
                :GENERALIZED-POSSESSION-INVERSE
                 (C2 / DM::style-element)
)))
```

The statement "`:CONJUNCTIVE then`" reflects the choice of the linguistic marker. It is a reference to the following SPL-macro:

```
(defspl-macro :conjunctive
  ((then
         :conjunctive-relation-q conjunctive
         :conjunctive-relation-id (?rr / rhetorical-relation)
         :process-regulated-q processregulated
         :necessity-q nonecessity
         :sequence-q notsequence
         :time-relation-q timerelation
         :time-precedence-q precedence
         :time-precede-q subsequent
         :time-separation-q separate
         :small-separation-q notsmall))
```

It is a matter of grammar development to implement the realisations of the linguistic markers for SEQUENCEs. The respective pieces of SPL then need to be used for each language. They are to be inserted by the Sentence planner.

In order for the Sentence planner to be able to insert the appropriate piece of SPL, it needs to know which element in a SEQUENCE is being realised and how many elements the SEQUENCE has in total (only two or more). In order to keep track of the position of an element within a sequence, we think we need to use a counter, which will be updated when a separate Task with an explicit linguistic marker is inserted. Moreover, since there can be explicitly marked sequences with linguistic markers at different levels of the hierarchical structure in a text, we need a different counter for each such level.

The piece of SPL inserted by the Sentence planner for an element with an explicit linguistic marker will then depend on the value of the appropriate counter. As noted above, for an element in a numbered list, the Sentence planner needs to insert a particular HTML-annotation, but no counters are necessary, because the numbering will be realised by the HTML viewer.

3.2.2.2 Explicit sequence markers with conjoined tasks

This is a rudimentary attempt to interleave aggregation and explicit linguistic sequence markers. The idea is to use explicit markers when there are more than two conjoined Tasks, and not to use explicit markers for less than two.

```
System: AGGREGATED-INSTRUCTION-TASKS-SEQUENCE
    More-Conjoined-Tasks →
          [Conjoined-Tasks-With-SeqM]
          [Conjoined-Tasks-Without-SeqM]

Chooser: AGGREGATED-INSTRUCTION-TASKS-SEQUENCE-CHOOSER
    If there are more than two SUBSTEPS,
    Then choose Conjoined-Tasks-With-SeqM
    Else choose Conjoined-Tasks-Without-SeqM
```

The difference between conjunction with and without explicit markers will be reflected in the SPL as follows:

- with explicit discourse marker we use RST-Sequence
- without explicit discourse marker we use Conjunction

Example SPLs for these two cases cases of realising the same content are shown below. The first SPL yields the Czech realisation of "Add an element and save the style", the second SPL should yield "Add an element, then save the style and then press OK".

```
(EXAMPLE
:NAME   5-add-conj
:SET-NAME   CONJ
:TARGETFORM   "Přidejte element a uložte styl"
:LOGICALFORM
    (c1 / conjunction
    :Domain
          (S1 / DM::ADD
                :SPEECHACT IMPERATIVE
                :ACTEE (P1 / DM::style-element))
    :Range
          (S2 / DM::SAVE
                :SPEECHACT IMPERATIVE
                :ACTEE (P1 / DM::STYLE))
))

(EXAMPLE
:NAME   5-add-RSTseq
:SET-NAME   CONJ
:TARGETFORM  "Přidejte element, potom uložte styl a potom stikněte OK."
:LOGICALFORM
(c1 / RST-SEQUENCE
    :Domain
          (S1 / DM::ADD
                :SPEECHACT IMPERATIVE
                :ACTEE (P1 / DM::style-element))
    :Range
    (c2 / RST-SEQUENCE
        :Domain
        (S2 / DM::SAVE
                :SPEECHACT IMPERATIVE
                :ACTEE (P1 / DM::style))
        :Range
        (S3 / DM::PRESS
                :SPEECHACT IMPERATIVE
                :ACTEE (P2 / OBJECT :NAME gui-ok)))
))
```

4. Summaries and basic steps instructions

In this section we consider a specific genre of instructions which differs from the full instructions in procedural style that is directed to the widest readership. We consider here an instruction genre which communicative goal is to address a specific readership – for example, militaries, specialists, people in emergency situations, and so on. This genre is motivated by the pragmatic goals of instructions - for example, the more lapidary style lacking motivating goals is preferred for maintenance manuals intended for use by the US military (expected to function in "robot mode"). In the case of CAD/CAM instructions it can be useful for a more experienced readership. The source of modeling this style is an A-box that describes the content for the full instructions. Having the content for the full procedural text presented as an A-box we can abbreviate the content in some definite manner when mapping it to the text plan structure. We describe the process as the functional definition for the text types of abbreviated text we consider on the basis of our A-box representations.

In the next section we model variants of the types of abbreviated texts following the functional definitions that are possible following investigation of the level of detail available to us in the A-box. The reason is the constrained research area, since we have no possibilities to consider the genres from the point of view of motivation by pragmatic goals of instructions. So it concerns the basic idea of presenting the content with a decreasing level of detail at each successive stage of the abbreviation process. We hope that the resulting types of texts could have their motivation, as we try to show.

4.1 Discussion of the text type

The main difference between this genre of instructions and the full procedural instructions is that the former cannot be expressed in descriptive style. Following the terminology of the AGILE project we call the full procedural instructions expressed in descriptive style, that is based on narrative discourse and is realised in indicative mood - “impersonal style”. So the abbreviated texts of instructions - summaries and basic steps instructions as we define them later - are always based on narrative discourse and can be realised only in the main style of procedural instructions – imperative (imperative mood). The obvious consequence of the general style of the text type realisation is that it does not have descriptive style inclusions like a side-effect that can be seen as additional, elaboration type information.

So the first level of abbreviation in our case is omitting side-effects, being descriptive inclusions in full instructional texts. The resulting text will be the briefest version of a full instructional text. This operation is valid only if the “with side-effect” variant of the content presentation is chosen by the user. If the user opted for “without side-effect” then this variant will be equal to the full procedural text. For example, in AGILE we consider two possibilities to express the content of the following sentence:

(1) Choose Color.
Select the element's color from the Select Color dialog box. [without side-effect]
(2) Choose Color.
The Select Color dialog box appears. Select the element's color. [with side-effect]

And we have thus abbreviated variant for the second one:

Choose Color. Select the element's color.

The next step of abbreviation is related to the PRECONDITION slot of a METHOD. In our model it is a normal concept of the full procedural style of type PROCEDURE (user action). From the content point of view we can look at the METHODS slot of the PRECONDITION as an elaboration, thus only keeping its (obligatory) GOAL slot as indispensable elements of the content. So our second level of abbreviating of our A-boxes content is neglecting all slots of a PRECONDITION except for the GOAL, while considering the latter as the first methods of the main PROCEDURE in its METHODS slot. So, for example, for the following beginning fragment of the Text1, pages 47-48:

To create a multiline style First open the Multiline Styles dialog box using one of these methods: **Windows**: From the Object Properties toolbar or the Data menu, choose Multiline Styles. **DOS and UNIX**: From the Data menu, choose Multiline Styles. 1. Choose Element Properties to add elements to the style….

We'll have:

To create a multiline style First open the Multiline Styles dialog box. 1. Choose Element Properties to add elements to the style….

At the third level of the abbreviation process, we can focus on a single element of A-box, since only one element is realised as a USER-ACTION. This is the GOAL slot of a PROCEDURE. There are three possibilities for it to be included into the structure: to be the GOAL of the main PROCEDURE slot, to be the goal of a PROCEDURE as a daughter under METHOD, and to be a goal under an intermediate METHOD. Their syntactic realisations in the text thus differ according to their position in the A-box structure under different configuration concepts. We have three possibilities:

- Summary (goal-oriented) instructions: We express only higher-level Goals in the A-box and leave out the steps that lead to their accomplishment. This corresponds to realising the higher-level nodes in the text plan hierarchy, in order to give an "abstract" description of the content rather of the whole text of the procedure. This will present the topic of the instructional text. If we do this for a collection of A-boxes, we obtain the content of a complex instructional text. This type is considered and described in Section 7 as TOC.
- We can also consider another way of abbreviating the content of an A-box, namely through summarization (essentially, the "why's without the how's"). We try to express only higher level nodes in the A-box hierarchy in order to give a description of the procedure that does not get down to specific steps. However, the IMD texts the hierarchies are rather flat, so this seems to be a less sensible variant.
- Basic (step-oriented) instructions: We can express only the most embedded steps of methods, thus omitting higher level goals – the 'how' without the 'why'. An example of the Basic steps type of the content presentation for the text 1 is as follows:

To create a multiline style

Open the dialog box Multiline style.

Choose Element Properties.

[In the Element properties dialog box] Enter the offset of the multiline element.

Select Add.

Choose Color.

Select the element's color [from the Select Color dialog box].

Choose Linetype.

Select the element's linetype [from the Linetype dialog box].

Repeat these steps to define another element.

Choose OK.

The following stage of abbreviation process could be the abbreviation of Location slots presented in the example in square brackets. Although this text style may not look very plausible for a consumer user manual, since it lacks explanatory and motivating goals, it can be appropriate for a quick reference sheet (cheat sheet).

For Russian we have an additional style of expressing a basic steps type of text that does not exist in Bulgarian and Czech. It is the possibility to use the infinitive to express an imperative. For this we could consider two styles of presenting the content. The first one employs a modal with a 'necessity' meaning before the first step and then other steps without the modal:

Чтобы создать стиль мультилинии

Необходимо открыть диалоговое окно Multiline style.

Нажать кнопку Element Properties.

[В диалоговом окне Element properties] ввести смещение элемента мультилинии.

Нажать кнопку Add.

Нажать кнопку Color.

Выбрать цвет элемента [в диалоговом окне Select Color].

Нажать кнопку Linetype.

Выбрать тип линии [в диалоговом окне Linetype].

Повторить эти шаги, чтобы определить еще один элемент.

Нажать кнопку OK.

The second type of presenting the content is by enumerating the steps without any modal element:

Чтобы создать стиль мультилинии
1. Открыть диалоговое окно Multiline style.
2. Нажать кнопку Element Properties.
3. [В диалоговом окне Element properties] ввести смещение элемента мультилинии.
4. Нажать кнопку Add.
5. Нажать кнопку Color.
6. Выбрать цвет элемента [в диалоговом окне Select Color].
7. Нажать кнопку Linetype.
8. Выбрать тип линии [в диалоговом окне Linetype].
9. Повторить эти шаги, чтобы определить еще один элемент.
10. Нажать кнопку OK.

There also exists a fourth possibility – to combine procedures under different configuration situations to get a reasonable text, but it needs elaborated reasoning which cannot be achieved currently due to a lack of sufficient ontological detail in the DM. For example:

To create a multiline style
1. Open the Multiline Styles dialog box.
2. Add elements to the style.
3. Enter the offset of the multiline element.
4. Add the element.
5. Repeat these steps to define another element.
6. Save the style of the multiline element and exit.

To draw line and arc combination polyline
Draw the first line segment.
Switch to the Arc mode.
Draw an arc segment.
Switch to the Line mode.
Draw the second line segment.

So in the next section we consider the formalisation of the Basic steps text plan constructing from the A-box representation for the full instructional text.

4.2 Formalisation

Following upon the issues sketched in the preceding discussion, we present here a more detailed discussion of the functionality that is needed in the TSM in order to handle abbreviated procedural instructions.

We model three stages of the abbreviation process:

The first stage – omitting the side-effect. The first stage can be modelled by controlling (setting) the parameter guiding the inserting of side-effects.

The second stage – omitting the elaboration of a PRECONDITION's slots. The second stage is modelled by revising the PRECONDITION system. The algorithm is as follows: insert a precondition, do identify it with the proper GOAL, but do not preselect the PRECONDITION as a TASK. This leads to omitting whatever there is further to the precondition (which is of type PROCEDURE and therefore has, besides GOAL, also a SIDE-EFFECT slot and a METHODS slot).

The third stage (Summaries). Intuitively, we are focusing on the "why's" without the "how's". This means that we only realise the METHODS of the topmost PROCEDURE (which corresponds to the TITLE). For the embedded TASK, we omit the METHODS that specify how they could be obtained. This can be obtained by splitting the systems dealing with TASK and TASK-TITLE. We split the TASK-TITLE from TASK such that only with TASK-TITLE we will insert TASK-INSTRUCTIONS, and not with TASK. This concerns the TASK-TYPE system. For TASK, we only identify the TASK with the GOAL. We leave the recursion of TASK-INSTRUCTIONS and INSTRUCTION-TASKS intact.

So we get, for Summaries:

SUMMARY-TASK-TITLE	↔ GOAL of topmost PROCEDURE
SUMMARY-TASK-INSTRUCTIONS	↔ METHODS of topmost PROCEDURE
SUMMARY-INSTRUCTION-TASKS	↔ A METHOD's SUBSTEPS
SUMMARY-TASK	↔ GOAL of a PROCEDURE
SUMMARY-PRECONDITION	↔ PRECONDITION of a METHOD

The current systems ought to be straightforwardly adapted on the basis of these mapping rules for the most part by cutting things out.

The third stage (Basic steps); For the basic steps we considered the intuitive picture of the "how's without the why's". We again have a TASK-TITLE, we again need to do the METHOD's of the topmost procedure, but now the picture becomes a bit more intricate. For, what should we put into the cheat sheet? It might be the case that some methods are simplex and others are not. And we want to display only the why's without the how's.

If we encounter a METHOD that is simplex and it has a PROCEDURE that is simplex (in the procedure-list filling the SUBSTEPS slot) then we should insert the corresponding TASK into the text plan as a basic step. On the other hand, if the PROCEDURE is not simplex, then we should not insert a node in the text plan corresponding this particular PROCEDURE but only insert a node in the text plan for its METHODS list. This will ensure then that (eventually) only the TASKs corresponding to the how's get inserted into the text plan, without all the why's.

Hence we could get a text plan as follows:

CHEAT-SHEET-TASK-TITLE	↔ GOAL of topmost PROCEDURE
CHEAT-SHEET-TASK-INSTRUCTIONS	↔ METHODS of PROCEDURE
CHEAT-SHEET-INSTRUCTION-TASKS	↔ SUBSTEPS of a METHOD
CHEAT-SHEET-TASK	↔ a simplex PROCEDURE

We do not need to require simplicity of the METHODs themselves - if we want to introduce a precondition (corresponding to a METHOD's PRECONDITION) we might still want to do that. We would have to ensure (again) that only the how's get inserted, not the why's. So, if the PRECONDITION is a simplex PROCEDURE, then insert a CHEAT-SHEET-TASK and identify it with the PROCEDURE's GOAL. Otherwise, only insert a CHEAT-SHEET-TASK-INSTRUCTIONS and let the simplex PROCEDUREs somewhere under there be realised.

We described here the general solutions and actions needed to realise the abbreviated styles. During implementation, we intend to test the obtained texts, and it may be necessary

to formulate some additional differences in the concepts DM classification or some additional elements for their grouping to achieve a coherent text.

5. Descriptive texts

5.1 Introduction

Although the main style in instruction text is procedural, our text structuring oriented corpus analysis of instructional texts other than the CAD/CAM manual revealed that descriptive texts are very common in software manuals and other instructional texts in Bulgarian, Czech and Russian (cf. the TEXS2 deliverable). Also the AutoCAD manual contains text in this style. Another example is the Word **Help** bubbles that use a similar style.

Descriptive fragments do not form distinct parts (or only sometimes do so). Their specific character is that they are not obtained directly from operations upon the procedural description content, but present some additional information in different fragments of the instruction text (procedural, overviews). So, in principle, they are not mirror the procedural part in their structure, as the full instructional texts do. Rather, they are characterized not only by a different communicative goal, but also by the type of additional information like warnings, encyclopaedic, or additional information about constrains the system has over the user behaviour. For example,

Редактор не позволит Вам создать строки большей длины.
(Editor will not allow you to create lines of bigger length.)

The additional information implies additional grammar means, for example, negation as it is shown in 0.

We must model different styles basing on the content presented in the procedurally oriented descriptions that are represented by our Domain Model and in the A-boxes. So we are restricted to the type of texts which we call functional descriptive texts. As a side-effect we obtain sentences that are the result of combining a special type of information (namely, object functions) with a descriptive style, that is in indicative mood.

5.2 Functional descriptive text type

Basing ourselves on the A-boxes we can try to model functional descriptive texts. They can be expressed by two types of descriptions:

- "Descriptions of what Goal(s) a Method (user-action) achieves"
- "Descriptions of what Goal(s) a GUI object (...) achieves".

Let us call the first type – UA functional descriptive text - User actions-based functional descriptive text and OF functional descriptive text – Object function-based functional descriptive text. We first consider the UA functional descriptive text.

There are two types of semantic situations depending on the character of the user's action. The first one contains situations where the user triggers the button and through that action realises the function fixed for the object by the software design. There are realised by sentences like these:

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.

These descriptions contain the location – tool bar or dialog box pointing out where the GUI object is located. Stylistically they are badly composed as they attribute the achievement of some goal (adds an element) to a nominalisation of a user action (Selecting…) that by itself lacks the conceptual relation to the function realised as the Goal. But their stylistic fitness depends on the syntactic form of realization. Syntactic variations with a nominalised user's action realised as a subject highlight this undesirable semantic lacuna:

37. En: *{Selecting / the selection of} Add adds an element.*
38. Ru:
    **Нажатие кнопки Add добавляет элемент.*
    Selection button-Gen Add adds element-Acc.
39. Bu:
    Избирането на клавиша Add добавя елемент.
    Selection-def of button-def Add add-3sg element.
40. Cz:
    ** Vybrání/výběr tlačítka Add přidává element.*
    Selecting/selection button-gen Add adds element

Or with the permissive modal:

41. En: *{Selecting/the selection of} Add {enables/permits/allows} {you/the user/someone/ϕ} to add an element.*

42. Ru:
    ** Нажатие кнопки Add позволяет Вам добавить элемент.*
    Selection button-Gen Add allows you to-add element-Acc.
43. Bu:
    Избирането на клавиша Add {Ви позволява да добавите /
    Selection-def of button-def Add {you permit-3sg to add-2pl /

    позволява на потребителя да добави} елемент.
    permit-3sg to user-def to add-3sg} element.
44. Cz:
    *Vybrání/výběr tlačítka Add {dovoluje/umožňuje} přidávat elementy.
    Selecting/selection button-gen Add {allows /enables} add-inf elements-acc

The sentences marked by *s are not valid stylistically in inflectional Slavic languages we consider – Czech and Russian, but seems rather valid in English and Bulgarian.

However, the syntactic variations with a user's action realised as manner (secondary action) or as an instrument (nominalised) hide an undesirable semantic lacuna. The sentences thus formed became valid in Russian and Czech:

45. En: *By selecting Add, you add an element.*

46. Ru:
{Нажатием кнопки Add, Нажав кнопку Add,} Вы добавляете элемент.
Selection-Instr button-Gen Add you add element-Acc.

47. Bu:
{С избиране на, Като изберете} клавиша Add Вие добавяте елемент.
By selection of button-def Add you add-2pl-imperf element.

48. Cz:
Vybráním tlačítka Add (můžete) přidávat elementy.
Selecting-ins button-gen Add can-2pl add-imperf-inf elements

Or with the possibility modal:

49. En: *By selecting Add, {you/the user/someone} {can/may} add an element.*

50. Ru:
{Нажатием кнопки Add, Нажав кнопку Add,}
Selection-Instr button-Gen Add

Вы можете добавить элемент.
you can add element-Acc.

51. Bu:
{С избиране на, Като изберете} клавиша Add
By selection of button-def Add

{Вие можете да добавите/ потребителят може да добави елемент.
{you can-2pl to add-2pl / User-def can-3sg to add-3sg} element.

52. Cz:
Vybráním tlačítka Add {můžete / lze / je možné} přidávat elementy.
Selecting-ins button-gen Add {can-2pl / it is possible} add-imperf-inf elements

53. En:
By selecting Add, it is possible to add an element.

54. Ru:
{Нажатием кнопки Add, Нажав кнопку Add,} можно добавить элемент.
By selecting button-Acc Add possible add element-Acc.

55. Bu:
С избиране на клавиша Add може да добавите елемент.
By selection of button-def Add can to add-2pl element.

56. Cz:
Vybráním Add {je možné / lze} přidat element.
Selecting-ins Add {is possible} add-inf element-acc

In the last example for Slavic the Addressee is impossible, because this frame changes the modal meaning of possibility to the modal meaning of permission, thus becomes inadequate

for the situation. The fourth variant for the "same" content are two variants with medio passive possible in Slavic, but impossible in English:

57. En: *By selecting Add**,** an element adds.

58. Ru:
Нажатием кнопки Add, добавляется элемент.
Selecting-ins button-gen Add add-3sg-refl element

59. Bu:
С избиране на клавиша Add се добавя елемент.
By selection of button-def Add refl add-3sg element.

60. Cz:
Vybráním tlačítka Add se přidávají elementy.
Selecting-ins button-gen Add refl add-3pl elements
By selecting the Add button one adds elements.

The following variant in medio passive with the modal word is possible in Czech, Bulgarian and Russian. There are some specifics of the distribution of the actualisation features among the sentence elements, which we do not consider here.

61. Cz:
Vybráním tlačítka Add se mohou přidávat elementy.
Selecting-ins button-gen Add refl can-3pl add-inf elements

62. Ru:
Нажатием кнопки Add могут добавляться элементы.
Selecting-ins button-gen Add can-3pl add-inf-refl elements

63. Bu:
С избиране на клавиша Add може да се добавят елементи.
By selection of button-def Add can to refl add-3pl elements.

By selecting the Add button one can add elements.

The above examples have no additional information different from the procedural text. So rather than to be considered a functional descriptions they could be considered as a descriptive style of the procedural instruction. It is naturally true for the variants with modals, which can also be considered as a style for overviews (see Chapter 6), presupposing enumeration of homogeneous objects or actions. The representations with modals in descriptive procedural text are valid rather in the situation of alternative (to add the element if you consider it valid, and not add, if you have made mistakes in defining it). For the impossible procedural fragments as far as for the situation of function description the modal element contradicts to the prescriptive character of the procedural part and fixed object's function in functional descriptive text and in both cases seem to be clumsy and unnatural. It is strange to use the modal of possibility in the situation when it is the only one possibility:

64. En: *By choosing Multiline Style {you/the user/someone} {can/may} open the Multiline Style dialog box.*

65. Ru:
{Нажатием кнопки Multiline Style, Нажав кнопку Multiline Style,}
By selecting button-Acc Multiline Style

можно открыть диалоговое окно Multiline Style.
it-is-possible to-open dialog box Multiline Style.

66. Bu:
С избиране на клавиша Multiline Style {Вие/потребителят} може да
By selection of button-def Multiline Style {you/the user} can to

{отворите/отвори} диалоговия прозорец Multiline Style.
{open-2pl/open-3sg} dialogue box Multiline Style.

67. Cz:
Vybráním tlačítka Multiline Style {můžete / lze / je možné }
Selecting-ins button-gen Multiline Style {can-2pl / it is possible}

otevřít dialogové okno Multiline Style.
open-inf dialogue-acc box-acc Multiline Style.

The other semantic situation is the realisation by the user of a creative action factually having two actions: the creation of a "virtual" GUI object (symbol "l" or "a" in the following examples) and its trigger:

Entering a (when PLINE is / has been started) switches to Arc mode.
Entering l (when PLINE is / has been started) switches to Line mode.

For these texts the location of the created and then triggered GUI object is always presented by the prompt (command line). So its environment is different from the precedent type as it is not described by a location, but by the software product or its module (command). Or, in terms of A-box by an (explicit or implicit) action, as evidenced by the paraphrase "*Starting the PLINE command and then entering l switches to Line mod*". For example, consider the following IMD text fragment:

1. Start the PLINE command …
2. Specify the start point of the line segment.
3. Specify the endpoint of the line segment.
4. Enter a to switch to Arc mode.
   …
5. Specify the endpoint of the arc.
6. Enter l to return to Line mode.
   …

As we see from the examples this environment is naturally presented by a temporal subordinate clause.

The following example describes the function of two consecutive, explicit actions, whereas all previous examples have involved only a single action. We must consider only actions of a special class (see the formalisation). The second Sentence of the following example shows a sentence where the "creative" (see formalisation) function "specify" is attributed not to a functional object (a program, for example), but to the complex actions of the user. These situations cannot be considered as examples of the functional description text type in a proper sense, because of the functional object lacking. It can be considered as the other type of description texts, where we have a list of "creative" functions ("create", "specify" and so on) and the actions for their realisation. However, this is not the topic at the current stage of the project. Now we try to do only the first steps in modelling the type variety in instructional texts.

*Choosing Arc then choosing 3 Points from the Draw menu under DOS and UNIX start / starts (??) the ARC command.

*Entering endp and then selecting the line so the arc snaps to the endpoint of the line when the ARC command is / has been started specifies / specify the endpoint.

Another distinct type of descriptive texts or propositions we named "functional description". They are enough often presented in chunks marked in the real instructional texts by a **Related** connector. They describe functions of commands or of GUI-objects:

PLINE draws polyline line and arc segments that form a single object.
MLINE draws multiple parallel lines. …

But in our A-boxes we have only content oriented at procedural instructions, so we cannot take the corresponding text semantics directly from A-boxes. Planning the descriptive text we must perform some reasoning upon the A-box. A more commonly encountered kind of descriptive text relates Goals and GUI objects like commands, menu items, toolbar icons etc. Usually, the functions of commands are described, yet in our A-boxes we have only the functions of individual buttons. The method of modelling them is quite similar, though. We consider them on the base of the following examples of functional descriptive texts composed for IDM demonstrator texts.

OF functional-descriptive texts for the IDM text 1: page 47-48:

Choose the Multiline Styles button from the Object Properties toolbar or the Data menu under Windows opens the Multiline Styles dialog box.
The Multiline styles button from the Data menu under DOS and UNIX opens the Multiline Styles dialog box.
The Add button in the Element Properties dialog box adds an element.
The Color button in the Element Properties dialog box opens the Select Color dialog box.
The Linetype button in the Element Properties dialog box opens the Select Linetype dialog box.
The OK button in the Element Properties dialog box saves the style of the multiline element and exits the Element Properties dialog box.

OF functional-descriptive texts for the IDM text 2: page 46:

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

OF functional-descriptive texts for the IDM text 3: page 46:

3 Points button on the Arc flyout on (or from?) the Draw toolbar starts the ARC command.
3 Points button chosen after the Arc button from the Draw menu starts the ARC command.

Again the second sentence demonstrates the action environment for the GUI object, and is accordingly expressed by a temporal prepositional phrase.

5.3 Formalisation

First of all, modelling descriptive texts from a procedural text content representation (A-box) needs some reasoning upon the A-boxes and hence additional information about the DM concepts that are used. We propose some additional classes for concepts described in our DM.

We need a classification of user actions taking the following into consideration. Our user's actions that can be considered in this domain as usual functions of functional objects: GUI-objects (buttons) and Hardware-objects (keys). These object's functions are to "activate" and "deactivate" Displayed-objects. This class of objects is described in the DM as the widest class of objects that can be viewed and clicked on the screen. But here we mean objects as "dialog box", "menu" etc., the other objects being defined by narrower classes (Configured, Graphical, Data, GUI –objects, Option). Functions "activate" and "deactivate" are valid also to the classification of the actions with Software-tools (command, program, operating system). Other functions are related to the actions with Configured, Graphical and, Data, –objects. As a function of GUI-object the action can be "save", "add-to", "apply-property" etc., which we can call the "dispose" function. The last class of functions can be attributed to Software-tools. They are actions with Graphical-objects that the user can realise using this Software tool – "draw a line", "add dimensions to a drawing", "create hatched area" and so on. We call these instances of the "create" function.

In our DM we have the following concepts:

- START-TOOL / QUIT-TOOL (software-actee) – "activate/ deactivate";
- OPEN-SCREEN-OBJECT/CLOSE-SCREEN-OBJECT (displayed-actee) – "activate / deactivate";
- RETURN-TO – composition of "activate" and "deactivate" functions upon two objects;
- ADD-TO (configured-actee) – "dispose";

etc.

The class of our user's actions which are physical actions that can be considered in this domain as methods of activating the functional objects: GUI objects (buttons) and Hardware objects (keys). They are:

- PRESS (hardware-actee) – "activate".
- CHOOSE (displayed-actee) – "activate".

We do not consider now the situations of the creative action of user as they were considered in the discussion section, like in the example *Enter "l"*, which must be presented as composition of "create" function and "activate" function.

Modelling descriptive texts from procedural A-boxes goes by stages.

The first stage has the aim to identify those portions of the content that are to be included in the descriptive text. They are the fragments containing the two actions (GOALs) belonging to the described DM classes. We consider the initial A-box fragment using for illustration examples from the first and second IMD texts.

The fragment of A-box corresponding to the OF functional description is the following:

```
PROCEDURE
 GOAL (*): Start the PLINE command / Add elements to the style /
 METHODS: METHOD-LIST
           FIRST:
              METHOD
                   CONSTRAINT: Windows
                   SUBSTEPS: PROCEDURE-LIST
                                 FIRST:
                                    PROCEDURE:
                                          GOAL (**): Choose Polyline /
                                                     Choose Element .Properties
```

In this fragment we have marked the higher level GOAL slot by (*) and the lower level GOAL slot by (**) to make referring to the concepts in the structure description easier.

In this fragment of structure presented by configuration concepts the GOAL (*) slot structure is the following:

```
USER-ACTION: Start-tool "activate"/ Add-to "dispose"→ Start / Add
ACTEE: Software-actee / Configurated-actee → command / elements
```

The GOAL (**) slot structure is the following:

```
USER-ACTION: Choose "activate"/ Choose "activate"→ choose / choose
ACTEE: Displayed-actee / Displayed-actee → button / button
OPTIONS: GUI-actee
LOCATION: GUI-actee → the Polyline flyout /.
```

The USER-ACTION at the GOAL (*) belongs to the functions possible for “activate / deactivate” or “dispose” DM classes of actions with software-actee or configurated-actee.

The USER-ACTION at the GOAL (**) must belong to “activate” DM class of actions upon Displayed-actee, because the descriptive texts just describe the functions of functional objects according to the software product design. We consider here only the “activate” function, since we have not “deactivate” in our IMD demonstrator texts. We must follow the general rule that “create” functions are only possible for Software objects as attributed to them in the text structure while being in DM representation the user’ goals. “Create” functions can not be attributed to GUI objects, which are rather triggers.

At the second stage, when the fragment of A-box is found we need to match the structure like this into a text plan that arranges all its content. The text plan structure for the OF description text can be presented as follows:

```
OF-DESCRIPTION : The Element Properties button adds elements to the
style.
FUNCTIONAL-OBJECT : The Element Properties button
LOCATION : 0
OPTIONS : 0
FUNCTION : adds elements to the style.
```

Functional objects usually have just one function, and therefore the FUNCTION slot does not need to contain a list structure.

Next, let us discuss the mapping from A-box into the Text plan structure. The difference from the text plan for procedural text concerns the Functional-object and Function elements of the text plan.

The FUNCTIONAL-OBJECT corresponds to the ACTEE of an action in the GOAL (**) slot.

The FUNCTION is the PROCEDURE, for which it holds that the METHOD, in whose SUBSTEPS' PROCEDURE we found the FUNCTION-OBJECT, is a member of that PROCEDURE's METHODS's METHOD-LIST. Or in other words it is a PROCEDURE of the GOAL (*) slot.

Mapping:
OF-DESCRIPTIONS:

FUNCTIONAL-OBJECT ↔ Actee in GOAL (**) slot.
LOCATION ↔ Location (optional) specified with the action in GOAL (*).
OPTIONS ↔ Options specified with the action in GOAL (*).
FUNCTION ↔ Action in GOAL (*) slot.

So the FUNCTION is the Action that FUNCTIONAL-OBJECT "obtains" in the sense that the METHOD (under which PROCEDURE's GOAL (*) occurs) is a possible way of obtaining a PROCEDURE's GOAL (**), and as such specified under that PROCEDURE's METHODS.

In the case there is a CONSTRAINT specified with the METHOD, we can opt for introducing another text structure element, CONSTRAINT. What that CONSTRAINT maps to is obvious: It should get identified with the CONSTRAINT of the METHOD under which the PROCEDURE with “the lower GOAL” occurs.

A last comment concerns the process of the wording the content representation. Here we have difference from the procedural texts for English, where the word for the Option, expressed usually by Labelled-GUI-object “button” is regularly omitted in procedural texts, but presented in functional descriptions:

Choose OK to save the style of the multiline element vs. The OK button saves the style of the multiline element.

6. Overviews

6.1 Discussion of the text type

An overview is defined as running text, presenting a summary of the operations described in a technical manual or chapter containing a set of procedural texts. Normally the aim of an overview is to introduce a group of possible user tasks, so it can be found in the first chapter of a manual or in the introductions to each chapter. The content to be revealed in the overview is contained in the list of A-boxes, representing the set of procedures. A global overview has to combine in a running text the main goals of the described procedures, expressed by the task titles. This approach permits to include hypertext links in the overview texts, each task title pointing to the corresponding full procedure text, if properly marked.

We may have two different styles of overview genre, similar to the styles for procedural texts (the layout of the examples do not follow the overview running text layout for better readibility):

Personal style, with explicit 2nd person plural reference to the user:[8]

68. En:
The AGILE-AUTOCAD enables **you**
to create a multiline style,
to specify the properties of a multiline,
to draw a line and arc combination polyline,
to draw an arc by specifying three points,
and to define a boundary set in a complex drawing.

69. Bu:
Системата AGILE-AUTOCAD **Ви** позволява да създадете стил на
System-def AGILE-AUTOCAD **you** enable-3sg to create-**2pl** style of

мултилиния, да зададете характеристиките на мултилиния,
multiline, to define-**2pl** properties-def of multiline,

да начертаете полилиния, съставена от отсечки и дъги,
to draw-**2pl** polyline, composed of line-segments and arcs,

да начертаете дъга по три точки,
to draw-**2pl** arc by three points,

и да дефинирате множество граници в сложен чертеж.
and to define-**2pl** set boundaries in complex drawing.

[8] We mark the reference to the user visually by typesetting it in **bold** face.

70. Cz:
Systém AGILE-AutoCAD **Vám** umožňuje,
System AGILE-AutoCAD **You-dat** enables

abyste vytvořili styl multičáry,
so-that-would-**2pl** create-**2pl** style-acc multiline-gen

(abyste) určili vlastnosti multičáry,
(so-that-would-**2pl**) specify-**2pl** properties-acc multiline-gen

(abyste) nakreslili čáru složenou z přímek a olouku
(so-that-would-**2pl**) draw-2pl line composed from line-segments-gen and arc-gen,

(abyste) nakreslili oblouk určením tří bodů,
(so-that-would-**2pl**) draw-**2pl** arc-acc specifying-ins three-gen points-gen

a (abyste) definovali hraniční množinu
and (so-that-would-**2pl**) define-**2pl** boundary-acc set-acc

v komplexním výkrese.
in complex-loc drawing-loc

71. Ru:
Система AGILE-AUTOCAD позволяет **Вам** создавать
System-Nom AGILE-AUTOCAD enable-3sg **you-Dat** create-imprf-inf

стиль мультилинии,
style-Acc multiline-Gen,

задавать свойства мултилинии,
define-imprf -inf properties-Acc multiline-Gen,

рисовать полилинии,
draw-imprf-inf polyline-PL-Acc,

состоящие из отрезков прямых и дуг,
composed by segments-Gen line-Gen and arcs-Gen,

рисовать дугу по трем точкам,
draw-imprf-inf arc-Acc by three points-Dat,

определять набор границ в сложном рисунке.
define-imprf-inf set-Acc boundaries-Gen in complex drawing.

Impersonal style, with no reference to the user:[9]

72. En:
The AGILE-AUTOCAD enables
the creation of multiline style, the specification of the properties of a multiline, the drawing of a line and arc combination polyline, the drawing of an arc by specifying three points, the definition of a boundary set in a complex drawing.

73. Bu: reflexive passive voice
Системата AGILE-AUTOCAD позволява
System-def AGILE-AUTOCAD enable-3sg

да се създаде стил на мултилиния,
to *refl* create-3sg style of multiline,

да се зададат характеристиките на мултилиния,
to *refl* define-3pl properties-def of multiline

да се начертае полилиния, съставена от отсечки и дъги,
to *refl* draw-3sg polyline, composed by line-segments and arcs,

да се начертае дъга по три точки,
to *refl* draw-3sg arc by three points

да се дефинира множество граници в сложен чертеж.
to *refl* define-3sg set boundaries in complex drawing.

74. Cz: infinitive
Systém AGILE-AutoCAD umožňuje
System-nom AGILE-AutoCAD enables

vytvořit styl multičáry,
create-inf style-acc multiline-gen

určit vlastnosti multičáry,
specify-inf properties-acc multiline-gen

nakreslit čáru složenou z přímek a olouku
draw-2pl line composed from line-segments-gen and arc-gen,

nakreslit oblouk určením tří bodů,
draw-inf arc-acc specifying-ins three-gen points-gen

a definovat hraniční množinu v komplexním výkrese.
and define-inf boundary-acc set-acc in complex-loc drawing-loc

[9] We mark the reference to the user visually by typesetting it in **bold** face.

75. Ru: infinitive
Система AGILE-AUTOCAD позволяет создавать
System-Nom AGILE-AUTOCAD enable-3sg create-imprf-inf

стиль мультилинии, задавать свойства мултилинии,
style-Acc multiline-Gen, define-imprf -inf properties-Acc multiline-Gen,

рисовать полилинии,
draw-imprf-inf polyline-PL-Acc,

состоящие из отрезков прямых и дуг,
composed by segments-Gen line-Gen and arcs-Gen,

рисовать дугу по трем точкам,
draw-imprf-inf arc-Acc by three points-Dat,

определять набор границ в сложном рисунке.
define-imprf-inf set-Acc set boundaries-Gen in complex drawing.

It has to be mentioned that this style does not lead to a good English text, but Bulgarian, Czech and Russian text are considered natural.

The OVERVIEW and TOC content cannot be expressed by a single A-box. Therefore the presentation of an OVERVIEW or TOC may requires a new Domain model concept, which can express the content of several procedural A-Boxes in a single structure. We call this concept ACTIONS-GROUP, and it can be defined in the following manner.

```
(define-concept ACTIONS-GROUP
    ((HYPER-THEME :type USER-ACTION)
    (ACTION-LIST :type PROCEDURE-LIST)))
```

This concept is instantiated by what we will call an ACTIONS-GROUP-A-BOX, which by definition thus has the following slots:

- HYPER-THEME, expressing the content of the overview leading clause;
- ACTION-LIST, filled with a structure of type PROCEDURE-LIST, being a list of PROCEDUREs. Each PROCEDURE has a structure as defined in the Domain model.

The elements expressing the OVERVIEW content are the main goals /the topmost goals/ of the procedures in the list of A-boxes, created by the user. The ACTIONS-GROUP-A-BOX is using only the obligatory slot – GOAL. Each GOAL corresponds to the GOAL of the topmost PROCEDURE of an A-Box from the user's list of A-Boxes.

We can consider the following mapping between text structure elements and configuration concepts:

- OVERVIEW-INTRODUCTION ↔ HYPER-THEME
- OVERVIEW-TASKS ↔ ACTION-LIST
- OVERVIEW-TASK ↔ GOAL of PROCEDURE

The slot HYPER-THEME for the overview texts corresponds to OVERVIEW-INTRODUCTION in the text structure. Each procedure's GOAL is represented by an OVERVIEW-TASK.

The procedure which pre-processes list of A-boxes and produces ACTIONS-GROUP-A-BOX can be described in the following abstract manner:

```
(construct-aboxes-generalization (list-of-aboxes)
(make-new-actions-group-a-box)
(insert HYPER-THEME content)
(while (there is more aboxes left)
(insert in next-procedure
(get next-abox-procedure))))
```

The resulting ACTIONS-GROUP-A-Box is processed further by the text planner module using the formal descriptions in the next section.

In order to obtain more natural overview texts it is desirable to form them additionally. The subordinate clauses in the sentences of the overview text may be aggregated according to different criteria:

- **Random ordering**: The subordinate clauses follow the order in which the user has described the A-boxes for generating procedural texts. The text planning process may limit the number of the subordinated clauses in a single sentence. A reasonable sentence would describe not more than 3-4 tasks.
- **Aggregation according to the task actee**: The text planning process may group the tasks, described in an overview text, by their actee, i.e. to combine in separate sentences the tasks concerning the same actee. Such an aggregation permits the construction of clauses that describe possible actions executed on given object or on objects of the same class. The later possibility requires that the text planning process considers the ontology, presented by the domain model.
- **Aggregation according to the task action**: The text planning process may group the tasks, described in an overview text, by their action, i.e. to combine in separate sentences the tasks concerning the same action. Such an aggregation permits to construct clauses, describing the same actions on different objects of the CADCAM area.

The following example illustrates an aggregation according to the actee in the first sentence and an aggregation according to the action in the second sentence:

76. En:
    The AGILE-AUTOCAD system enables you to create a multiline style and to specify the properties of a multiline. You can also draw a line and arc combination polyline and an arc by specifying three points.

77. Bu:
Системата AGILE-AUTOCAD Ви позволява
System-def AGILE-AUTOCAD you enable-3sg

да създадете стил на мултилиния
to create-2pl style of multiline

и да зададете характеристиките на мултилиния,
and to define-2pl properties-def of multiline,

Вие можете също
You may-2pl also

да начертаете полилиния, съставена от отсечки и дъги,
to draw-2pl polyline, composed by line-segments and arcs,

и дъга по три точки.
and arc by three points.

78. Cz:
Systém AGILE-AutoCAD Vám umožňuje vytvořit styl multičáry,
System-nom AGILE-AutoCAD you-dat enables create-inf style-acc multiline-gen

a určit vlastnosti multičáry.
and specify-inf properties-acc multiline-gen

Můžete také nakreslit čáru složenou z přímek a olouku
Can-2pl also draw-2pl line composed from line-segments-gen and arc-gen

nebo nakreslit oblouk určením tří bodů,
or draw-inf arc-acc specifying-ins three-gen points-gen

79. Ru:
Система AGILE-AUTOCAD позволяет создавать
System-Nom AGILE-AUTOCAD enable-3sg create-imprf-inf

стиль мультилинии и задавать свойства мултилинии,
style-Acc multiline-Gen and define-imprf -inf properties-Acc multiline-Gen.

Вы также можете рисовать полилинии,
You also may-2pl draw-imprf-inf polyline-PL-Acc,

состоящие из отрезков прямых и дуг,
composed by segments-Gen line-Gen and arcs-Gen,

и дугу по трем точкам.
and arc-Acc by three points-Dat.

Different aggregation options may be specified by explicit user control, which will complicate the work with the system. The alternative solution is to make aggregation

decisions automatically by analysing the list of the tasks main goals looking for possible aggregations according to common actee and/or to common action.

6.2 Formalisation

The text structure elements are implemented by means of the following functional-grammar-like systems:

The following system inserts an OVERVIEW-TASK or an OVERVIEW-INTRODUCTION into the text plan.

```
(SYSTEM OVERVIEW-ELEMENTS
    :INPUTS OVERVIEW-GENRE
    ((OVERVIEW-DOCUMENT
          (INSERT OVERVIEW-INTRODUCTION)
          (ORDERATFRONT OVERVIEW-INTRODUCTION))
    (OVERVIEW-TASK
          (INSERT OVERVIEW-TASK))))

(CHOOSER OVERVIEW-ELEMENTS-CHOOSER
    ((CHOOSE OVERVIEW-DOCUMENT
          if HYPER-THEME needs to be realized)
          else
                (CHOOSE OVERVIEW-TASK)))
```

The next system goes recursively through the ACTION-LIST, which has as its value a PROCEDURE-LIST containing PROCEDURES. For each head in the ACTION-LIST we insert a text structure element OVERVIEW-TASKS in the text plan, alike INSTRUCTION-TASKS in the text plans for full procedural instructional texts. By preselecting OVERVIEW-TASKS as OVERVIEW-TASK we ensure that on the next traversal through the systems, we enter the system dealing with OVERVIEW-TASK (OVERVIEW-ELEMENTS). Upon entering that system an OVERVIEW-TASK will be inserted in the text plan, and identified with the GOAL of the PROCEDURE that constitutes the head of the (currently observed part of the) ACTION-LIST.

```
(SYSTEM OVERVIEW-TASK-TYPE
    :INPUTS OVERVIEW-DOCUMENT, OVERVIEW-TASK
    ((MORE-OVERVIEW-TASKS
                (INSERT OVERVIEW-TASKS)
                (ORDERATEND OVERVIEW-TASKS)
                (PRESELECT OVERVIEW-TASKS OVERVIEW-TASK))
    (NO-MORE-OVERVIEW-TASKS)))

    (CHOOSER OVERVIEW-TASK-TYPE-CHOOSER
          ((ask (longer-list-q)
                (more (CHOOSE MORE-TOC-TASKS))
                (no-more (CHOOSE NO-MORE-TOC-TASKS)))))
```

Finally, the next two systems impose the proper constraints on the realisation of OVERVIEW-INTRODUCTION and OVERVIEW-TASK. These constraints will be

included in the SPLs (for the respective elements) that will be generated by the sentence planner.

```
(SYSTEM OVERVIEW-INTRODUCTION-REALIZATION-STYLE
      :INPUTS OVERVIEW-DOCUMENT
      ((INTRO-PERSONAL
            (INSERT INTRO-PERSONAL)
            (REALIZE-WITH INTRO-PERSONAL
      (:SPEECHACT IMPERATIVE)))
      ((INTRO-IMPERSONAL
            (INSERT INTRO-IMPERSONAL)
            (REALIZE-WITH INTRO-IMPERSONAL
      (:PREFER-MENTION-AGENT-Q WITHHOLD))))))

(SYSTEM OVERVIEW-TASK-REALIZATION
      :INPUTS OVERVIEW-TASK
      ((OVERVIEW-TASK-PERSONAL
            (INSERT OVERVIEW-TASK-PERSONAL)
            (REALIZE-WITH OVERVIEW-TASK-PERSONAL
                  :SPEECHACT IMPERATIVE)))
      (OVERVIEW-TASK-IMPERSONAL
            (INSERT OVERVIEW-TASK-IMPERSONAL)
            (REALIZE-WITH OVERVIEW-TASK-IMPERSONAL
                  (:PREFER-MENTION-AGENT-Q WITHHOLD)))))
```

Personal style, with explicit 2nd person plural reference to the user:[10]

80. En:
    The AGILE-AUTOCAD enables **you**
    to create a multiline style,
    to specify the properties of a multiline,
    to draw a line and arc combination polyline,
    to draw an arc by specifying three points,
    and to define a boundary set in a complex drawing.

[10] We mark the reference to the user visually by typesetting it in **bold** face.

81. Bu:
Системата AGILE-AUTOCAD **Ви** позволява да създадете стил на
System-def AGILE-AUTOCAD **you** enable-3sg to create-**2pl** style of

мултилиния, да зададете характеристиките на мултилиния,
multiline, to define-**2pl** properties-def of multiline,

да начертаете полилиния, съставена от отсечки и дъги,
to draw-**2pl** polyline, composed of line-segments and arcs,

да начертаете дъга по три точки,
to draw-**2pl** arc by three points,

и да дефинирате множество граници в сложен чертеж.
and to define-**2pl** set boundaries in complex drawing.

82. Cz:
Systém AGILE-AutoCAD **Vám** umožňuje,
System AGILE-AutoCAD **You-dat** enables

abyste vytvořili styl multičáry,
so-that-would-**2pl** create-**2pl** style-acc multiline-gen

(abyste) určili vlastnosti multičáry,
(so-that-would-**2pl**) specify-**2pl** properties-acc multiline-gen

(abyste) nakreslili čáru složenou z přímek a olouku
(so-that-would-**2pl**) draw-2pl line composed from line-segments-gen and arc-gen,

(abyste) nakreslili oblouk určením tří bodů,
(so-that-would-**2pl**) draw-**2pl** arc-acc specifying-ins three-gen points-gen

a (abyste) definovali hraniční množinu
and (so-that-would-**2pl**) define-**2pl** boundary-acc set-acc

v komplexním výkrese.
in complex-loc drawing-loc

83. Ru:
Система AGILE-AUTOCAD позволяет **Вам** создавать
System-Nom AGILE-AUTOCAD enable-3sg **you-Dat** create-imprf-inf

стиль мультилинии,
style-Acc multiline-Gen,

задавать свойства мултилинии,
define-imprf -inf properties-Acc multiline-Gen,

рисовать полилинии,
draw-imprf-inf polyline-PL-Acc,

состоящие из отрезков прямых и дуг,
composed by segments-Gen line-Gen and arcs-Gen,

рисовать дугу по трем точкам,
draw-imprf-inf arc-Acc by three points-Dat,

определять набор границ в сложном рисунке.
define-imprf-inf set-Acc boundaries-Gen in complex drawing.

7. Table of Contents

7.1 Discussion of the text type

This genre covers a presentation of the chapters/paragraphs titles of the texts, generated during an AGILE session. These texts are supposed to form a consistent part of a technical manual, describing the operation of a CAD-CAM system. During an AGILE session the technical author specifies the content of the texts through the authoring interface, constructing a list of semantic descriptions - A-boxes. The Table-of-Contents has to present a numbered list of the headings of different texts, generated from a set of A-boxes, as discussed in the other chapters of the present deliverable. This basic requirement leads to the following structure:

TABLE OF CONTENTS

1. Overview
2. Procedures
   - 2.1…….
   - 2.2…….
   - ……..
3. Command descriptions
   - 3.1 …….
   - 3.2 …….
   - ……..
4. Quick reference
   - 4.1 ……
   - 4.2 ……
   - ……

The sections of the Table-of-Contents are numbered lists of the Task titles, possibly with hyperlinks to the corresponding full texts. As the most common texts in AGILE are procedural texts, the Task titles present the main goals of the “Full procedural Instructions” texts. The “Quick reference” section present the “Basic step” texts and the “Command descriptions” section presents “Functional descriptive texts”.

The TOC content cannot be expressed by single A-box. Therefore a new Domain model concept is necessary, which can express the content of several procedural A-Boxes in a single structure. The concept name is CHAPTER-CONTENT and is described in the chapter on overviews (Chapter 9).

We can consider the following mapping between text structure elements and configuration concepts:

- TOC-TITLE ↔ INITIAL-EVENT
- TOC-TASKS ↔ ACTION-LIST
- TOC-TASK ↔ GOAL of PROCEDURE

The INITIAL-EVENT can be just a label like ‘TABLE OF CONTENTS’.

The text planner for planning the Table-of-contents text genre will need a procedure that pre-processes a list of A-boxes and produces an ACTIONS-GROUP-A-BOX. The procedure and this type of A-box are described in the chapter on overviews. The resulting ACTIONS-GROUP-A-BOX is processed further by the text planner module using, the formal descriptions given in the next section.

For better text layout one could imagine the following HTML mapping applied to the Table-of-content text:

Text Structure Element	Layout	HTML Tags
TOC-TITLE	Heading level *N*	<H*N*> … </H*N*>
TOC-TASKS	Ordered list	<OL> … </OL>
TOC-TASK	List Item	<LI>

7.2 Formalisation

The text structure elements are implemented by means of the following functional-grammar-like systems.

The following system inserts a TOC-TASK or a TOC-TITLE into the text plan.

```
(SYSTEM TOC-ELEMENTS
      :INPUTS TOC-GENRE
      ((TOC-DOCUMENT
            (INSERT TOC-TITLE)
            (ORDERATFRONT TOC-TITLE))
      (TOC-TASK
            (INSERT TOC-TASK))))

(CHOOSER TOC-ELEMENTS-CHOOSER
((CHOOSE TOC-DOCUMENT
      if INITIAL-EVENT needs to be realized)
      else
            (CHOOSE TOC-TASK)))
```

The next system goes recursively through the ACTION-LIST, which has as its value a PROCEDURE-LIST containing PROCEDURES. For each head in the ACTION-LIST we insert a text structure element TOC-TASKS in the text plan, alike INSTRUCTION-TASKS in the text plans for full procedural instructional texts. By preselecting TOC-TASKS as TOC-TASK we ensure that on the next traversal through the systems, we enter the system dealing with TOC-TASK (TOC-ELEMENTS). Upon entering that system a TOC-TASK will be inserted in the text plan, and identified with the GOAL of the PROCEDURE that constitutes the head of the (currently observed part of the) ACTION-LIST.

```
(SYSTEM TOC-TASK-TYPE
      :INPUTS TOC-DOCUMENT, TOC-TASK
      ((MORE-TOC-TASKS
            (INSERT TOC-TASKS)
            (ORDERATEND TOC-TASKS)
            (PRESELECT TOC-TASKS TOC-TASK))
      (NO-MORE-TOC-TASKS)))

(CHOOSER TOC-TASK-TYPE-CHOOSER
      ((ask (longer-list-q)
            (more (CHOOSE MORE-TOC-TASKS))
            (no-more (CHOOSE NO-MORE-TOC-TASKS)))))
```

Finally, the next two systems - gates, since they do not need any underlying choosers, impose the proper constraints on the realisation of TOC-TITLE and TOC-TASK. These constraints will be included in the SPLs (for the respective elements) that will be generated by the sentence planner.

```
(GATE TOC-TITLE-REALIZATION
      :INPUTS TOC-DOCUMENT
      ((TOC-TEMPLATE
            (INSERT TOC-TEMPLATE)
            (REALIZE-WITH TOC-TEMPLATE
                  (insert Template "TABLE OF CONTENTS"))))

(GATE TOC-TASK-REALIZATION
      :INPUTS TOC-TASK
      (NOMINALIZED-TOC-TASK
            (INSERT NOMINALIZED-TOC-TASK)
            (REALIZE-WITH NOMINALIZED-TOC-TASK
                  (:EXIST-SPEECH-ACT-Q NOSPEECHACT
                        :AGENT-MENTION-Q WITHHOLD))))
```

8. Closing remarks

In this document we discussed various extensions to the Text Structuring Module (TSM) as it was specified and implemented in WP 5.2 (cf. TEXS2, TEXM2). We started the deliverable by extending the coverage in depth, by specifying additional stylistic variations for, for example, overall text style (Chapter 2).

In subsequent chapters (Chapters 3-7) we extended the coverage of the TSM in breadth. We provided a more elaborate picture of discourse modelling in Chapter 3, in a way that will enable us to develop "text grammars" in a manner similar to the way we have been elaborating the lexicogrammars. In Chapters 4 through 7 we discussed genres that are different from the full instructions type generated by the intermediate demonstrator - summaries, descriptions, overviews and TOCs were described, and we specified how they could be generated on the basis of the A-boxes specifying the content for full instructions.

The generation of function descriptions, overviews and tables of content amounts to generating "meta-texts" on the basis of multiple input content specifications. Applications generating such "meta-texts" are is fairly unique in current NLG, which means that our work in this are is quite original.

During the implementation phase of WP 5.3 we will extend the TSM as specified here. Furthermore, we believe that the following (implementational) issues should be addressed as well at that point:

- *Style sheets*, as groupings of coherent settings for different options of stylistic variation. The idea of style sheets was already discussed in TEXM2, but the increase in possible stylistic variation specified in this deliverable enables us to elaborate this notion in more detail.
- *Semanticization of concepts*, being the proper realisation of concepts. Currently, there are two ways in which the realisation of a concept can be specified. The first way is a more or less isomorphic mapping from concept to word (i.e. modulo morphological cases). The second way is by inclusion of more intricate SPL code in the SPL for the sentence in which the concept is to be realised. This case is used for concepts that need to be realised as multi-word expressions. For example, the concept DIALOGUE-BOX needs to be realised by two separate words in the Slavic languages under consideration. These two ways suffice for bringing about the natural realisation of most concepts. However, there are cases that seem to warrant a third way in which the proper realisation of particular concepts should be specified. These cases concern the realisation of concepts that have particular slots (due to their definition in the DM and inheritance from the UM) that should not be straightforwardly realised as the same slots, but as different slots (e.g. SOURCE versus LOCATION).

References

[CORP, AGILE 3] Anthony Hartley, Danail Dochev, Nevena Gromova, Kamenka Staykova, Alla Bémová, Alexandr Rosen, Jiří Trojánek, Lena Sokolova. *Tagging and analysis of instructional texts in the software domain*. AGILE deliverable 3.1 deliverable, June 1998. (Deliverable comprises CORP-Cz, CORP-Bu, CORP-Ru).

[TEXS1+TEXM1, AGILE 5.1] J. Bateman, A. Hartley, I. Kruijff- Korbayová, D. Dochev, N. Gromova, J. Hana, S. Sharoff, L. Sokolova, *Generation of simple text structures in Bulgarian, Czech and Russian*. AGILE deliverable 5.1, June 1998. (Deliverable comprises TEXS1-Cz, TEXS1-Bu, TEXS1-Ru, TEXM1.)

[TEXS2, AGILE 5.2] I. Kruijff-Korbayová, G.J.M. Kruijff, J. Bateman, D. Dochev, N. Gromova, A. Hartley, E. Teich, S. Sharoff, L. Sokolova, and K. Staykova, *Specification of elaborated text structures*. AGILE deliverable 5.2, April 1999. (Deliverable comprises TEXS2-Cz, TEXS2-Bu, TEXS2-Ru)

[TEXM2, AGILE 5.2] Geert-Jan Kruijff, Ivana Kruijff-Korbayova and John Bateman. The Text Structuring Module for the Intermediate Prototype. July 1999

[Asher 1993] Nicholas Asher, Referring to Abstract Objects in Discourse. Kluwer Academic Publishers. 1993.

[Delin et al. 1996] Delin, J. A. Hartley and D. Scott 'Towards a contrastive pragmatics: syntactic choice in English and French instructions' *Language Sciences* 18 (3—4): 897—931. 1996.

[Cohen 1987] Robin Cohen, "Analyzing the Structure of Argumentative Discourse." *Computational Linguistics*. 13(1-2):11-24. 1987.

[Gardent 1994] Claire Gardent, Discourse Multiple Dependencies. CLAUS Report No. 45. University of Saarland, Saarbruecken. September 1994.

[Grosz and Sidner, 1986] Barbara J. Grosz and Candace L. Sidner, "Attention, intention, and the structure of discourse." *Computation Linguistics,* **12** (3): 175-204, 1986.

[Hartley and Paris 1996] Anthony Hartley and Cécile Paris, "Two Sources of Control over the Generation of Software Instructions" *Proceedings of ACL96* 192-196, 1996.

[Hobbs 1979] Jerry R. Hobbs, "Coherence and coreference". *Cognitive Science.* 3(1):67-90. 1979.

[Hovy, 1993] Eduard H. Hovy, "Automated Discourse Generation Using Discourse Structure Relations". *Artificial Intelligence* **63**: 341-385, 1993. Reprinted in Barbara J. Grosz and Fernando C.N. Pereira (eds.), *Natural Language Processing*, The MIT Press: Cambridge MS, 1994

[Knott 1996] Alistair Knott, *A Data-Driven Methodology for Motivating a Set of Coherence Relations*. PhD thesis, University of Edinburgh, Edinburgh. 1995.

[Knott and Mellish 1996] Alistair Knott and Chris Mellish, "A Feature-based Account of the Relations Signalled by Sentences and Clause Connectives." *Language and Speech*. 39(2-3):143-183, 1996.

[Lagerwerf 1998] Luuk Lagerwerf, *Causal Connectives Have Presuppositions: Effects on Ceherence and Discourse Structure*. PhD thesis, Katholieke Universiteit Brabant, Tilburg.. Netherlands Graduate School of Linguistics. Holland Academic Graphics. 1998.

[Mann and Thompson, 1988] William Mann and Susan Thompson, "Rhetorical Structure Theory: towards a Functional Theory of Text Organisation" *Text*, 8(2):243-281, 1987.

[Martin 1992] James Martin *English Text: Systems and Structure.* Benjamins: Amsterdam. 1992.

[Meteer, 1991] M.W. Meeter, "Bridging the generation gap between text planning and linguistic realisation." *Computational Intelligence,* **7**(4):296-304, 1991.

[Moore and Paris, 1993] Johanna D. Moore and Cécile L. Paris, "Planning text for advisory dialogues: Capturing intentional and rhetorical information." *Computational Linguistics*, **19** (4): 651-694, 1993

[Moore and Pollack, 1992] Johanna D. Moore and Martha E. Pollack, "A Problem for RST: The Need for Multi-Level Discourse Analysis". *Computational Linguistics*, **18** (4): 537-544, 1992.

[Polanyi 1988] Livia Polanyi, A formal model of the structure of discourse." *Journal of Pragmatics.*12: 601--638. 1988.

[Radev & McKeown 1998] Dragomir R. Radev and Kathleen R. McKeown, "Generating Natural Language Summaries from Multiple On-Line Sources", Computational Linguistics (Special Issue on Natural Language Generation), Volume 24, Number 3, September 1998, pages 469-500.

[Reape & Mellish 1999] Mike Reape and Chris Mellish. "Just what *is* aggregation anyway?" In the Proceedings of ENLG'99, Tolouse. 1999.

[Reiter & Dale, 1997] Ehud Reiter and Robert Dale, "Building Applied Natural Language Generation Systems". *Natural Language Engineering*, **1** (1), 1997.

[Reiter, Mellish & Levine, 1995] Ehud Reiter, Chris Mellish and John Levine, "Automatic Generation of Technical Documentation". *Applied Artificial Intelligence*, **9**, 1995

[Schiffrin 1987] Deborah Shiffrin, *Discourse Markers*. Cambridge University Press, second edition. 1987.

[Stone & Webber 1998] Matthew Stone and Bonnie Webber. "Textual Economy through Close Coupling of Syntax and Semantics." International Workshop on Natural Language Generation, Niagara-on-the-Lake, Canada, August 1998.

[Thompson 1985] Thompson, S.A. 1985 'Grammar and written discourse: initial vs final purpose clauses in English' *Text: an interdisciplinary journal for the study of text* 5 (1-2): 55—84. 1985.

[Webber 1988] Bonnie Webber, Discourse Deixis: reference to discourse segments. In: Proceedings of the Association for Computational Linguistics, pp. 113-122. 1988.

[van der Linden & Martin 1995] van der Linden, K. and J. Martin. 'Expressing local rhetorical relations in instructional text: a case study of the purpose relation' *Computational Linguistics* 21 (1): 29--57. 1995

[Webber et al. 1999a] Bonnie Webber, Alistair Knott, Matthew Stone and Aravind Joshi. "Discourse Relations: A Structural and Presuppositional Account using Lexicalised TAG." 1999 Meeting of the Association for Computational Linguistics, College Park MD, June 1999.

[Webber et al. 1999b] Bonnie Webber, Alistair Knott and Aravind Joshi. Multiple Discourse Connectives in a Lexicalized Grammar for Discourse. Third International Workshop on Computational Semantics, Tilburg, The Netherlands, January 1999.

9. Appendices

9.1 Brief descriptions of inquiries available in the current TSM

Below the inquiries of the CADCAM-INSTRUCTIONS region are described, as actually used. The implementations (CODE) of these inquiries are placed in the <tsm2000.lisp> file. The headings can be used as formal representation. In CADCAM-INSTRUCTIONS.choosers it is easy to see which chooser relies on which inquiries.

9.1.1 Descriptions of general inquiries

9.1.1.1 agile-abox-root-id-code (ONUS)

Assumes: nothing
Returns: The content under the current generation-root ONUS.

9.1.1.2 agile-abox-root-q-code (ONUS)

Assumes: nothing
Returns: Declares the generation-root, resets *text-element-associations* and returns the value '*document* if we are at the root of the A-box and '*component* otherwise. *text-element-associations* contains constraints associated to a particular text-element, notably those obtained by REALIZE-WITH.

9.1.1.3 get-abox-slot (SLOT)

Assumes: a root ROOT (determined in the context)
Returns: the value of the slot SLOT under ROOT, when that slot is filled.

9.1.1.4 get-abox-slot-sloppy (SLOT)

Assumes: a root ROOT (determined in the context)
Returns: the value of SLOT under ROOT, provided such a slot exists under ROOT and is filled.

9.1.2 Description of TASK-related inquiries

9.1.2.1 first-instruction-id-code (ROOT)

Assumes: Root
Returns: If there is a FIRST slot under the ROOT, checked using **get-abox-slot-sloppy**, then the value of that FIRST slot is returned, otherwise ROOT is returned. ROOT is assumed to be a TASK/PROCEDURE's METHODS's METHOD-LIST.

9.1.2.2 multiple-methods-id-code (ROOT)

Assumes: Root
Returns: Returns the value of the METHODS slot under ROOT using **get-abox-slot**.

9.1.2.3 rest-instruction-id-code (ROOT)

Assumes: Root
Returns: If there is a REST slot under the ROOT, checked using **get-abox-slot**, then the value of that REST slot is returned. ROOT is assumed to be a TASK/PROCEDURE's METHODS's METHOD-LIST.
Comments: Compare with the above inquiry, **first-instruction-id-code**.

9.1.2.4 task-id-code (ROOT)

Assumes: nothing
Returns: The value of the GOAL slot of ROOT by means of **get-abox-slot**.

9.1.2.5 task-sideeffect-id-code (ROOT)

Assumes: Root to be set in the context
Returns: The value of the slot SIDE-EFFECT under the current ROOT, using **get-abox-slot**. We might be dealing with a list - if so, the ROOT is redirected to the FIRST element of that list.

9.1.2.6 task-side-effect-q-code (ROOT)

Assumes: ROOT to be set in the context, possibly pointing to a PROCEDURE-LIST.
Returns: Checks whether there is a filled slot SIDE-EFFECT, if so then return *'with-effect* otherwise *'without-effect*. The code checks whether we are dealing with a list or with a procedure (e.g. the top-most procedure to which TASK-TITLE corresponds) by checking (using **get-abox-slot-sloppy**) whether there is a FIRST slot. If so, then the root is redirected to the FIRST of the list.

9.1.2.7 task-title-id-code (ROOT)

Assumes: nothing
Returns: The value of the GOAL slot of ROOT by means of **get-abox-slot**.

9.1.2.8 task-type-q-code (ROOT)

Assumes: ROOT to be set in the context
Returns: If there is a filled slot METHODS (checked using **get-abox-slot-sloppy**) then return *'with-instruct* otherwise *'without-instruct*.
Comments: Note the use of the sloppy version of get-abox-slot: it may well be the case that the inquiry looks at a PROCEDURE-LIST rather than a PROCEDURE, when called. However, changing the code such that ROOT is redirected to the FIRST procedure in a list *when* we are dealing with a list seems to make no observable difference.

9.1.3 Descriptions of INSTRUCTION-related inquiries

9.1.3.1 instruction-constraint-id-code (ROOT)

Assumes: Root to be set in the context
Returns: The value of the slot CONSTRAINT under the current ROOT, using **get-abox-slot**.

9.1.3.2 instruction-constraint-q-code (ROOT)

Assumes: ROOT to be set in the context
Returns: If there is a filled slot CONSTRAINT under the current root then return *'with-constraint* else return *'without-constraint*. Uses **get-abox-slot**.

9.1.3.3 instruction-precondition-id-code (ROOT)

Assumes: Root to be set in the context
Returns: The value of the slot PRECONDITION under the current ROOT, using **get-abox-slot**.

9.1.3.4 instruction-precondition-q-code (ROOT)

Assumes: ROOT to be set in the context.
Returns: If there is a filled slot PRECONDITION under the current root then return *'with-precondition* else return *'without-precondition*. Uses **get-abox-slot**.

9.1.3.5 instruction-substeps-id-code (ROOT)

Assumes: Root to be set in the context
Returns: The value of the slot SUBSTEPS under the current ROOT, using **get-abox-slot**.

9.1.3.6 instruction-type-q-code (ROOT)

Assumes: nothing
Returns: Defaults to '*complexinstruct*.

9.2 Target texts for Full Procedural Instructions

We present the versions of the texts differing in degree of aggregation and in sequence marking, to be considered as targets for the Final Prototype. What we call "Baseline" for each text is supposed to correspond very closely to the A-box. Aggregation is done only for most simplex steps. Vertical lists are used rather than running text. Sequences are marked by numbering or conjunctives, as discussed in the deliverable. Indentation reflects hierarchy in A-box. There are (a,b) variants regarding explicit realisation of side-effects (this gives two versions of each text, by taking consistently all the (a) or all the (b) variants; no (a-b) combinations).

The subsequent versions of the texts exercise aggregation and explicit sequence marking as discussed in the deliverable. There are two version for each text, one labelled as "more aggregated" and the other as "running text".

As for the variation between personal and impersonal style, we only include the personal style, using imperative mood. The impersonal style, using reflexive passive, is of course also available, and is to be generated, but we do not include it in this appendix, for the sake of space.

9.2.1 ENGLISH

9.2.1.1 Based on IMD Text 1: pages 47-48

9.2.1.1.1 Baseline Text 1

Versions (i) and (ii) reflect different possibilities of anaphoric reference.

To create a multiline style

First open the *Multiline Styles* dialog box using one of these methods:[11]

Windows: Choose *Multiline Style* from the *Object Properties* toolbar or choose *Multiline Style* from the *Data* menu.

DOS and UNIX: Choose *Multiline Style* from the *Data* menu.

Now add elements to the style.

(a)

1. Choose *Element Properties.*
2. In the *Element Properties* dialog box, enter the offset of the multiline elements.
3. Define an element.

First select *Add* in the *Element Properties* dialog box to add an/one element.

(i) Now specify the element's color. First choose *Color* in the *Element Properties* dialog box to. Then select the element's color from the *Select Color* dialog box.[12]

(ii) Now specify the element's color. First choose *Color* in the *Element Properties* dialog box. Then select a color from the *Select Color* dialog box.[13][14]

(i) Finally, specify the element's line type. First choose *Linetype* in the *Element Properties* dialog box. Then select the element's linetype from the *Select Linetype* dialog box.

(ii) Finally, specify the element's line type. First choose *Linetype* in the *Element Properties* dialog box. Then select a line type from the *Select Linetype* dialog box.

Repeat these steps to define another element.

4. Choose OK to save the properties of the multiline elements.

(b)

1. Choose *Element Properties.* The *Element Properties* dialogue box appears.

2.Enter the offset of the multiline elements.

3. Define an element.

First select *Add* to add an/one element.

(i) Now specify the element's color. First choose *Color.* The *Select Color* dialog box appears. Select a color. The *Select Color* dialog box disappears[15]

(ii) Now specify the element's color. First choose *Color.* The *Select Color* dialog box appears. Select the element's color. The *Select Color* dialog box disappears[16]

(i) Finally, specify the element's line type. Choose *Linetype*. The *Select Linetype* dialog box appears. Select a line type. The *Select Linetype* dialog box disappears.

---

[11] Here, we are treating this piece not as a precondition, but as a normal step. The modelling of the text is sort of funny anyway, because in this case we treat opening a dialogue box as a Goal (corresp. To the user's intention), while in other cases dialogue boxes appear (open) as side-effects of the user's actions.

[12] We added a Goal here, in order to reflect a more hierarchical structure of the A-box.

[13] We added a Goal here, in order to reflect a more hierarchical structure of the A-box.

[14] Note *a/the color.* It appears that both are possible here, depending on the exact meaning the speaker would have in mind. A color = any color from the color-chart, vs. the color = the value for the element's color.

[15] We added the disappearing piece for explicitness' sake, as if that was in the A-box.

[16] We added the disappearing piece for explicitness' sake, as if that was in the A-box.

(ii) Finally, specify the element's line type. Choose *Linetype*. The *Select Linetype* dialog box appears. Select the element's linetype. The *Select Linetype* dialog box disappears.

Repeat these steps to define another element.

4. Choose OK to save the properties of the multiline elements. The Element Properties dialog box disappears.[17]

This is a version with the following schematic structure:

Define an element.

1.(First) …

2. (Now)…

3. (Finally)…

4. () Repeat these steps to define another element.

Another possibility is to employ the following schematic structure:

Define elements by repeating the following steps.

1.(First) …

2. (Now)…

3. (Finally)…

If there would be only one step to repeat, singular number should be used, and running text rather than list.

9.2.1.1.2 More aggregated Version of Text 1

Given the discussion of aggregation in the deliverable, we can observe that more aggregation is possible only for the steps in version (a), while the presence of explicit side-effects in version (b) actually prevents more aggregation. So we only show version (a) here. We include possible additional explicit sequence markers in brackets.

To create a multiline style

First open the *Multiline Styles* dialog box using one of these methods:[18]

Windows: Choose *Multiline Style* from the *Object Properties* toolbar or from the *Data* menu.

DOS and UNIX: Choose *Multiline Style* from the *Data* menu.

Now add elements to the style.

(a)

1. Choose *Element Properties* and (then) enter the offset of the multiline elements in the *Element Properties* dialog box.

2. Define an element.

First select *Add* in the *Element Properties* dialog box to add an/one element.

Now specify the element's color by (first) choosing *Color* in the *Element Properties* dialog box and (then) selecting the (element's) color from the *Select Color* dialog box.[19]

---

[17] Originally: *Choose OK to save the properties of the multiline element and to exit the Element Properties dialog box.* --but, these are two Goals with one "shared" Method. Would have to be modelled by co-identification in the A-box, and then aggregation would have to take care of it…Currently the A-box contains a side-effect corresponding to the closing =disappearing of the dialogue box (it is not exit, because side-effect should not be a user-action, right?).

[18] Here, I am treating this piece not as a precondition, but as a normal step. The modelling of the text is sort of funny anyway, because in this case we treat opening a dialogue box as a Goal (corresp. To the user's intention), while in other cases dialogue boxes appear (open) as side-effects of the user's actions.

[19] I added a Goal here, in order to reflect a more hierarchical structure of the A-box.

Finally, specify the element's line type by (first) choosing *Linetype* in the *Element Properties* dialog box and (then) selecting the (element's) linetype from the *Select Linetype* dialog box.

Repeat these steps to define another element.

3. Choose OK to save the properties of the multiline elements.

9.2.1.1.3 Running version of Text 1

For the more aggregated version of Text 1 it also makes sense to consider its realisation using running text for the third level of embedding, rather than a list. We show the text for version (a) below. Running text is not a viable option for version (b), again because of the explicit side-effects.

To create a multiline style

First open the *Multiline Styles* dialog box using one of these methods:[20]

Windows: Choose *Multiline Style* from the *Object Properties* toolbar or from the *Data* menu.

DOS and UNIX: Choose *Multiline Style* from the *Data* menu.

Now add elements to the style.

(a)

1. Choose *Element Properties* and (then) enter the offset of the multiline elements in the *Element Properties* dialog box.

2. To define an element, first select *Add* in the *Element Properties* dialog box to add an/one element. Now specify the element's color by (first) choosing *Color* in the *Element Properties* dialog box and (then) selecting the (element's) color from the *Select Color* dialog box. Finally, specify the element's line type by (first) choosing *Linetype* in the *Element Properties* dialog box and (then) selecting the (element's) linetype from the *Select Linetype* dialog box. Repeat these steps to define another element.

3. Choose OK to save the properties of the multiline elements.

[20] Here, I am treating this piece not as a precondition, but as a normal step. The modelling of the text is sort of funny anyway, because in this case we treat opening a dialogue box as a Goal (corresp. To the user's intention), while in other cases dialogue boxes appear (open) as side-effects of the user's actions.

9.2.1.2 Based on IMD Text 2: page 46

9.2.1.2.1 Baseline Text 2

To draw a line and arc combination polyline

First draw a line segment.

1.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*.

2.Specify the start point of the line segment.

3.Specify the endpoint of the line segment.

Then draw an arc segment.

1.(a) Switch to *Arc mode*. First enter *a*. Then select *OK* in the *Arc mode confirmation* dialog box.

(b) Switch to *Arc mode*. Enter *a*. The *Arc mode confirmation* dialog box appears. Select *OK*.

2.Specify the endpoint of the arc.

Then draw another line segment.

1.(a) Return to *Line mode*. First enter *l*. Then select *OK* in the *Line mode confirmation* dialog box.

(b) Return to *Line mode*. Enter *l*. The *Line mode confirmation* dialog box appears. Select *OK*.

2.Enter the distance of the line in relation to the endpoint of the arc.

3.Enter the angle of the line in relation to the endpoint of the arc.

Finally, press *Return* to end the polyline.

9.2.1.2.2 More Aggregated Version of Text 2

Again, this version only makes sense for version (a), because the explicit side-effects in version (b) block aggregation.

To draw a line and arc combination polyline

First draw a line segment.

1.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*.

2.Specify the start point and the endpoint of the line segment.

Then draw an arc segment.

(a) 1. Switch to *Arc mode* by (first) entering *a* and (then) selecting *OK* in the *Arc mode confirmation* dialog box.

2. Specify the endpoint of the arc.

Then draw another line segment.

(a) 1. Return to *Line mode* by (first) entering *l* and (then) selecting *OK* in the *Line mode confirmation* dialog box.

2. Enter the distance and the angle of the line in relation to the endpoint of the arc.

Finally, press *Return* to end the polyline.

9.2.1.2.3 Running version of Text 2

Again, the more aggregated version of the text lends itself to realisation by a running text at the more deeply embedded level. Note that here, we use numbering at the highest level.

To draw a line and arc combination polyline

1. Draw a line segment.

   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*.

   Then specify the start point and the endpoint of the line segment.

2. Draw an arc segment. First switch to *Arc mode* by entering *a* and selecting *OK* in the *Arc mode confirmation* dialog box. Then specify the endpoint of the arc.

3. Draw another line segment. First return to *Line mode* by entering *l* and selecting *OK* in the *Line mode confirmation* dialog box. Then enter the distance and the angle of the line in relation to the endpoint of the arc.

4. Press *Return* to end the polyline.

9.2.1.3 Based on IMD Text 3: page 58

9.2.1.3.1 Baseline Version of Text 3

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 / its endpoint}.

2. Specify the second point (of the arc). First enter *poi*. Then select a point. The arc snaps to {the point / it}.

3. Specify the endpoint (of the arc).

9.2.1.3.2 More Aggregated Version of Text 3

To draw an arc by specifying three points.

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) by entering *endp* and selecting a line so that the arc snaps to the endpoint of the line.

2. (i) Specify the second point (of the arc) by entering *poi* and selecting a point so that the arc snaps to {the point / it}.

   (ii) Specify the second point (of the arc) by entering *poi* and selecting a point (for the arc) to snap to.

3. Specify the endpoint (of the arc).

9.2.1.3.3 Running Version of Text 3

To draw an arc by specifying three points.

1. 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*.

2. (Specify three points of the arc.) First 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. Then specify the second point (of the arc) by entering *poi* and selecting a point {so that the arc snaps to the point / so that the arc snaps to it / (for the arc) to snap to}. Finally, specify the endpoint (of the arc).

9.2.1.4 Based on IMD Text 4: pages 48/9

9.2.1.4.1 Baseline Version of Text 4

To define a multiline style

(a)

First choose *Multiline Style* from the *Data* menu.

Then specify the properties of the multiline.

1. In the *Multiline Styles* dialog box, choose *Multiline Properties*.
2. In the *Multiline Properties* dialog box, select *Display Joints* to display a line at the vertices of the multiline.
3. (In the *Multiline Properties* dialog box,) Define the startpoint of the multiline. Select a line under *Caps* or select an arc Under *Caps.*
4. (In the *Multiline Properties* dialog box,) Define the endpoint of the multiline. Select a line under *Caps* or select an arc under *Caps*.
5. (In the *Multiline Properties* dialog box,) Enter an angle.
6. (In the *Multiline Properties* dialog box,) Define background fill color. First select *On* Under *Fill* to display a background color. Then choose *Color*. Finally, select the background fill color from the *Select Color* dialog box.
7. (In the *Multiline Properties* dialog box,) Choose *OK* to return to the *Multiline Styles* dialog box.

(b)

First choose *Multiline Style* from the *Data* menu. The *Multiline Styles* dialog box appears.

Then specify the properties of the multiline.

1. Choose *Multiline Properties*. The *Multiline Properties* dialog box appears.
2. Select *Display Joints* to display a line at the vertices of the multiline.
3. Define the startpoint of the multiline. Select a line under *Caps* or select an arc Under *Caps.*
4. Define the endpoint of the multiline. Select a line under *Caps* or select an arc under *Caps*.
5. Enter an angle.
6. Define background fill color. First select *On* Under *Fill* to display a background color. Then choose *Color*. The *Select Color* dialog box appears. Select the background fill color.
7. Choose *OK* to return to the *Multiline Styles* dialog box. The *Multiline Properties* dialog box disappears.

(a,b)

Then save the style.

1. Under *Name*, enter {the name of the style / a name}.
2. Under *Description*, enter {the description (of the style) / a description}.
3. Select *Add* to add the style to the drawing.
4. Select *Save* to save the style to a file.

(a) Finally, choose *OK.*

(b) Finally, choose *OK*. The dialog box *Multiline Style* disappears.

9.2.1.4.2 More Aggregated Version of Text 4

To define a multiline style

(a)

First choose *Multiline Style* from the *Data* menu.

Then specify the properties of the multiline.

1. In the *Multiline Styles* dialog box, choose *Multiline Properties*.
2. In the *Multiline Properties* dialog box, select *Display Joints* to display a line at the vertices of the multiline.
3. (In the *Multiline Properties* dialog box,) Define the startpoint and the endpoint of the multiline by selecting a line or an arc under *Caps.*
4. (In the *Multiline Properties* dialog box,) Enter an angle.
5. (In the *Multiline Properties* dialog box,) Define background fill color by first selecting *On* Under *Fill* to display a background color, then choosing *Color*, and finally selecting the background fill color from the *Select Color* dialog box.
6. (In the *Multiline Properties* dialog box,) Choose *OK* to return to the *Multiline Styles* dialog box.

(b)

First choose *Multiline Style* from the *Data* menu. The *Multiline Styles* dialog box appears.

Then specify the properties of the multiline.

1. Choose *Multiline Properties*. The *Multiline Properties* dialog box appears.
2. Select *Display Joints* to display a line at the vertices of the multiline.
3. Select a line or an arc Under *Caps* to define the startpoint of the multiline and to define the endpoint of the multiline.
4. Enter an angle.
5. Define background fill color by first selecting *On* Under *Fill* to display a background color and then choosing *Color*. The *Select Color* dialog box appears. Select the background fill color.
6. Choose *OK* to return to the *Multiline Styles* dialog box, so that the *Multiline Properties* dialog box disappears.

(a,b)

Then save the style.

1. Under *Name*, enter {the name of the style / a name}.
2. Under *Description*, enter {the description (of the style) / a description}.
3. Select *Add* to add the style to the drawing.
4. Select *Save* to save the style to a file.

(a) Finally, choose *OK.*

(b) Finally, choose *OK,* so that the dialog box *Multiline Style* disappears.

9.2.1.4.3 Semi-Running version of Text 4

We have swapped numbering, and tried a running version. However, the specification of the multiline properties has so many substeps, that the realisation by a running text does not really seem viable. Therefore, we include here a list using numbers at the top level and linguistic markers at the embedded level.

To define a multiline style

(a)

1. Choose *Multiline Style* from the *Data* menu.
2. Specify the properties of the multiline.

First choose *Multiline Properties* in the *Multiline Styles* dialog box.

Then select *Display Joints* in the *Multiline Properties* dialog box, to display a line at the vertices of the multiline.

Then define the startpoint and endpoint of the multiline by selecting a line or an arc Under *Caps,* and (then) enter an angle.

Then define the background fill color by first selecting *On* Under *Fill* to display a background color, then choosing *Color*, and finally selecting the background fill color from the *Select Color* dialog box.

Finally, choose *OK* to return to the *Multiline Styles* dialog box.

(b)

1. Choose *Multiline Style* from the *Data* menu. The *Multiline Styles* dialog box appears.
2. Specify the properties of the multiline. First choose *Multiline Properties*. The *Multiline Properties* dialog box appears.

Then select *Display Joints* to display a line at the vertices of the multiline.

Then select a line or an arc Under *Caps* to define the startpoint of the multiline and to define the endpoint of the multiline.

Then enter an angle.

Then define the background fill color by first selecting *On* Under *Fill* to display a background color and then choosing *Color,* so that the *Select Color* dialog box appears. Select the background fill color.

Finally, choose *OK* to return to the *Multiline Styles* dialog box, so that the *Multiline Properties* dialog box disappears.

(a,b)

3. Save the style. First, enter {the name of the style / a name} under *Name*, then enter {the description (of the style) / a description}under *Description*, then select *Add* to add the style to the drawing, and finally select *Save* to save the style to a file.

4. (a) Finally, choose *OK.*

(b) Finally, choose *OK,* so that the dialog box *Multiline Style* disappears.

9.2.1.5 Based on IMD Text 5: page 75

9.2.1.5.1 Baseline version of Text 5

To define a boundary set in a complex drawing

1. Open the *Boundary Hatch* dialog box using one of these methods:

   Windows: From the *Hatch* flyout on the *Draw* toolbar, choose *Hatch*.

   DOS and UNIX: From the *Draw* menu, choose *Hatch*

2. (a) Under *Boundary* choose *Advanced*.

   (b) Under *Boundary* choose *Advanced*. The *Advanced Options* dialog box appears.

3. (a) In the *Advanced Options* dialog box, choose *Make New Boundary Set*.

   (b) Choose *Make New Boundary Set*.

4. Define the boundary set. First specify the corner points at the *Select Objects* prompt. Then Press *Return*.

5. (a) In the *Advanced Options* dialog box, choose *OK*.

   (b) In the *Advanced Options* dialog box, choose *OK*. The *Advanced Options* dialog box disappears.

6. In the *Boundary Hatch* dialog box, choose *Pick Points*.

7. Specify the internal point.

8. Press *Return*.

9. In the *Boundary Hatch* dialog box, choose *Apply* to apply the hatch.

9.2.1.5.2 More Aggregated Version of Text 5

To define a boundary set in a complex drawing

1. Open the *Boundary Hatch* dialog box using one of these methods:

   Windows: From the *Hatch* flyout on the *Draw* toolbar, choose *Hatch*.

   DOS and UNIX: From the *Draw* menu, choose *Hatch*

(a)

2. Under *Boundary* choose *Advanced* and (then) in the *Advanced Options* dialog box, choose *Make New Boundary Set*.

3. To define the boundary set, (first) specify the corner points at the *Select Objects* prompt and (then) press *Return*.

4. In the *Advanced Options* dialog box, choose *OK*.

5. In the *Boundary Hatch* dialog box, choose *Pick Points*, (then) specify the internal point, (then) press return, and (finally) choose *Apply* to apply the hatch.

(b)

2. Under *Boundary* choose *Advanced*, so that the *Advanced Options* dialog box appears, then choose *Make New Boundary Set*.

3. To define the boundary set, (first) specify the corner points at the *Select Objects* prompt and (then) Press *Return*.

4. In the *Advanced Options* dialog box, choose *OK*, so that the *Advanced Options* dialog box disappears.

5. In the *Boundary Hatch* dialog box, choose *Pick Points,* (then) specify the internal point, (then) press return and finally choose *Apply* to apply the hatch.

9.2.1.5.3 Running Version of Text 5

To define a boundary set in a complex drawing

First open the *Boundary Hatch* dialog box using one of these methods:

Windows: From the *Hatch* flyout on the *Draw* toolbar, choose *Hatch.*

DOS and UNIX: From the *Draw* menu, choose *Hatch*

(a)

Then choose *Advanced* under *Boundary.* In the *Advanced Options* dialog box, choose *Make New Boundary Set*. Then define the boundary set by (first) specifying the corner points at the *Select Objects* prompt and (then) pressing *Return*. Then choose *OK* in the *Advanced Options* dialog box. Finally, in the *Boundary Hatch* dialog box, choose *Pick Points*, (then) specify the internal point, (then) press return, and (finally) choose *Apply* to apply the hatch.

(b)

Then choose *Advanced* under *Boundary*, so that the *Advanced Options* dialog box appears, Choose *Make New Boundary Set*. Then define the boundary set by (first) specifying the corner points at the *Select Objects* prompt and (then) pressing *Return.* Then choose *OK* in the *Advanced Options* dialog box. The *Advanced Options* dialog box disappears. Finally, ín the *Boundary Hatch* dialog box, choose *Pick Points,* (then) specify the internal point, (then) press return and finally choose *Apply* to apply the hatch.

9.2.2 CZECH

9.2.2.1 Based on IMD Text 1: pages 47-48

9.2.2.1.1 Baseline Text 1

Vytvoření stylu multičáry

Nejdříve otevřete dialogové okno *Styly multičáry* jedním z následujících způsobů:

Windows: Vyberte *Styly multičáry* z nástrojového panelu *Vlastnosti objektů* nebo vyberte *Styl multičáry* z menu *Data.*

DOS and UNIX: Vyberte *Styl multičáry* z menu *Data.*

Nyní přidejte elementy ke stylu.

(a)

1. Vyberte *Vlastnosti prvků*.

2. V dialogovém okně *Vlastnosti prvků* zadejte {posunutí/vzájemnou vzdálenost} elementů multičáry.

3. Definujte element.

Nejdříve vyberte *Přidat* v dialogovém panelu *Vlastnosti prvků,* abyste přidali (jeden) element.

Nyní v dialogovém panelu *Vlastnosti prvků* vyberte *Barva*, abyste určili barvu tohoto elementu. Potom barvu (elementu) vyberte v dialogovém panelu *Výběr barvy.*

Nakonec v dialogovém okně *Vlastnosti prvků* vyberte *Typ čáry,* abyste určili typ čáry tohoto elementu. Potom typ čáry (elementu) vyberte v dialogovém panelu *Výběr typů čar.*

Opakujte tyto kroky, abyste definovali další elementy.

4. Vyberte OK, abyste uložili vlastnosti elementů multičáry.

(b)

1. Vyberte *Vlastnosti prvků*. Objeví se dialogový panel *Vlastnosti prvků.*

2. Zadejte {posunutí/vzájemnou vzdálenost} elementů multičáry.

3. Definujte element.

Nejdříve vyberte *Přidat,* abyste přidali (jeden) element.

Nyní vyberte *Barva*, abyste určili barvu tohoto elementu. Objeví se dialogový panel *Výběr barvy*. Vyberte barvu (elementu). Dialogový panel *Výběr barvy* zmizí

Nakonec vyberte *Typ čáry,* abyste určili typ čáry tohoto elementu. Objeví se v dialogový panel *Výběr typů čar.* Vyberte typ čáry (elementu). Dialogový panel *Výběr typů čar* zmizí.

Opakujte tyto kroky, abyste definovali další elementy.

4. Vyberte OK, abyste uložili vlastnosti elementů multičáry. Dialogový panel *Vlastnosti prvků* zmizí.

This is a version with the following schematic structure:

Definujte element.

1.(Nejdříve) …

2. (Nyní)…

3. (Nakonec)…

4. () Opakujte tyto kroky, abyste definovali další elementy.

Another possibility is to employ the following schematic structure:

Definujte elementy opakováním následujících kroků.

1.(Nejdříve) …

2. (Nyní)…

3. (Nakonec)…

If there would be only one step to repeat, singular number should be used, and running text rather than list.

9.2.2.1.2 More aggregated Text 1

Vytvoření stylu multičáry

Nejdříve otevřete dialogové okno *Styly multičáry* jedním z následujících způsobů:

Windows: Vyberte *Styly multičáry* z nástrojového panelu *Vlastnosti objektů* nebo z menu *Data.*

DOS and UNIX: Vyberte *Styl multičáry* z menu *Data.*

Nyní přidejte elementy ke stylu.

(a)

1. Vyberte *Vlastnosti prvků* a potom v dialogovém okně *Vlastnosti prvků* zadejte {posunutí/vzájemnou vzdálenost} elementů multičáry.

2. Definujte element.

Nejdříve vyberte *Přidat* v dialogovém panelu *Vlastnosti prvků,* abyste přidali (jeden) element.

Nyní v dialogovém panelu *Vlastnosti prvků* vyberte *Barva*, abyste určili barvu tohoto elementu, a potom barvu (elementu) vyberte v dialogovém panelu *Výběr barvy.*

Nakonec v dialogovém okně *Vlastnosti prvků* vyberte *Typ čáry,* abyste určili typ čáry tohoto elementu, a potom typ čáry (elementu) vyberte v dialogovém panelu *Výběr typů čar.*

Opakujte tyto kroky, abyste definovali další elementy.

4. Vyberte OK, abyste uložili vlastnosti elementů multičáry.

9.2.2.1.3 Running version of Text 1

Vytvoření stylu multičáry

Nejdříve otevřete dialogové okno *Styly multičáry* jedním z následujících způsobů:

Windows: Vyberte *Styly multičáry* z nástrojového panelu *Vlastnosti objektů* nebo z menu *Data.*

DOS and UNIX: Vyberte *Styl multičáry* z menu *Data.*

Nyní přidejte elementy ke stylu.

(a)

1. Vyberte *Vlastnosti prvků* a potom v dialogovém okně *Vlastnosti prvků* zadejte {posunutí/vzájemnou vzdálenost} elementů multičáry.

2. Definujte element.

Nejdříve vyberte *Přidat* v dialogovém panelu *Vlastnosti prvků,* abyste přidali (jeden) element. Nyní v dialogovém panelu *Vlastnosti prvků* vyberte *Barva*, abyste určili barvu tohoto elementu, a potom barvu (elementu) vyberte v dialogovém panelu *Výběr barvy.* Nakonec v dialogovém okně *Vlastnosti prvků* vyberte *Typ čáry,* abyste určili typ čáry tohoto elementu, a potom typ čáry (elementu) vyberte v dialogovém panelu *Výběr typů čar.* Opakujte tyto kroky, abyste definovali další elementy.

4. Vyberte OK, abyste uložili vlastnosti elementů multičáry.

9.2.2.2 Based on IMD Text 2: page 46

9.2.2.2.1 Baseline Text 2

Nakreslení křivky (složené) z přímek a oblouku

Nejdříve nakreslete rovný segment.

1.Spusťte příkaz *Křivka* jedním z následujících způsobů:

Windows: Z plovoucího ikonového menu *Křivka* na nástrojovém panelu *Kresli* vyberte *Křivka.*

DOS and UNIX: Z menu *Kresli* vyberte *Křivka.*

2.Určete počáteční bod rovného segmentu.

3.Určete koncový bod rovného segmentu.

Potom nakreslete oblouk.

1.(a) Přepněte do režimu *Kreslení oblouků*. Nejdříve zadejte *o*. Potom vyberte *OK* v dialogovém panelu *Potvrzení režimu kreslení oblouků*.

(b) Přepněte do režimu *Kreslení oblouků*. Zadejte *o*. Objeví se dialogový panel *Potvrzení režimu kreslení oblouků.* Vyberte *OK.*

2.Určete koncový bod oblouku.

Potom nakreslete další rovný segment.

1.(a) Vraťte se do režimu *Kreslení úseček*. Nejdříve zadejte *l*. Potom vyberte *OK* v dialogovém panelu *Potvrzení režimu kreslení úseček.*

(b) Vraťte se do režimu *Kreslení úseček*. Zadejte *l*. Objeví se dialogový panel *Potvrzení režimu kreslení úseček.* Vyberte *OK.*

2.Zadejte vzdálenost úsečky od koncového bodu oblouku.

3.Zadejte úhel úsečky ve vztahu ke koncovému bodu oblouku.

Nakonec stiskněte *Enter*, abyste křivku ukončili.

9.2.2.2.2 More Aggregated Version of Text 2

Nakreslení křivky (složené) z přímek a oblouku

Nejdříve nakreslete rovný segment.

1.Spusťte příkaz *Křivka* jedním z následujících způsobů:

Windows: Z plovoucího ikonového menu *Křivka* na nástrojovém panelu *Kresli* vyberte *Křivka.*

DOS and UNIX: Z menu *Kresli* vyberte *Křivka.*

2.Určete počáteční a koncový bod rovného segmentu.

Potom nakreslete oblouk.

(a) 1. Přepněte do režimu *Kreslení oblouků* zadáním *o* vybráním *OK* v dialogovém panelu *Potvrzení režimu kreslení oblouků*.

2. Určete koncový bod oblouku.

Potom nakreslete další rovný segment.

(a) 1.Vraťte se do režimu *Kreslení úseček* zadáním *l* a vybráním *OK* v dialogovém panelu *Potvrzení režimu kreslení úseček.*

2. Zadejte vzdálenost a úhel úsečky ve vztahu ke koncovému bodu oblouku.

Nakonec stiskněte *Enter*, abyste křivku ukončili.

9.2.2.2.3 Running version of Text 2

Nakreslení křivky (složené) z přímek a oblouku

1.Nakreslete rovný segment.

Nejdříve spusťte příkaz *Křivka* jedním z následujících způsobů:

Windows: Z plovoucího ikonového menu *Křivka* na nástrojovém panelu *Kresli* vyberte *Křivka.*

DOS and UNIX: Z menu *Kresli* vyberte *Křivka.*

Potom určete počáteční a koncový bod rovného segmentu.

2. Nakreslete oblouk. Nejdříve přepněte do režimu *Kreslení oblouků* zadáním *o* vybráním *OK* v dialogovém panelu *Potvrzení režimu kreslení oblouků*. Potom určete koncový bod oblouku.

3. Nakreslete další rovný segment. Nejdříve se vraťte do režimu *Kreslení úseček* zadáním *l* a vybráním *OK* v dialogovém panelu *Potvrzení režimu kreslení úseček.* Potom zadejte vzdálenost a úhel úsečky ve vztahu ke koncovému bodu oblouku.

4. Stiskněte *Enter*, abyste křivku ukončili.

9.2.2.3 Based on IMD Text 3: page 58

9.2.2.3.1 Baseline Version of Text 3

Nakreslení oblouku

Nejdříve spusťte příkaz *Oblouk* jedním z následujících způsobů:

Windows: Z plovoucího ikonového menu *Oblouk* na nástrojovém panelu *Kresli* vyberte *3 body*.

DOS and UNIX: Z menu *Kresli* vyberte *Oblouk*. Potom vyberte *3 body*.

Nyní určete tři body oblouku.

1. Určete počáteční bod (oblouku). Nejdříve zadejte *kon.* Potom vyberte čáru. Oblouk se přichytí {ke koncovému bodu čáry / kjejímu koncovému bodu}.
2. Určete druhý bod (oblouku). Nejdříve zadejte *bod.* Potom vyberte bod. Oblouk se {k bodu / k němu} přichytí.
3. Určete koncový bod (oblouku).

9.2.2.3.2 More Aggregated Version of Text 3

Nakreslení oblouku určením (jeho) tří bodů

Nejdříve spusťte příkaz *Oblouk* jedním z následujících způsobů:

Windows: Z plovoucího ikonového menu *Oblouk* na nástrojovém panelu *Kresli* vyberte *3 body*.

DOS and UNIX: Z menu *Kresli* vyberte *Oblouk*. Potom vyberte *3 body*.

Nyní určete tři body oblouku.

1. Určete počáteční bod (oblouku) zadáním *kon* a vybráním čáry, takže oblouk se přichytí {ke koncovému bodu čáry / kjejímu koncovému bodu}.
2. Určete druhý bod (oblouku) zadáním *bod* a vybráním bodu, takže oblouk se {k bodu / k němu} přichytí.
3. Určete koncový bod (oblouku).

9.2.2.3.3 Running Version of Text 3

Nakreslení oblouku určením (jeho) tří bodů

1. Spusťte příkaz *Oblouk* jedním z následujících způsobů:

   Windows: Z plovoucího ikonového menu *Oblouk* na nástrojovém panelu *Kresli* vyberte *3 body*.

   DOS and UNIX: Z menu *Kresli* vyberte *Oblouk*. Potom vyberte *3 body*.

2. (Určete tři body oblouku.) Nejdříve určete počáteční bod (oblouku) zadáním *kon* a vybráním čáry, takže oblouk se přichytí {ke koncovému bodu čáry / kjejímu koncovému bodu}. Potom určete druhý bod (oblouku) zadáním *bod* a vybráním bodu, takže oblouk se {k bodu / k němu} přichytí. Nakonec určete koncový bod (oblouku).

9.2.2.4 Based on IMD Text 4: pages 48/9

9.2.2.4.1 Baseline Version of Text 4

Definování stylu multičáry

(a)

Nejdříve vyberte *Styl multičáry* z menu *Data*.

Potom určete vlastnosti multičáry.

1. V dialogovém panelu *Styly multičáry* vyberte *Vlastnosti multičáry*.
2. V dialogovém panelu *Vlastnosti multičáry* vyberte *Zobraz klouby,* abyste zobrazili čáru/y ve vrcholech multičáry.
3. (V dialogovém panelu *Vlastnosti multičáry*) Definujte počáteční bod multičáry. Pod *Zakončení* Vyberte úsečku nebo pod *Zakončení* vyberte oblouk
4. (V dialogovém panelu *Vlastnosti multičáry*) Definujte koncová bod multičáry. Pod *Zakončení* Vyberte úsečku nebo pod *Zakončení* vyberte oblouk
5. (V dialogovém panelu *Vlastnosti multičáry*) Zadejte úhel.
6. (V dialogovém panelu *Vlastnosti multičáry*) Definujte barvu pro vyplnění pozadí. Nejdříve vyberte *Ano* pod *Vyplnění,* abyste barvu pozadí zobrazili. Potom vyberte *Barva*. Nakonec vyberte barvu pro vyplnění pozadí z dialogového panelu *Výběr barvy*.
7. (V dialogovém panelu *Vlastnosti multičáry*) Vyberte *OK,* abyste se vrátili do dialogového panelu *Styly multičár.*

(b)

Nejdříve vyberte *Styl multičáry* z menu *Data*. Objeví se dialogový panel *Styly multičár*.

Potom určete vlastnosti multičáry.

1. Vyberte *Vlastnosti multičáry*. Objeví se dialogový panel *Vlastnosti multičáry*.
2. Vyberte *Zobraz klouby,* abyste zobrazili čáru/y ve vrcholech multičáry.
3. Definujte počáteční bod multičáry. Pod *Zakončení* Vyberte úsečku nebo pod *Zakončení* vyberte oblouk
4. Definujte koncová bod multičáry. Pod *Zakončení* Vyberte úsečku nebo pod *Zakončení* vyberte oblouk
5. Zadejte úhel.
6. Definujte barvu pro vyplnění pozadí. Nejdříve vyberte *Ano* pod *Vyplnění,* abyste barvu pozadí zobrazili. Potom vyberte *Barva*. Objeví se dialogový panel *Výběr barvy.* Vyberte barvu pro vyplnění pozadí.
7. Vyberte *OK,* abyste se vrátili do dialogového panelu *Styly multičár.* Dialogový panel *Vlastnosti multičáry* zmizí.

(a,b)

Potom styl uložte.

1. Pod *Jméno* zadejte {název stylu / název}.
2. Pod *Popis* zadejte {popis (stylu) / jeho popis}.
3. Vyberte *Přidat,* abyste styl přidali k výkresu.
4. Vyberte *Uložit,* abyste styl uložili do souboru.

(a) Nakonec vyberte *OK*.

(b) Nakonec vyberte *OK*. Dialogový panel *Styly multičár* zmizí.

9.2.2.4.2 More Aggregated Version of Text 4

Definování stylu multičáry

(a)

Nejdříve vyberte *Styl multičáry* z menu *Data*.

Potom určete vlastnosti multičáry.

1. V dialogovém panelu *Styly multičáry* vyberte *Vlastnosti multičáry*.
2. V dialogovém panelu *Vlastnosti multičáry* vyberte *Zobraz klouby,* abyste zobrazili čáru/y ve vrcholech multičáry.
3. (V dialogovém panelu *Vlastnosti multičáry*) Definujte počáteční a koncový bod multičáry vybráním úsečky nebo oblouku pod *Zakončení*.
5. (V dialogovém panelu *Vlastnosti multičáry*) Zadejte úhel.
6. (V dialogovém panelu *Vlastnosti multičáry*) Abyste definovali barvu pro vyplnění pozadí, nejdříve vyberte *Ano* pod *Vyplnění,* abyste barvu pozadí zobrazili, potom vyberte *Barva* a nakonec z dialogového panelu *Výběr barvy* vyberte barvu pro vyplnění pozadí.
7. (V dialogovém panelu *Vlastnosti multičáry*) Vyberte *OK,* abyste se vrátili do dialogového panelu *Styly multičár.*

(b)

Nejdříve vyberte *Styl multičáry* z menu *Data*. Objeví se dialogový panel *Styly multičár*.

Potom určete vlastnosti multičáry.

1. Vyberte *Vlastnosti multičáry*. Objeví se dialogový panel *Vlastnosti multičáry*.
2. Vyberte *Zobraz klouby,* abyste zobrazili čáru/y ve vrcholech multičáry.
3. Definujte počáteční a koncový bod multičáry vybráním úsečky nebo oblouku pod *Zakončení.*
5. Zadejte úhel.
6. Abyste definovali barvu pro vyplnění pozadí, nejdříve vyberte *Ano* pod *Vyplnění,* abyste barvu pozadí zobrazili, potom vyberte *Barva*. Objeví se dialogový panel *Výběr barvy.* Vyberte barvu pro vyplnění pozadí.
7. Vyberte *OK,* abyste se vrátili do dialogového panelu *Styly multičár,* takže dialogový panel *Vlastnosti multičáry* zmizí.

(a,b)

Potom styl uložte.

1. Pod *Jméno* zadejte {název stylu / název}.
2. Pod *Popis* zadejte {popis (stylu) / jeho popis}.
3. Vyberte *Přidat,* abyste styl přidali k výkresu.
4. Vyberte *Uložit,* abyste styl uložili do souboru.

(a) Nakonec vyberte *OK*.

(b) Nakonec vyberte *OK*, takže dialogový panel *Styly multičár* zmizí.

9.2.2.4.3 Semi-Running version of Text 4

Definování stylu multičáry

(a)

1. Vyberte *Styl multičáry* z menu *Data.*

2. Určete vlastnosti multičáry.

   Nejdříve vyberte *Vlastnosti multičáry* v dialogovém panelu *Styly multičáry.*

   Potom vyberte *Zobraz klouby* vdialogovém panelu *Vlastnosti multičáry,* abyste zobrazili čáru/y ve vrcholech multičáry.

   Potom (v dialogovém panelu *Vlastnosti multičáry*) definujte počáteční a koncový bod multičáry vybráním úsečky nebo oblouku pod *Zakončení* a (potom) zadejte úhel.

? Potom (V dialogovém panelu *Vlastnosti multičáry*) abyste definovali barvu pro vyplnění pozadí, nejdříve vyberte *Ano* pod *Vyplnění,* abyste barvu pozadí zobrazili, potom vyberte *Barva* a nakonec z dialogového panelu *Výběr barvy* vyberte barvu pro vyplnění pozadí.

   Nakonec (V dialogovém panelu *Vlastnosti multičáry*) vyberte *OK,* abyste se vrátili do dialogového panelu *Styly multičár.*

(b)

1. vyberte *Styl multičáry* z menu *Data.* Objeví se dialogový panel *Styly multičár.*

2. určete vlastnosti multičáry.

   Nejdříve vyberte *Vlastnosti multičáry*. Objeví se dialogový panel *Vlastnosti multičáry.*

   Potom vyberte *Zobraz klouby,* abyste zobrazili čáru/y ve vrcholech multičáry.

   Potom definujte počáteční a koncový bod multičáry vybráním úsečky nebo oblouku pod *Zakončení* a (potom) zadejte úhel.

   Potom, abyste definovali barvu pro vyplnění pozadí, nejdříve vyberte *Ano* pod *Vyplnění,* abyste barvu pozadí zobrazili, potom vyberte *Barva.* Objeví se dialogový panel *Výběr barvy.* Vyberte barvu pro vyplnění pozadí.

   Potom vyberte *OK,* abyste se vrátili do dialogového panelu *Styly multičár,* takže dialogový panel *Vlastnosti multičáry* zmizí.

(a,b)

3. Styl uložte. Nejdříve pod *Jméno* zadejte {název stylu / název}, potom pod *Popis* zadejte {popis (stylu) / jeho popis}, potom vyberte *Přidat,* abyste styl přidali k výkresu a nakonec vyberte *Uložit,* abyste styl uložili do souboru.

(a) Nakonec vyberte *OK.*

(b) Nakonec vyberte *OK*, takže dialogový panel *Styly multičár* zmizí.

9.2.2.5 Based on IMD Text 5: page 75

9.2.2.5.1 Baseline version of Text 5

Definování hraniční množiny v komplexním výkrese

1. Otevřete dialogový panel *Hraniční množina* jedním z následujících způsobů:

   Windows: Z plovoucího ikonového menu *Šrafy*na nástrojovém panelu *Kresli* vyberte *Šrafy*.

   DOS a UNIX: Z menu *Kresli* vyberte *Šrafy*.

2. (a) Pod *Hranice šrafování* vyberte *Pokročilé*.

   (b) Pod *Hranice šrafování* vyberte *Pokročilé*. Objeví se dialogový panel *Pokročilé možnosti*.

3. (a) V dialogovém panelu *Pokročilé možnosti* vyberte *Tvořit novou hraniční množinu.*

   (b) Vyberte *Tvořit novou hraniční množinu.*

4. Definujte hraniční množinu. Nejdříve při výzvě *Výběr objektů* určete rohové body hraniční množiny. Potom stiskněte *Enter*.

5. (a) V dialogovém panelu *Pokročilé možnosti* vyberte *OK*.

   (b) V dialogovém panelu *Pokročilé možnosti* vyberte *OK*. Dialogový panel *Pokročilé možnosti* zmizí.

6. V dialogovém panelu *Hraniční šrafování* vyberte *Výběr objektů*.

7. Určete vnitřní bod.

8. Stiskněte *Enter*.

9. V dialogovém panelu *Hraniční šrafování* vyberte *Aplikovat,* abyste vyšrafovali plochu.

9.2.2.5.2 More Aggregated Version of Text 5

Definování hraniční množiny v komplexním výkrese

1. Otevřete dialogový panel *Hraniční množina* jedním z následujících způsobů:

   Windows: Z plovoucího ikonového menu *Šrafy*na nástrojovém panelu *Kresli* vyberte *Šrafy*.

   DOS a UNIX: Z menu *Kresli* vyberte *Šrafy*.

(a)

   2. Pod *Hranice šrafování* vyberte *Pokročilé* a (potom) v dialogovém panelu *Pokročilé možnosti* vyberte *Tvořit novou hraniční množinu*.

   3. Abyste definovali hraniční množinu, nejdříve při výzvě *Výběr objektů* určete rohové body hraniční množiny, a potom stiskněte *Enter.*

   4. V dialogovém panelu *Pokročilé možnosti* vyberte *OK*.

   5. V dialogovém panelu *Hraniční šrafování* vyberte *Výběr objektů,* (potom) určete vnitřní bod, (potom) stiskněte *Enter* a (nakonec) vyberte *Aplikovat* v dialogovém panelu *Hraniční šrafování,* abyste vyšrafovali plochu.

(b)

   2. Pod *Hranice šrafování* vyberte *Pokročilé*, takže se objeví dialogový panel *Pokročilé možnosti*, potom vyberte *Tvořit novou hraniční množinu*.

   3. Abyste definovali hraniční množinu, nejdříve při výzvě *Výběr objektů* určete rohové body hraniční množiny, a potom stiskněte *Enter.*

   4. V dialogovém panelu *Pokročilé možnosti* vyberte *OK,* takže dialogový panel *Pokročilé možnosti* zmizí.

   5. V dialogovém panelu *Hraniční šrafování* vyberte *Výběr objektů,* (potom) určete vnitřní bod, (potom) stiskněte *Enter* a (nakonec) vyberte *Aplikovat* v dialogovém panelu *Hraniční šrafování,* abyste vyšrafovali plochu.

9.2.2.5.3 Running Version of Text 5

Definování hraniční množiny v komplexním výkrese

Nejdříve otevřete dialogový panel *Hraniční množina* jedním z následujících způsobů:

Windows: Z plovoucího ikonového menu *Šrafy*na nástrojovém panelu *Kresli* vyberte *Šrafy*.

DOS a UNIX: Z menu *Kresli* vyberte *Šrafy*.

(a)

Potom pod *Hranice šrafování* vyberte *Pokročilé*. V dialogovém panelu *Pokročilé možnosti* vyberte *Tvořit novou hraniční množinu*. Abyste definovali hraniční množinu, nejdříve při výzvě *Výběr objektů* určete rohové body hraniční množiny, a potom stiskněte *Enter*. V dialogovém panelu *Pokročilé možnosti* vyberte *OK*. V dialogovém panelu *Hraniční šrafování* vyberte *Výběr objektů,* potom určete vnitřní bod, potom stiskněte *Enter* a nakonec vyberte *Aplikovat* v dialogovém panelu *Hraniční šrafování,* abyste vyšrafovali plochu.

(b)

2. Pod *Hranice šrafování* vyberte *Pokročilé*, takže se objeví dialogový panel *Pokročilé možnosti.* Vyberte *Tvořit novou hraniční množinu*. Abyste definovali hraniční množinu, nejdříve při výzvě *Výběr objektů* určete rohové body hraniční množiny, a potom stiskněte *Enter*. V dialogovém panelu *Pokročilé možnosti* vyberte *OK,* takže dialogový panel *Pokročilé možnosti* zmizí. V dialogovém panelu *Hraniční šrafování* vyberte *Výběr objektů,* potom určete vnitřní bod, potom stiskněte *Enter* a nakonec vyberte *Aplikovat* v dialogovém panelu *Hraniční šrafování,* abyste vyšrafovali plochu.

9.2.3 RUSSIAN

9.2.3.1 Based on IMD Text 1: pages 47-48

9.2.3.1.1 Baseline Text 1

Создание стиля мультилинии

Сначала откройте диалоговое окно одним из следующих способов:

Windows: Выберите пункт *Multiline Style* на панели инструментов *Object Properties* или выберите пункт *Multiline Style* в меню *Data.*

DOS and UNIX: Выберите пункт *Multiline Style* в меню *Data*.

Теперь добавьте элементы в стиль.

(a)

1. Нажмите кнопку *Element Properties.*

2. В диалоговом окне *Element Properties* введите смещение элемента мультилинии.

3. Определите элемент.

Сначала нажмите кнопку Add в диалоговом окне *Element Properties,* чтобы добавить элемент.

Затем в диалоговом окне *Element Properties* нажмите кнопку *Color*, чтобы задать цвет этого элемента. Затем выберите цвет элемента в диалоговом окне в *Select Color.*

И наконец в диалоговом окне *Element Properties* нажмите кнопку *Linetype*, чтобы задать тип линии этого элемента. Затем выберите тип линии элемента в диалоговом окне в *Select Linetype.*

Повторите эти шаги, чтобы задать еще один элемент.

4. Нажмите кнопку OK, чтобы сохранить стиль элементов мультилинии.

(b)

1. Нажмите кнопку *Element Properties*. На экране появится диалоговое окно *Element Properties.*

2. Введите смещение элемента мультилинии.

. 3. Определите элемент.

Сначала нажмите кнопку Add, чтобы добавить элемент.

Затем нажмите кнопку *Color*, чтобы задать цвет этого элемента. На экране появится диалоговое окно *Select Color*. Выберите цвет элемента. Диалоговое окно *Select Color* исчезнет с экрана.

И наконец нажмите кнопку *Linetype,* чтобы задать тип линии этого элемента. На экране появится диалоговое окно *Select Linetype*. Выберите тип линии элемента. Диалоговое окно *Select Linetype* исчезнет с экрана.

Повторите эти шаги, чтобы задать еще один элемент.

4. Нажмите кнопку OK, чтобы сохранить стиль элементов мультилинии. Диалоговое окно *Element Properties* исчезнет с экрана.

9.2.3.1.2 More aggregated Text 1

Создание стиля мультилинии

Сначала откройте диалоговое окно одним из следующих способов:

Windows: Выберите пункт *Multiline Style* на панели инструментов *Object Properties* или в меню *Data.*

DOS and UNIX: Выберите пункт *Multiline Style* в меню *Data*.

Теперь добавьте элементы в стиль.

(a)

1. Нажмите кнопку *Element Properties,* а затем в диалоговом окне *Element Properties* введите смещение элемента мультилинии.

2. Определите элемент.

Сначала нажмите кнопку Add в диалоговом окне *Element Properties,* чтобы добавить элемент.

Затем в диалоговом окне *Element Properties* нажмите кнопку *Color*, чтобы задать цвет этого элемента, затем выберите цвет элемента в диалоговом окне в *Select Color.*

И наконец в диалоговом окне *Element Properties* нажмите кнопку *Linetype*, чтобы задать тип линии этого элемента, затем выберите тип линии элемента в диалоговом окне в *Select Linetype*.

Повторите эти шаги, чтобы задать еще один элемент.

4. Нажмите кнопку ОК, чтобы сохранить стиль элементов мультилинии.

9.2.3.1.3 Running version of Text 1

Создание стиля мультилинии

Сначала откройте диалоговое окно одним из следующих способов:

Windows: Выберите пункт *Multiline Style* на панели инструментов *Object Properties* или в меню *Data.*

DOS and UNIX: Выберите пункт *Multiline Style* в меню *Data*.

Теперь добавьте элементы в стиль.

(a) Нажмите кнопку *Element Properties,* а затем в диалоговом окне *Element Properties* введите смещение элемента мультилинии.

Определите элемент.

Сначала нажмите кнопку Add в диалоговом окне *Element Properties,* чтобы добавить элемент. Затем в диалоговом окне *Element Properties* нажмите кнопку *Color*, чтобы задать цвет этого элемента, затем выберите цвет элемента в диалоговом окне в *Select Color.* И наконец в диалоговом окне *Element Properties* нажмите кнопку *Linetype*, чтобы задать тип линии этого элемента, затем выберите тип линии элемента в диалоговом окне в *Select Linetype*. Повторите эти шаги, чтобы задать еще один элемент.

4. Нажмите кнопку ОК, чтобы сохранить стиль элементов мультилинии.

9.2.3.2 Based on IMD Text 2: page 46

9.2.3.2.1 Baseline Text 2

Рисование полилинии, состоящей из отрезков прямых и дуг

Сначала нарисуйте отрезок прямой.

1.Запустите команду PLINE одним из следующих способов:

Windows: В палитре Polyline на панели инструментов Draw выберите пункт Polyline.

DOS and UNIX: В меню Draw выберите пункт Polyline.

2.Укажите начальную точку отрезка прямой.

3.Укажите конечную точку отрезка прямой.

Затем нарисуйте дугу.

1.(a) Перейдите в режим *Arc*. Сначала нажмите клавишу a. Затем нажмите *OK* в диалоговом окне *Arc mode.*

(b) Перейдите в режим *Arc*. Нажмите клавишу a.. На экране появится диалоговое окно *Arc mode.* Нажмите кнопку *OK*.

2.Укажите конечную точку дуги.

Затем нарисуйте отрезок прямой.

1.(a) Вернитесь в режим *Line*. Сначала нажмите клавишу *l*. Затем нажмите *OK* в диалоговом окне *Line mode confirmation.*

(b) Вернитесь в режим *Line*. Нажмите клавишу *l*. На экране появится диалоговое окно *Line mode confirmation.* Нажмите кнопку *OK*.

2.Укажите расстояние линии по отношению к конечной точке дуги.

3.Укажите угол линии по отношению к конечной точке дуги.

И наконец нажмите клавишу Return, чтобы завершить рисование полилинии.

9.2.3.2.2 More Aggregated Version of Text 2

Рисование полилинии, состоящей из отрезков прямых и дуг

Сначала нарисуйте отрезок прямой.

1. Запустите команду *PLINE* одним из следующих способов:

   Windows: В палитре *Polyline* на панели инструментов *Draw* выберите пункт *Polyline.*

   DOS and UNIX: В меню *Draw* выберите пункт *Polyline*.

2. Укажите начальную и конечную точку отрезка прямой.

Затем нарисуйте дугу.

1. Перейдите в режим *Arc* нажатием клавиши *a* и нажатием пункта *OK* в диалоговом окне *Arc mode.*

2. Укажите конечную точку дуги.

Затем нарисуйте отрезок прямой.

(a) 1. Вернитесь в режим *Line* нажатием клавиши *l* и нажатием *OK* в диалоговом окне *Line mode confirmation.*

2. Укажите расстояние и угол линии по отношению к конечной точке дуги.

И наконец нажмите клавишу Return, чтобы завершить рисование полилинии.

9.2.3.2.3 Running version of Text 2

Рисование полилинии, состоящей из отрезков прямых и дуг

Сначала нарисуйте отрезок прямой.

1. Запустите команду *PLINE* одним из следующих способов:

   Windows: В палитре *Polyline* на панели инструментов *Draw* выберите пункт *Polyline*.

   DOS and UNIX: В меню *Draw* выберите пункт *Polyline*.

Затем укажите начальную и конечную точку отрезка прямой.

2. Нарисуйте дугу. Сначала перейдите в режим *Arc* нажатием клавиши *a* и нажатием пункта *OK* в диалоговом окне *Arc mode*. Затем укажите конечную точку дуги.

3. Нарисуйте отрезок прямой. Сначала вернитесь в режим *Line* нажатием клавиши *l* и нажатием *OK* в диалоговом окне *Line mode confirmation.* Затем укажите расстояние и угол линии по отношению к конечной точке дуги.

4. Нажмите клавишу Return, чтобы завершить рисование полилинии.

9.2.3.3 Based on IMD Text 3: page 58

9.2.3.3.1 Baseline Version of Text 3

Рисование дуги

Сначала запустите команду *Arc* одним из следующих способов:

Windows: В палитре *Arc* на панели инструментов *Draw* выберите пункт *3 points.*

DOS and UNIX: В меню *Draw* выберите пункт *Arc.* Затем выберите пункт *3 points.*

Теперь задайте три точки дуги.

1. Укажите начальную точку (дуги). Сначала введите endp. Затем задайте линию. Дуга привязывается к конечной точке линии.

2. Укажите вторую точку (дуги). Сначала введите poi. Затем задайте точку. Дуга привязывается к этой точке.

3. Укажите конечную точку (дуги).

9.2.3.3.2 More Aggregated Version of Text 3

Рисование дуги по трем точкам

Сначала запустите команду *Arc* одним из следующих способов:

Windows: В палитре *Arc* на панели инструментов *Draw* выберите пункт *3 points.*

DOS and UNIX: В меню *Draw* выберите пункт *Arc.* Затем выберите пункт *3 points.*

Теперь задайте три точки дуги.

1. Укажите начальную точку дуги введением endp и заданием линии, причем дуга привязывается к конечной точке линии.

2. Укажите вторую точку дуги введя poi и выбав точку, причем дуга привязывается к конечной точке линии.

3. Укажите конечную точку (дуги).

9.2.3.3.3 Running Version of Text 3

Рисование дуги по трем точкам

Сначала запустите команду *Arc* одним из следующих способов:

Windows: В палитре *Arc* на панели инструментов *Draw* выберите пункт *3 points.*

DOS and UNIX: В меню *Draw* выберите пункт *Arc.* Затем выберите пункт *3 points.*

2. (Задайте три точки дуги.) Укажите начальную точку дуги введением endp и заданием линии, причем дуга привязывается к конечной точке линии. Затем укажите вторую точку дуги введя poi и выбав точку, причем дуга привязывается к конечной точке линии. Наконец укажите конечную точку (дуги).

9.2.3.4 Based on IMD Text 4: pages 48/9

9.2.3.4.1 Baseline Version of Text 4

Определение стиля мультилинии

(a)

Сначала выберите пункт Multiline Style в меню Data.

Затем задайте свойства мультилинии.

1. В диалоговом окне Multiline Styles выберите пункт Multiline Properties.
2. В диалоговом окне Multiline Properties выберите пункт Display joints, чтобы показать (отобразить) линию у вершин мультилинии.
3. (В диалоговом окне Multiline Properties) Задайте начальную точку мультилинии. В окне Caps задайте прямую или в окне Caps задайте дугу.
4. (В диалоговом окне Multiline Properties) Задайте конечную точку мультилинии. В окне Caps задайте прямую или в окне Caps задайте дугу.
5. (В диалоговом окне Multiline Properties) Задайте угол.
6. (В диалоговом окне Multiline Properties) Задайте заполняющий цвет фона. Сначала в окне Fill нажмите кнопку On, чтобы показать цвет фона. Затем нажмите кнопку Color. И наконец укажите заполняющий цвет фона в диалоговом окне Select Color.
7. (В диалоговом окне Multiline Properties) Нажмите кнопку OK, чтобы вернуться в диалоговое окно Multiline Styles.

(b)

Сначала выберите пункт Multiline Style в меню Data. (На экране) появится диалоговое окно Multiline Styles.

Затем задайте свойства мультилинии.

1. Выберите пункт Multiline Properties. (На экране) появится диалоговое окно Multiline Properties.
2. Выберите пункт Display joints, чтобы показать (отобразить) линию у вершин мультилинии.
3. Задайте начальную точку мультилинии. В окне Caps задайте прямую или в окне Caps задайте дугу.
4. Задайте конечную точку мультилинии. В окне Caps задайте прямую или в окне Caps задайте дугу.
5. Задайте угол.
6. Задайте заполняющий цвет фона. Сначала в окне Fill нажмите кнопку On, чтобы показать цвет фона. Затем нажмите кнопку Color. (На экране) появится диалоговое окно Selec Color. И наконец укажите заполняющий цвет фона.
7. Нажмите кнопку OK, чтобы вернуться в диалоговое окно Multiline Styles. Диалоговое окно Multiline Styles исчезнет (с экрана).

(a,b)

Теперь сохраните стиль.

1. В окне Name задайте имя стиля.
2. В окне Description задайте описание стиля.
3. Нажмите кнопку Add, чтобы добавить стиль мультилинии в рисунок (к рисунку).
4. Нажмите кнопку Save, чтобы сохранить стиль в файл.

(a) И наконец нажмите кнопку OK.

(b) И наконец нажмите кнопку OK. Диалоговое окно Multiline Styles исчезнет (с экрана).

9.2.3.4.2 More Aggregated Version of Text 4

Определение свойств мультилинии

(a)

Сначала выберите пункт Multiline Style в меню Data.

Затем задайте свойства мультилинии.

1. В диалоговом окне Multiline Styles выберите пункт Multiline Properties.
2. В диалоговом окне Multiline Properties выберите пункт Display joints, чтобы показать (отобразить) линию у вершин мультилинии.
3. (В диалоговом окне Multiline Properties) Задайте начальную точку мультилинии заданием прямой или дуги в окне Caps.
4. (В диалоговом окне Multiline Properties) Задайте конечную точку мультилинии заданием прямой или дуги в окне Caps.
5. (В диалоговом окне Multiline Properties) Задайте угол.
6. (В диалоговом окне Multiline Properties) Чтобы задать заполняющий цвет фона сначала нажмите кнопку On в окне Fill, чтобы показать цвет фона, затем нажмите кнопку Color, и наконец в диалоговом окне Selec Color укажите заполняющий цвет фона.
7. (В диалоговом окне Multiline Properties) Нажмите кнопку OK, чтобы вернуться в диалоговое окно Multiline Styles.

(b)

Сначала выберите пункт Multiline Style в меню Data. (На экране) появится диалоговое окно Multiline Styles..

Затем задайте свойства мультилинии.

1. Выберите пункт Multiline Properties. (На экране) появится диалоговое окно Multiline Properties.
2. Выберите пункт Display joints, чтобы показать (отобразить) линию у вершин мультилинии.
3. Задайте начальную точку и конечную точку мультилинии заданием прямой или дуги в окне Caps.
4. Задайте угол.
6. Чтобы задать заполняющий цвет фона сначала нажмите кнопку On в окне Fill, чтобы показать цвет фона, затем нажмите кнопку Color. (На экране) появится диалоговое окно Selec Color. Укажите заполняющий цвет фона.
7. И наконец нажмите кнопку OK, чтобы вернуться в диалоговое окно Multiline Styles, при этом диалоговое окно Multiline Properties исчезнет (с экрана).

(a,b)

Теперь сохраните стиль.

1. В окне Name задайте имя стиля.
2. В окне Description задайте описание стиля.
3. Нажмите кнопку Add, чтобы добавить стиль мультилинии в рисунок (к рисунку).
4. Нажмите кнопку Save, чтобы сохранить стиль в файл.

(a) И наконец нажмите кнопку OK.

(b) И наконец нажмите кнопку OK, при этом диалоговое окно Multiline Styles исчезнет (с экрана).

9.2.3.4.3 Semi-Running version of Text 4

Определение стиля мультилинии

(a)

1. Выберите пункт Multiline Style в меню Data.
2. Задайте свойства мультилинии.

   Сначала выберите пункт Multiline Properties диалоговом окне Multiline Styles.

   Затем выберите пункт Display joints, чтобы показать (отобразить) линию у вершин мультилинии.

   Затем (В диалоговом окне Multiline Properties) Задайте начальную точку и конечную точку мультилинии заданием прямой или дуги в окне Caps и (затем) задайте угол.

? Затем (В диалоговом окне Multiline Properties) Чтобы задать заполняющий цвет фона сначала нажмите кнопку On в окне Fill, чтобы показать цвет фона, затем нажмите кнопку Color, и наконец в диалоговом окне Selec Color укажите заполняющий цвет фона.

   И наконец (В диалоговом окне Multiline Properties) нажмите кнопку OK, чтобы вернуться в диалоговое окно Multiline Styles.

(b)

1. Выберите пункт Multiline Properties. (На экране) появится диалоговое окно Multiline Properties.
2. Задайте свойства мультилинии.

   Сначала выберите пункт Multiline Properties. (На экране) появится диалоговое окно Multiline Properties.

   Затем выберите пункт Display joints, чтобы показать (отобразить) линию у вершин мультилинии.

   Затем задайте начальную точку и конечную точку мультилинии заданием прямой или дуги в окне Caps и (затем) задайте угол.

   Затем, чтобы задать заполняющий цвет фона сначала нажмите кнопку On в окне Fill, чтобы показать цвет фона, затем нажмите кнопку Color. Появится диалоговое окно Selec Color. Укажите заполняющий цвет фона.

Затем нажмите кнопку OK, при этом диалоговое окно Multiline Styles исчезнет (с экрана).

(a,b)

3. Сохраните стиль. Сначала в окне Name задайте имя стиля, затем в окне Description задайте описание стиля, затем нажмите кнопку Add, чтобы добавить стиль мультилинии в рисунок (к рисунку) и наконец нажмите кнопку Save, чтобы сохранить стиль в файл.

(a) И наконец нажмите кнопку OK.

(b) И наконец нажмите кнопку OK, при этом диалоговое окно Multiline Styles исчезнет (с экрана).

9.2.3.5 Based on IMD Text 5: page 75

9.2.3.5.1 Baseline version of Text 5

Определение набора границ в сложном рисунке

1. Откройте диалоговое окно Boundary Hatch одним из следующих способов:

   Windows: В палитре Hatch на панели инструментов Draw выберите пункт Hatch.

   DOS a UNIX: В меню Draw выберите пункт Hatch.

2. (a) В пункте Boundary нажмите кнопку Advanced.

   (b) В пункте Boundary нажмите кнопку Advanced. На экране появится диалоговое окно Advanced Options.

3. (a) В диалоговом окне Advanced Options нажмите кнопку Make New Boundary Set.

   (b) Нажмите кнопку Make New Boundary Set.

4. Определите набор границ. Сначала в строке запроса Select Objects укажите угловые точки набора границ. Затем нажмите Return.

5. (a) В диалоговом окне Advanced Options нажмите кнопку OK.

   (b) В диалоговом окне Advanced Options нажмите кнопку OK. Диалоговое окно Advanced Options исчезнет (с экрана).

6. В диалоговом окне Boundary Hatch нажмите кнопку Pick Points.

7. Укажите внутреннюю точку.

8. Нажмите Enter.

9. В диалоговом окне Boundary Hatch нажмите кнопку Apply, чтобы применить штриховку.

9.2.3.5.2 More Aggregated Version of Text 5

Определение набора границ в сложном рисунке

1. Откройте диалоговое окно Boundary Hatch одним из следующих способов:

Windows: В палитре Hatch на панели инструментов Draw выберите пункт Hatch.

DOS a UNIX: В меню Draw выберите пункт Hatch.

(a)

2. В пункте Boundary нажмите кнопку Advanced и (затем) в диалоговом окне Advanced Options нажмите кнопку Make New Boundary Set.

3. Чтобы определить набор границ сначала в строке запроса Select Objects укажите угловые точки набора границ и затем нажмите Enter.

4. В диалоговом окне Advanced Options нажмите кнопку OK.

5. В диалоговом окне Boundary Hatch нажмите кнопку Pick Points, (затем) укажите внутреннюю точку, (затем) нажмите Enter и (наконец) нажмите кнопку Apply в диалоговом окне Boundary Hatch, чтобы применить штриховку.

(b)

2. В пункте Boundary нажмите кнопку Advanced при этом (на экране) появится диалоговое окно Advanced Options, затем нажмите кнопку Make New Boundary Set.

3. Чтобы определить набор границ сначала в строке запроса Select Objects укажите угловые точки набора границ и затем нажмите Enter.

4. В диалоговом окне Advanced Options нажмите кнопку OK, при этом диалоговое окно Advanced Options исчезнет (с экрана)..

5. В диалоговом окне Boundary Hatch нажмите кнопку Pick Points, (затем) укажите внутреннюю точку, (затем) нажмите Enter и (наконец) нажмите кнопку Apply в диалоговом окне Boundary Hatch, чтобы применить штриховку.

9.2.3.5.3 Running Version of Text 5

Определение набора границ в сложном рисунке

Сначала откройте диалоговое окно Boundary Hatch одним из следующих способов:

Windows: В палитре Hatch на панели инструментов Draw выберите пункт Hatch.

DOS a UNIX: В меню Draw выберите пункт Hatch.

(a)

Затем нажмите кнопку Advanced в пункте Boundary. В диалоговом окне Advanced Options нажмите кнопку Make New Boundary Set. Затем определите набор границ (a) указанием угловых точек набора границ и нажатием кнопки Enter В диалоговом окне Advanced Options нажмите кнопку OK. В диалоговом окне Boundary Hatch нажмите кнопку Pick Points, (затем) укажите внутреннюю точку, (затем) нажмите Enter и (наконец) нажмите кнопку Apply в диалоговом окне Boundary Hatch, чтобы применить штриховку

(b)

2. В пункте Boundary нажмите кнопку Advanced, при этом (на экране) появится диалоговое окно Advanced Options. Нажмите кнопку Make New Boundary Set. Чтобы определите набор границ, сначала в строке запроса Select Objects укажите угловые точки набора границ и затем нажмите Enter. В диалоговом окне Advanced Options нажмите кнопку OK, при этом диалоговое окно Advanced Options исчезнет (с экрана). В диалоговом окне Boundary Hatch нажмите кнопку Pick Points, затем укажите внутреннюю точку, затем нажмите Enter и наконец нажмите кнопку Apply в диалоговом окне Boundary Hatch, чтобы применить штриховку

9.2.4 BULGARIAN

9.2.4.1 Based on IMD Text 1: pages 47-48

9.2.4.1.1 Baseline Text 1

Създаване на стил на мултилиния

Първо отворете диалоговия прозорец *Multiline Style* като използвате един от следните методи:

Windows: От функционалния ред *Object Properties* или менюто *Data*, изберете *Multiline Style*.

DOS and UNIX: От менюто *Data*, изберете *Multiline Style*.

Сега добавете елементи към стила.

(a)

1. Изберете *Element Properties*.
2. В диалоговия прозорец *Element Properties* въведете отместването на елемента на мултилинията.
3. Дефинирайте елемент.

   Първо изберете *Add* в диалоговия прозорец *Element Properties* за да добавите елемент.

   Сега в диалоговия прозорец *Element Properties* изберете *Color,* за да посочите цвета на елемента. След това изберете цвят(елемент) от диалоговия прозорец *Select Color.*

   Накрая в диалоговия прозорец *Element Properties* изберете *Linetype*, за да посочите вида на линията на елемента. След това изберете вида на линията на елемента от диалоговия прозорец *Select Linetype*.

   Повторете тези стъпки, за да дефинирате друг елемент.
4. Изберете OK, за да запишете характеристиките на елементите на мултилинията.

(b)

1. Изберете *Element Properties*. Отваря се диалоговия прозорец *Element Properties.*
2. Въведете отместването на елементите на мултилинията.
3. Дефинирайте елемент.

   Първо изберете *Add*, за да добавите елемент.

   Сега изберете *Color,* за да посочите цвета на елемента. Отваря се диалоговия прозорец *Select Color*. Изберете цвят. Диалоговия прозорец *Select Color* се затваря.

   Накрая изберете *Linetype*, за да посочите вида на линията на елемента. Отваря се диалоговия прозорец *Select Linetype*. Изберете вида на линията на елемента. Диалоговия прозорец *Select Linetype* се затваря.

   Повторете тези стъпки, за да дефинирате друг елемент.
4. Изберете OK, за да запишете характеристиките на елементите на мултилинията. Диалоговия прозорец *Element Properties* се затваря.

This is a version with the following schematic structure:

Дефинирайте елемент.

1.(Първо) …

2. (Сега)…

3. (Накрая)…

4. () Повторете тези стъпки, за да дефинирате друг елемент.

Another possibility is to employ the following schematic structure:

Дефинирайте елемент като използвате следните стъпки.

1.(Първо) …

2. (Сега)…

3. (Накрая)…

If there would be only one step to repeat, singular number should be used, and running text rather than list.

9.2.4.1.2 More aggregated Text 1

Създаване на стил на мултилиния

Първо отворете диалоговия прозорец *Multiline Style* като използвате един от следните методи:

Windows: От функционалния ред *Object Properties* или менюто *Data*, изберете *Multiline Style*.

DOS and UNIX: Изберете *Multiline Style* от менюто *Data*.

Сега добавете елементи към стила.

(a)

1. Изберете *Element Properties* и след това въведете отместването на елемента на мултилинията в диалоговия прозорец *Element Properties*.

2. Дефинирайте елемент.

Първо изберете *Add* в диалоговия прозорец *Element Properties,* за да добавите елемент.

Сега в диалоговия прозорец *Element Properties* изберете *Color,* за да посочите цвета на елемента, а след това изберете цвят от диалоговия прозорец *Select Color*.

Накрая в диалоговия прозорец *Element Properties* изберете *Linetype*, за да посочите линията на елемента, а след това изберете линия на елемента от диалоговия прозорец *Select Linetype.*

Повторете тези стъпки, за да дефинирате друг елемент.

3. Изберете OK, за да запишете характеристиките на елемента на мултилинията.

9.2.4.1.3 Running version of Text 1

Създаване на стил на мултилиния

Първо отворете диалоговия прозорец *Multiline Style* като използвате един от следните методи:

Windows: От функционалния ред *Object Properties* или менюто *Data*, изберете *Multiline Style*.

DOS and UNIX: От менюто *Data*, изберете *Multiline Style*.

Сега прибавете елементи към стила.

(a)

1. Изберете *Element Properties,* а след това в диалоговия прозорец *Element Properties* въведете отместването на елемента на мултилинията.

2. Дефинирайте елемент.

Първо изберете *Add* в диалоговия прозорец *Element Properties,* за да добавите елемент. Сега в диалоговия прозорец *Element Properties* изберете *Color*, за да посочите цвета на елемента, а след това изберете цвят от диалоговия прозорец *Select Color*. Накрая в диалоговия прозорец *Element Properties* изберете *Linetype,* за да посочите вида на линията на елемента, а след това изберете вида на линията от диалоговия прозорец *Select Linetype*. Повторете тези стъпки, за да дефинирате друг елемент.

3. Изберете OK, за да запишете характеристиките на елемента на мултилинията.

9.2.4.2 Based on IMD Text 2: page 46

9.2.4.2.1 Baseline Text 2

Чертаене на полилиния, съставена от отсечки и дъги.

Първо начертайте отсечката.

1.Стартирайте командата *PLINE* като използвате един от следните методи:

Windows: От плаващото меню *Polyline* на функционалния ред *Draw*, изберете *Polyline*.

DOS and UNIX: От менюто *Draw* изберете *Polyline*.

2.Задайте началната точка на отсечката.

3.Задайте крайната точка на отсечката.

След това начертайте дъга.

1.(a) Превключете в режим *Arc*. Първо въведете *a*. След това изберете *OK* в диалоговия прозорец на режима Arc.

(b) Превключете в режим *Arc*. Въведете *a*. Появява се диалоговият прозорец на режима Arc. Изберете *OK*.

2.Задайте крайната точка на дъгата.

След това начертайте друга отсечка.

1.(a) Върнете се в режим *Line*. Първо въведете *l*. След това изберете *OK* в диалоговия прозорец на режима *Line*.

(b) Върнете се в режим *Line*. Въведете *l*. Появява се диалоговият прозорец на режима *Line*. Изберете *OK*.

2.Въведете дължината на отсечката от крайната точка на дъгата.

3.Въведете ъгъла на отсечката спрямо крайната точка на дъгата.

Накрая натиснете *Return*, за да завършите полилинията.

9.2.4.2.2 More Aggregated Version of Text 2

Чертаене на полилиния, съставена от отсечки и дъги

Първо начертайте отсечката.

1.Стартирайте командата *PLINE* като използвате един от следните методи:

Windows: От плаващото меню *Polyline* на функционалния ред *Draw*, изберете *Polyline*.

DOS and UNIX: От менюто *Draw* изберете *Polyline*.

2.Задайте началната и крайната точка на отсечката.

След това начертайте дъга.

(a) 1. Превключете в режим *Arc* като (първо) въведете *a* и (след това) изберете *OK* в диалоговия прозорец на режима *Arc*.

2. Задайте крайната точка на дъгата.

След това начертайте друга отсечка.

(a) 1.Върнете се в режим *Line* като (първо) въведете *l* и (след това) изберете *OK* в диалоговия прозорец на режима *Line*.

2. Въведете дължината и ъгъла на отсечката спрямо крайната точка на дъгата.

Накрая натиснете *Return,* за да завършите полилинията.

9.2.4.2.3 Running version of Text 2

Чертаене на полилиния, съставена от отсечки и дъги

1. Начертайте отсечката.

   Първо стартирайте командата PLINE като използвате един от следните методи:

   Windows: От плаващото меню *Polyline* на функционалния ред *Draw*, изберете *Polyline*.

   DOS and UNIX: От менюто *Draw* изберете *Polyline*.

   След това задайте началната и крайната точка на отсечката.

2. Начертайте дъга. Първо превключете в режим *Arc* като въведете *a* и изберете *OK* в диалоговия прозорец на режима *Arc*. След това задайте крайната точка на дъгата.
3. Начертайте друга дъга. Първо се върнете в режим *Line* като въведете *l* и изберете *OK* в диалоговия прозорец на режима *Line*. След това въведете дължината и ъгъла на отсечката спрямо крайната точка на дъгата.
4. Натиснете *Return*, за да завършите полилинията.

9.2.4.3 Based on IMD Text 3: page 58

9.2.4.3.1 Baseline Version of Text 3

Чертаене на дъга

Първо стартирайте командата *ARC* като използвате един от тези методи:

Windows: От плаващото меню *Arc* на функционалния ред *Draw* toolbar, изберете *3 Points*.

DOS and UNIX: От менюто *Draw* изберете *Arc*. След това изберете *3 Points*.

Сега задайте три точки на дъгата.

1. Задайте началната точка (на дъгата). Първо въведете *endp.* След това изберете отсечка. Дъгата се захваща за {крайната точка на линията/ за своята крайна точка}.
2. Задайте втората точка (на дъгата). Първо въведете *poi.* След това изберете точка. Дъгата се захваща за {точката/ нея}.
3. Задайте крайната точка (на дъгата).

9.2.4.3.2 More Aggregated Version of Text 3

Чертаене на дъга по три точки.

Първо стартирайте командата *ARC* като използвате един от тези методи:

Windows: От плаващото меню *Arc* на функционалния ред *Draw*, изберете *3 Points.*

DOS and UNIX: От менюто *Draw* изберете *Arc*. След това изберете *3 Points*.

Сега задайте три точки на дъгата.

1. Задайте началната точка (на дъгата), като въведете *endp* и посочите линия, така че дъгата да се захване за крайната точка на линията.
2. Задайте втората точка (на дъгата), като въведете *poi* и посочите точка,така че дъгата да се захване (за точката/ за нея).
3. Задайте крайната точка (на дъгата).

9.2.4.3.3 Running Version of Text 3

Чертаене на дъга по три точки.

1. Стартирайте командата ARC като използвате един от тези методи:

   Windows: От плаващото меню *Arc* на функционалния ред *Draw*, изберете *3 Points.*

   DOS and UNIX: От менюто *Draw* изберете *Arc*. След това изберете *3 Points*.

2. (Задайте три точки на дъгата.) Първо задайте началната точка(на дъгата), като въведете *endp* и посочите линията, така че дъгата да се захване за крайната точка на линията. След това задайте втората точка(на дъгата), като въведете *poi* и посочите точка {така че дъгата да се захване за точката/ така че дъгата да се захване за нея}. Накрая задайте крайната точка(на дъгата).

9.2.4.4 Based on IMD Text 4: pages 48/9

9.2.4.4.1 Baseline Version of Text 4

Задаване стил на мултилиния

(a)

Първо изберете *Multiline Style* от менюто *Data*.

След това задайте характеристиките на мултилинията.

1. В диалоговия прозорец *Multiline Styles* изберете *Multiline Properties*.
2. В диалоговия прозорец *Multiline Properties* изберете *Display Joints*, за да се появи линия във върховете на мултилинията.
3. (В диалоговия прозорец *Multiline Properties*) Задайте началото на мултилинията. Изберете линия от подменюто *Caps* или изберете дъга от подменюто *Caps*.
4. (В диалоговия прозорец *Multiline Properties*) Задайте края на мултилинията. Изберете линия от подменюто *Caps* или изберете дъга от подменюто *Caps*.
5. (В диалоговия прозорец *Multiline Properties*) Въведете ъгъл.
6. (В диалоговия прозорец *Multiline Properties*) Задайте основен запълващ цвят. Първо изберете *On* от подменюто *Fill*, за да се появи основният цвят. След това изберете *Color*. Накрая посочете основен запълващ цвят от диалоговия прозорец *Select Color*.
7. (В диалоговия прозорец *Multiline Properties*) Изберете *OK*, за да се върнете в диалоговия прозорец *Multiline Styles*.

(b)

Първо изберете *Multiline Style* от менюто *Data*. Появява се диалоговият прозорец *Multiline Styles*.

След това задайте характеристиките на мултилинията.

1. Изберете *Multiline Properties*. Появява се диалоговият прозорец *Multiline Properties*.
2. Изберете *Display Joints*, за да се появи линия във върховете на мултилинията.
3. Задайте началото на мултилинията. Изберете линия от подменюто *Caps* или изберете дъга от подменюто *Caps*.
4. Задайте края на мултилинията. Изберете линия от подменюто *Caps* или изберете дъга от подменюто *Caps*.
5. Въведете ъгъл.
6. Задайте основен запълващ цвят. Първо изберете *On* от подменюто *Fill*, за да се появи основният цвят. След това изберете *Color*. Появява се диалоговият прозорец *Select Color*.

Посочете основен запълващ цвят.

7. Изберете *OK*, за да се върнете в диалоговия прозорец *Multiline Styles*. Диалоговият прозорец *Multiline Properties* се затваря.

(a,b)

След това запишете стила.

1. В полето *Name* въведете {името на стила/ име}.
2. В полето *Description* въведете {описанието (на стила) / описание }.
3. Изберете *Add*, за да добавите стила към чертежа.
4. Изберете *Save*, за да запишете стила във файл.

(a) Накрая изберете *OK*.

(b) Накрая изберете *OK*. Диалоговият прозорец *Multiline Style* се затваря.

9.2.4.4.2 More Aggregated Version of Text 4

Задаване стил на мултилиния

(a)

Първо изберете *Multiline Style* от менюто *Data*.

След това задайте характеристиките на мултилинията.

1. В диалоговия прозорец *Multiline Styles* изберете *Multiline Properties*.
2. В диалоговия прозорец *Multiline Properties* изберете *Display Joints*, за да се появи линия във върховете на мултилинията.
3. (В диалоговия прозорец *Multiline Properties*) Задайте началото и края на мултилинията, като изберете линия или дъга от подменюто *Caps*.
4. (В диалоговия прозорец *Multiline Properties*) Въведете ъгъл.
5. (В диалоговия прозорец *Multiline Properties*) Задайте основен запълващ цвят, като първо изберете *On* от подменюто *Fill*, за да се появи основният цвят, като след това изберете *Color* и накрая посочите основен запълващ цвят в диалоговия прозорец *Select Color*.
6. (В диалоговия прозорец *Multiline Properties*) Изберете *OK*, за да се върнете в диалоговия прозорец *Multiline Styles*.

(b)

Първо изберете *Multiline Style* от менюто *Data*. Появява се диалоговият прозорец *Multiline Styles*.

След това задайте характеристиките на мултилинията.

1. Изберете *Multiline Properties*. Появява се диалоговият прозорец *Multiline Properties*.
2. Изберете *Display Joints*, за да се появи линия във върховете на мултилинията.
3. В подменюто *Caps* изберете линия или дъга, за да зададете началото на мултилинията и за да зададете края на мултилинията.
4. Въведете ъгъл.
6. Задайте основен запълващ цвят, като първо изберете *On* в подменюто *Fill*, за да се появи основният цвят и след това изберете *Color*. Появява се диалоговият прозорец *Select Color*.

Посочете основен запълващ цвят.

6. Изберете *OK*, за да се върнете в диалоговия прозорец *Multiline Styles*, за да се затвори диалоговият прозорец *Multiline Properties*.

(a,b)

След това запишете стила.

1. В полето *Name* въведете {името на стила/ име}.
2. В полето *Description* въведете {описанието (на стила) / описание }.
3. Изберете *Add*, за да добавите стила към чертежа.
4. Изберете *Save*, за да запишете стила във файл.

(a) Накрая изберете *OK*.

(b) Накрая изберете *OK*, за да се затвори диалоговият прозорец *Multiline Style*.

9.2.4.4.3 Semi-Running version of Text 4

Задаване стил на мултилиния

(a)

1. Изберете *Multiline Style* от менюто *Data*.

2. Задайте характеристиките на мултилинията.

   Първо изберете *Multiline Properties* в диалоговия прозорец *Multiline Styles*.

   След това изберете *Display Joints* в диалоговия прозорец *Multiline Properties*, за да се появи линия във върховете на мултилинията.

   След това задайте началото и края на мултилинията, като изберете линия или дъга в подменюто *Caps* и (след това) въведете ъгъл.

   След това задайте основен запълващ цвят, като първо изберете *On* от подменюто *Fill*, за да се появи основният цвят, след това изберете *Color* и накрая посочите основен запълващ цвят в диалоговия прозорец *Select Color*.

   Накрая изберете *OK*, за да се върнете в диалоговия прозорец *Multiline Styles*.

(b)

1. Изберете *Multiline Style* от менюто *Data*. Появява се диалоговият прозорец *Multiline Styles*.

2. Задайте характеристиките на мултилинията. Първо изберете *Multiline Properties*. Появява се диалоговият прозорец *Multiline Properties*.

   След това изберете *Display Joints*, за да се появи линия във върховете на мултилинията.

   След това изберете линия или дъга в подменюто *Caps*, за да зададете началото на мултилинията и за да зададете края на мултилинията.

   След това въведете ъгъл.

   След това задайте основен запълващ цвят, като първо изберете *On* в подменюто *Fill*, за да се появи основният цвят и след това изберете Color, за да се появи диалоговият прозорец *Select Color*. Посочете основния запълващ цвят.

   Накрая изберете *OK*, за да се върнете в диалоговия прозорец *Multiline Styles*, така че да се затвори диалоговият прозорец *Multiline Properties*.

(a,b)

3. Запишете стила. Първо въведете {името на стила/ име} в полето *Name,* след това въведете {описанието (на стила) / описание }в полето *Description,* след това изберете *Add*, за да добавите стила към чертежа и накрая изберете *Save*, за да запишете стила във файл.

.

4. (a) Накрая изберете *OK*.

   (b) Накрая изберете *OK*, за да се затвори диалоговият прозорец *Multiline Style*.

9.2.4.5 Based on IMD Text 5: page 75

9.2.4.5.1 Baseline version of Text 5

Дефиниране на област за щриховане в сложен чертеж

1. Отворете диалоговия прозорец *Boundary Hatch*, като използвате един от следните методи:

   Windows: От плаващото меню *Hatch* на функционалния ред *Draw*, изберете *Hatch*.

   DOS a UNIX: От менюто *Draw*, изберете *Hatch*.

2. (a) От подменюто *Boundary*, изберете *Advanced.*

   (b) От подменюто *Boundary*, изберете *Advanced*. Появява се диалоговият прозорец *Advanced Options.*

3. (a) От диалоговия прозорец *Advanced Options*, изберете *Make New Boundary Set*.

   (b) Изберете *Make New Boundary Set*.

4. Дефинирайте границите на множеството. След подсказващото съобщение *Select Objects*, задайте ъгловите точки на областта за щриховане и натиснете *Return.*

5. (a) Изберете *OK* в диалоговия прозорец *Advanced Options.*

   (b) Изберете *OK* в диалоговия прозорец *Advanced Options.* Диалоговият прозорец *Advanced Options се затваря.*

6. От диалоговия прозорец *Boundary Hatch*, изберете *Pick Points*.

7. Посочете вътрешна точка.

8. Натиснете *Return*.

9. От диалоговия прозорец *Boundary Hatch* изберете *Apply*,.за да получите щриховката.

9.2.4.5.2 More Aggregated Version of Text 5

Дефиниране на област за щриховане в сложен чертеж

1. Отворете диалоговия прозорец *Boundary Hatch*, като използвате един от следните методи:

Windows: От плаващото меню *Hatch* на функционалния ред *Draw*, изберете *Hatch*.

DOS a UNIX: От менюто *Draw*, изберете *Hatch*.

(a)

2. От подменюто *Boundary* изберете *Advanced*, а (след това) изберете Make *New Boundary Set* от диалоговият прозорец *Advanced Options*.

3. За да дефинирате границите на множеството, (първо) задайте ъгловите точки на областта за щриховане, а (след това) натиснете *Return.*

4. Изберете *OK* в диалоговия прозорец *Advanced Options.*

5. От диалоговия прозорец *Boundary Hatch*, изберете *Pick Points*, (след това) посочете вътрешна точка, (след това) натиснете *Return*, и (накрая) изберете *Apply*,.за да получите щриховката.

(b)

2. От подменюто *Boundary* изберете *Advanced*, за да се появи диалоговият прозорец *Advanced Options,* след това изберете Make *New Boundary Set*.

3. За да дефинирате границите на множеството, (първо) задайте ъгловите точки на областта за щриховане, а (след това) натиснете *Return.*

4. Изберете *OK* в диалоговия прозорец *Advanced Options*, за да се затвори диалоговият прозорец *Advanced Option.*

5. От диалоговия прозорец *Boundary Hatch*, изберете *Pick Points*, (след това) посочете вътрешна точка, (след това) натиснете *Return*, и (накрая) изберете *Apply*,.за да получите щриховката. *Running Version of Text 5*

9.2.4.5.3 Running Version of Text 5

Дефиниране на област за щриховане в сложен чертеж

Първо отворете диалоговия прозорец *Boundary Hatch*, като използвате един от следните методи:

Windows: От плаващото меню *Hatch* на функционалния ред *Draw*, изберете *Hatch*.

DOS a UNIX: От менюто *Draw*, изберете *Hatch.*

(a)

След това от подменюто *Boundary* изберете *Advanced.* От диалоговият прозорец *Advanced Options* изберете Make *New Boundary Set.* След това дефинирайте границите на множеството като (първо) зададате ъгловите точки на областта за щриховане и (след това) натиснете *Return.* След това изберете *OK* в диалоговия прозорец *Advanced Options*. Накрая от диалоговия прозорец *Boundary Hatch*, изберете *Pick Points*, (след това) посочете вътрешна точка, (след това) натиснете *Return*, и (накрая) изберете *Apply*,.за да получите щриховката.

(b)

След това от подменюто *Boundary* изберете *Advanced* за да се появи диалоговият прозорец *Advanced Options.* Изберете Make *New Boundary Set.* След това дефинирайте границите на множеството като (първо) зададате ъгловите точки на областта за щриховане и (след това) натиснете *Return.* След това изберете *OK* в диалоговия прозорец *Advanced Options.* Накрая от диалоговия прозорец *Boundary Hatch*, изберете *Pick Points*, (след това) посочете вътрешна точка, (след това) натиснете *Return*, и (накрая) изберете *Apply*,.за да получите щриховката.

9.3 Basic steps

9.3.1 English

9.3.1.1 IMD Text 1: pages 47-48

To create a multiline style
Open the dialog box Multiline style.
Choose Element Properties.
[In the Element properties dialog box] Enter the offset of the multiline element.
Select Add.
Choose Color.
Select the element's color [from the Select Color dialog box].
Choose Linetype.
Select the element's linetype [from the Linetype dialog box].
Repeat these steps to define another element.
Choose OK.

9.3.1.2 IMD Text 2: page 46

To draw a line and arc combination polyline
Start the PLINE command.
1. Specify the start point of the line segment.
2. Specify the endpoint of the line segment.
3. Enter a.
4. Select OK in the Arc mode confirmation dialog box.
5. Specify the endpoint of the arc.
6. Enter l.
7. Select OK in the Line mode confirmation dialog box.
8. Enter the distance of the line in relation to the endpoint of the arc.
9. Enter the angle of the line in relation to the endpoint of the arc.
10. Press Return.

9.3.1.3 IMD Text 3: page 58

To draw an arc by specifying three points.
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.
Enter endp.
Select the line (so the arc snaps to the endpoint of the line).
Enter poi.
Select a point to snap to.
Specify the endpoint.

9.3.1.4 IMD Text 4: pages 48-49

To specify the properties of a multiline and save the style.

From the Data menu, choose Multiline Style.

Choose Multiline Properties

Select Display Joints.

Under Caps: Select a line or an arc for the startpoint of the multiline. Select a line or an arc for the endpoint of the multiline. Enter an angle.

Under Fill, select On.

Choose Color.

Select the background fill color.

Choose OK.

Under Name, enter the name of the style.

Under Description, enter a description.

Select Add.

Select Save.

9.3.1.5 IMD Text 5: page 75

To define a boundary set in a complex drawing

Windows: From the Hatch flyout on the Draw toolbar, choose Hatch.

DOS and UNIX: From the Draw menu, choose Hatch

Under Boundary choose Advanced.

Choose Make New Boundary Set.

At the Select Objects prompt, specify the corner points for the boundary set.

Press Return.

Choose OK.

Choose Pick Points.

Specify the internal point.

Press return.

In the Boundary Hatch dialog box, choose Apply.

9.3.2 Bulgarian

9.3.2.1 IMD Text 1: pages 47-48

9.3.2.1.1 Personal style

Създаване стил на мултилиния

Отворете диалоговия прозорец Multiline Style.

Изберете Element Properties.

[в диалоговия прозорец Element Properties] Въведете отместването на елемента на мултилинията .

Посочете Add.

Изберете Color.

Посочете цвета на елемента [от диалоговия прозорец Select Color].

Изберете Linetype.

Посочете вида на линията на елемента [от диалоговия прозорец Select Linetype] .

Повторете тези стъпки.

Изберете OK.

9.3.2.1.2 Impersonal style

Създаване стил на мултилиния

Отваря се диалоговият прозорец Multiline Style.

Избира се Element Properties.

Въвежда се отместването на елемента на мултилинията [в диалоговия прозорец Element Properties].

Посочва се Add.

Избира се Color.

Посочва се цветът на елемента [в диалоговия прозорец Select Color].

Избира се Linetype.

Посочва се видът на линията на елемента [в диалоговия прозорец Select Linetype].

Повтарят се тези стъпки.

Избира се OK.

9.3.2.2 IMD Text 2: page 46

9.3.2.2.1 Personal style

Чертаене на полилиния, съставена от отсечки и дъги
Стартирайте командата PLINE.
1. Задайте началната точка на отсечката.
2. Задайте крайната точка на отсечката.
3. Въведете **a**.
4. Изберете OK в диалоговия прозорец на режима Arc.
5. Задайте крайната точка на дъгата.
6. Въведете **l**.
7. Изберете OK в диалоговия прозорец на режима Line.
8. Въведете дължината на отсечката от крайната точка на дъгата.
9. Въведете ъгъла на отсечката спрямо крайната точка на дъгата.
10. Натиснете Return.

9.3.2.2.2 Impersonal style

Чертаене на полилиния, съставена от отсечки и дъги
Стартира се командата PLINE.
1. Задава се началната точка на отсечката.
2. Задава се крайната точка на отсечката.
3. Въвежда се **a**.
4. Избира се OK в диалоговия прозорец на режима Arc.
5. Задава се крайната точка на дъгата.
6. Въвежда се **l**.
7. Избира се OK в диалоговия прозорец на режима Line.
8. Въвежда се дължината на отсечката от крайната точка на дъгата.
9. Въвежда се ъгълът на отсечката спрямо крайната точка на дъгата.
10. Натиска се Return.

9.3.2.3 IMD Text 3: page 58

9.3.2.3.1 Personal style

Чертаене на дъга по три точки.
Windows: От плаващото меню Arc на функционалния ред Draw, изберете 3 Points.
DOS and UNIX: От менюто Draw, изберете Arc. След това изберете 3 Points.
Въведете *endp*.
Посочете линия (така че дъгата да се захване за крайната точка на линията).
Въведете *poi*.
Посочете точка на захващане.
Задайте крайната точка.

9.3.2.3.2 Impersonal style

Чертаене на дъга по три точки.
Windows: От плаващото меню Arc на функционалния ред Draw се избира 3 Points.
DOS and UNIX: От менюто Draw се избира Arc. След това се избира 3 Points.
Въвежда се *endp*.
Посочва се линия (така че дъгата да се захване за крайната точка на линията).
Въвежда се *poi*.
Посочва се точка на захващане.
Задава се крайната точка.

9.3.2.4 IMD Text 4: pages 48-49

9.3.2.4.1 Personal style

Задаване характеристиките на мултилиния и записване на стила

От менюто Data изберете Multiline Style.

Изберете Multiline Properties.

Изберете Display Joints.

От подменюто Caps: Изберете линия или дъга за началото на мултилинията. Изберете линия или дъга за края на мултилинията. Въведете ъгъл.

От подменюто Fill изберете On.

Изберете цвят.

Изберете основен запълващ цвят.

Изберете OK.

В подменюто Name въведете име на стила.

В подменюто Description въведете описание.

Изберете Add.

Изберете Save.

9.3.2.4.2 Impersonal style

Задаване характеристиките на мултилиния и записване на стила

От менюто Data се избира Multiline Style.

Избира се Multiline Properties.

Избира се Display Joints.

От подменюто Caps: Избира се линия или дъга за началото на мултилинията. Избира се линия или дъга за края на мултилинията. Въвежда се ъгъл.

От подменюто Fill се избира On.

Избира се цвят.

Избира се основен запълващ цвят.

Избира се OK.

В подменюто Name се въвежда име на стила.

В подменюто Description се въвежда описание.

Избира се Add.

Избира се Save.

9.3.2.5 IMD Text 5: page 75

9.3.2.5.1 Personal style

Дефиниране на област за щриховане в сложен чертеж

Windows: От плаващото меню Hatch flyout на функционалния ред Draw изберете Hatch.

DOS and UNIX: От менюто Draw изберете Hatch.

От подменюто Boundary изберете Advanced.

Изберете Make New Boundary Set.

След подсказващото съобщение Select Objects задайте ъгловите точки на областта за щриховане.

Натиснете Return.

Изберете OK.

Изберете Pick Points.

Посочете вътрешната точка.

Натиснете Return.

От диалоговия прозорец Boundary Hatch изберете Apply.

9.3.2.5.2 Impersonal style

Дефиниране на област за щриховане в сложен чертеж

Windows: От плаващото меню Hatch flyout на функционалния ред Draw се избира Hatch.

DOS and UNIX: От менюто Draw се избира Hatch.

От подменюто Boundary се избира Advanced.

Избира се Make New Boundary Set.

След подсказващото съобщение Select Objects се задават ъгловите точки на областта за щриховане.

Натиска се Return.

Избира се OK.

Избира се Pick Points.

Посочва се вътрешната точка.

Натиска се Return.

От диалоговия прозорец Boundary Hatch се избира Apply.

9.3.3 Czech

9.3.3.1 IMD Text 1: pages 47-48

9.3.3.1.1 Personal style

Vytvoření stylu multičáry
Windows: Z nástrojového panelu Vlastnosti objektů nebo z menu Data vyberte *Styl mutičáry*.
DOS a UNIX: Z menu Data vyberte *Styl multičáry.*
1. Vyberte *Vlastnosti prvků*.
2. V dialogovém panelu Vlastnosti prvků zadejte rozměr posunutí multičáry.
3. Vyberte *Přidat*.
4. Vyberte *Barva.*
5. Zvolte barvu elementu z dialogového panelu Výběr barvy.
6. Vyberte *Typ čáry*.
7. Zvolte typ čáry daného elementu z dialogového panelu Výběr typů čar.
8. Opakujte tyto kroky.
9. Vyberte OK.

9.3.3.1.2 Impersonal style

Vytvoření stylu multičáry
Windows: Z nástrojového panelu Vlastnosti objektů nebo z menu Data se vybere *Styl mutičáry*.
DOS a UNIX: Z menu Data se vybere *Styl multičáry.*
1. Vybere se *Vlastnosti prvků*.
2. V dialogovém panelu Vlastnosti prvků se zadá rozměr posunutí multičáry.
3. Vybere se *Přidat*.
4. Vybere se *Barva.*
5. Zvolí se barva elementu z dialogového panelu Výběr barvy.
6. Vybere se *Typ čáry*.
7. Zvolí se typ čáry daného elementu z dialogového panelu Výběr typů čar.
8. Tyto kroky se opakují.
9. Vybere se OK.

9.3.3.2 IMD Text 2: page 46

9.3.3.2.1 Personal style

Nakreslení křivky kombinované z přímek a oblouků
Windows: Z plovoucího ikonového menu Křivka na nástrojovém panelu Kresli vyberte *Křivka*.
DOS a UNIX: Z menu Kresli vyberte *Křivka.*
1. Určete počáteční bod rovného segmentu.
2. Určete koncový bod rovného segmentu.
3. Zadejte **o**.
4. Vyberte OK v dialogovém panelu Potvrzení režimu kreslení oblouků.
5. Určete koncový bod oblouku.
6. Zadejte e.
7. Vyberte OK v dialogovém panelu Potvrzení režimu kreslení úseček.
8. Zadejte vzdálenost úsečky ve vztahu ke koncovému bodu oblouku.
9. Zadejte úhel úsečky ve vztahu ke koncovému bodu oblouku.
10. Stiskněte ENTER.

9.3.3.2.2 Impersonal style

Nakreslení křivky kombinované z přímek a oblouků
Windows: Z plovoucího ikonového menu Křivka na nástrojovém panelu Kresli se vybere *Křivka*.
DOS a UNIX: Z menu Kresli se vybere *Křivka.*
1. Určí se počáteční bod rovného segmentu.
2. Určí se koncový bod rovného segmentu.
3. Zadá se **o**.
4. Vybere se OK v dialogovém panelu Potvrzení režimu kreslení oblouků.
5. Určí se koncový bod oblouku.
6. Zadá se **e**.
7. Vybere se OK v dialogovém panelu Potvrzení režimu kreslení úseček.
8. Zadá se vzdálenost úsečky ve vztahu ke koncovému bodu oblouku.
9. Zadá se úhel úsečky ve vztahu ke koncovému bodu oblouku.
10. Stiskne se ENTER.

9.3.3.3 IMD Text 3: page 58

9.3.3.3.1 Personal style

Nakreslení oblouku určením tří bodů
Windows: Z plovoucího ikonového menu Oblouk na nástrojovém panelu Kresli vyberte 3 body.
DOS a Unix: Z menu Kresli vyberte Oblouk. Vyberte 3 body.
1. Zadejte *kon* a vyberte čáru.
2. Zadejte *bod* a vyberte bod.
3. Určete koncový bod.

9.3.3.3.2 Impersonal style

Nakreslení oblouku určením tří bodů
Windows: Z plovoucího ikonového menu Oblouk na nástrojovém panelu Kresli se vybere 3 body.
DOS a Unix: Z menu Kresli se vybere Oblouk. Vybere se 3 body.
1. Zadá se *kon* a vybere se čára.
2. Zadá se *bod* a vybere se bod
3. Určí se koncový bod.

9.3.3.4 IMD Text 4: pages 48-49

9.3.3.4.1 Personal style

Určení vlastností multičáry a uložení stylu

1. Z menu Data vyberte *Styl multičáry.*
2. Vyberte *Vlastnosti multičáry.*
3. V dialogovém panelu *Vlastnosti multičáry* vyberte *Zobraz klouby.*
4. Pod *Zakončení* zvolte úsečku nebo oblouk pro počáteční bod multičáry
5. Vyberte úsečku nebo oblouk pro koncový bod multičáry.
6. Zadejte úhel.
7. Pod *Vyplnění* vyberte *Ano.*
8. Vyberte *Barva.*
9. Zvolte barvu pro vyplnění pozadí.
10. Vyberte *OK*
11. Pod *Jméno* zadejte název stylu.
12. Pod *Popis* zadejte popis.
13. Vyberte Přidat.
14. Vyberte *Uložit.*
15. Vyberte *OK.*

9.3.3.4.2 Impersonal style

Určení vlastností multičáry a uložení stylu

1. Z menu Data se vybere *Styl multičáry.*
2. Vybere se *Vlastnosti multičáry.*
3. V dialogovém panelu *Vlastnosti multičáry* se vybere *Zobraz klouby*
4. Pod *Zakončení* se zvolí úsečka nebo oblouk pro počáteční bod multičáry.
5. Vybere se úsečka nebo oblouk pro koncový bod multičáry.
6. Zadá se úhel.
7. Pod *Vyplnění* se vybere *Ano.*
8. Vybere se *Barva.*
9. Z dialogového panelu Výběr barvy se zvolí barva pro vyplnění pozadí.
10. Vybere se *OK*
11. Pod *Jméno* se zadá název stylu.
12. Pod *Popis* se zadá popis.
13. Vybere se Přidat.
14. Vybere se *Uložit.*
15. Vybere se *OK.*

9.3.3.5 IMD Text 5: page 75

9.3.3.5.1 Personal style

Definování hraniční množiny v komplexním výkrese
Windows: Z plovoucího ikonového menu Šrafy na nástrojovém panelu Kresli vyberte Šrafy.
DOS a Unix: Z menu Kresli vyberte Šrafy.
1. Pod Hranice šrafování vyberte Pokročilé.
2. V dialogovém panelu Pokročilé možnosti vyberte Tvořit novou hraniční množinu.
3. Při výzvě Výběr objektů určete rohové body hraniční množiny.
4. Stiskněte Enter.
5. V dialogovém panelu Pokročilé možnosti vyberte OK.
6. V dialogovém panelu Hraniční šrafování vyberte Výběr objektů.
7. Určete vnitřní bod.
8. Stiskněte Enter.
9. V dialogovém panelu Hraniční šrafování vyberte Aplikovat.

9.3.3.5.2 Impersonal style

Definování hraniční množiny v komplexním výkrese
Windows: Z plovoucího ikonového menu Šrafy na nástrojovém panelu Kresli se vybere Šrafy.
DOS a Unix: Z menu Kresli se vybere Šrafy.
1. Pod Hranice šrafování se vybere Pokročilé.
2. V dialogovém panelu Pokročilé možnosti se vybere Tvořit novou hraniční množinu.
3. Při výzvě Výběr objektů se určí rohové body hraniční množiny.
4. Stiskne se Enter.
5. V dialogovém panelu Pokročilé možnosti se vybere OK.
6. V dialogovém panelu Hraniční šrafování se vybere Výběr objektů.
7. Určí se vnitřní bod.
8. Stiskne se Enter.
9. V dialogovém panelu Hraniční šrafování se vybere Aplikovat.

9.3.4 Russian

9.3.4.1 IMD Text 1: pages 47-48

9.3.4.1.1 Personal style

Создание стиля мультилинии
Откройте диалоговое окно Multiline style.
Нажмите кнопку Element Properties.
[В диалоговом окне Element properties] введите смещение элемента мультилинии.
Нажмите кнопку Add.
Нажмите кнопку Color.
Выберите цвет элемента [в диалоговом окне Select Color].
Нажмите кнопку Linetype.
Выберите тип линии [в диалоговом окне Linetype].
Повторите эти шаги, чтобы определить еще один элемент.
Нажмите кнопку OK.

9.3.4.1.2 Impersonal style

Чтобы создать стиль мультилинии
Необходимо открыть диалоговое окно Multiline style.
Нажать кнопку Element Properties.
[В диалоговом окне Element properties] ввести смещение элемента мультилинии.
Нажать кнопку Add.
Нажать кнопку Color.
Выбрать цвет элемента [в диалоговом окне Select Color].
Нажать кнопку Linetype.
Выбрать тип линии [в диалоговом окне Linetype].
Повторить эти шаги, чтобы определить еще один элемент.
Нажать кнопку OK.

9.3.4.2 IMD Text 2: page 46

9.3.4.2.1 Personal style

Рисование полилинии, состоящей из отрезков прямых и дуг

Запустите команду PLINE.

1. Укажите начальную точку отрезка прямой.
2. Укажите конечную точку отрезка прямой.
3. Нажмите клавишу a.
4. Нажмите кнопку OK в диалоговом окне Arc mode.
5. Укажите конечную точку дуги.
6. Нажмите клавишу l.
7. Нажмите кнопку OK в диалоговом окне Line mode.
8. Укажите расстояние линии по отношению к конечной точке дуги.
9. Укажите угол линии по отношению к конечной точке дуги.
10. Нажмите клавишу Return.

9.3.4.2.2 Impersonal style

Чтобы нарисовать полилинию, состоящую из отрезков прямых и дуг

Необходимо запустить команду PLINE.

1. Указать начальную точку отрезка прямой.
2. Указать конечную точку отрезка прямой.
3. Нажать клавишу a.
4. Нажать кнопку OK в диалоговом окне Arc mode.
5. Укарзать конечную точку дуги.
6. Нажать клавишу l.
7. Нажать кнопку OK в диалоговом окне Line mode.
8. Указать расстояние линии по отношению к конечной точке дуги.
9. Указать угол линии по отношению к конечной точке дуги.
10. Нажать клавишу Return.

9.3.4.3 IMD Text 3: page 58

9.3.4.3.1 Personal style

Рисование дуги по трем (заданным) точкам
Windows: В палитре Arc на панели инструментов Draw выберите пункт 3 Points.
DOS and UNIX: В меню Draw выберите Arc. Затем выберите пункт 3 Points.
Введите endp.
Задайте линию, к конечной точке которой привязана дуга.
Введите poi.
Задайте точку привязки дуги.
Задайте точку привязки дуги

9.3.4.3.2 Impersonal style

Чтобы нарисовать дугу по трем (заданным) точкам
Необходимо
Windows: В палитре Arc на панели инструментов Draw выбрать пункт 3 Points.
DOS and UNIX: В меню Draw выбрать Arc. Затем выбрать пункт 3 Points.
Ввести endp.
Задать линию, к конечной точке которой привязана дуга.
Ввести poi.
Задать точку привязки дуги.
Задать точку привязки дуги.

9.3.4.4 IMD Text 4: pages 48-49

9.3.4.4.1 Personal style

Определение свойств и сохранение стиля мультилинии
В меню Data выберите пункт Multiline Style.
В диалоговом окне Multiline Styles выберите пункт Multiline Properties.
Выберите пункт Display Joints.
В окне Caps задайте прямую или дугу для начальной точки мультилинии.
Задайте прямую или дугу для конечной точки мультилинии.
Задайте угол.
В окне Fill, выберите пункт On.
Нажмите кнопку Color.
В диалоговом окне Select Color укажите заполняющий цвет фона.
Нажмите OK.
В окне Name введите имя стиля.
В окне Description введите описание.
Нажмите кнопку Add.
Выберите пункт Save.

9.3.4.4.2 Impersonal style

Чтобы определить свойства мультилинии и сохранить ее стиль
Необходимо
В меню Data выбрать пункт Multiline Style.
В диалоговом окне Multiline Styles выбрать пункт Multiline Properties.
Выбрать пункт Display Joints.
В окне Caps задать прямую или дугу для начальной точки мультилинии.
Задать прямую или дугу для конечной точки мультилинии.
Задать угол.
В окне Fill выбрать пункт On.
Нажать кнопку Color.
В диалоговом окне Select Color указать заполняющий цвет фона.
Нажать OK.
В окне Name ввести имя стиля.
В окне Description ввести описание.
Нажать кнопку Add.
Выбрать пункт Save.

9.3.4.5 IMD Text 5: page 75

9.3.4.5.1 Personal style

Определение набора границ в сложном рисунке
Откройте диалоговое окно Boundary Hatch.
В пунктеBoundary нажмите кнопку Advanced.
Нажмите кнопку Make New Boundary Set.
В строке запроса Select Objects укажите граничные точки набора границ.
Нажмите кнопку Return.
В диалоговом окне Advanced Options нажмите кнопку OK.
В диалоговом окне Boundary Hatch выберите кнопку Pick Points.
Укажите внутреннюю точку.
Нажмите кнопку Return.
В диалоговом окне Boundary Hatch нажмите кнопку Apply.

9.3.4.5.2 Impersonal style

Чтобы определить набор границ в сложном рисунке
Необходимо открыть диалоговое окно Boundary Hatch.
В пунктеBoundary нажать кнопку Advanced.
Нажать кнопку Make New Boundary Set.
В строке запроса Select Objects указать граничные точки набора границ.
Нажать кнопку Return.
В диалоговом окне Advanced Options нажать кнопку OK.
В диалоговом окне Boundary Hatch выбрать кнопку Pick Points.
Указать внутреннюю точку.
Нажать кнопку Return.
В диалоговом окне Boundary Hatch нажать кнопку Apply.

9.4 Function descriptions

9.4.1 English

9.4.1.1 IMD Text 1: pages 47-48

9.4.1.1.1 User actions-based functional descriptive text.

* Choosing Multiline Style from the Object Properties toolbar or the Data menu under Windows opens the Multiline Styles dialog box.
* Choosing Multiline Style from the Data menu under DOS and UNIX opens the Multiline Styles dialog box.
Selecting Add in the Element Properties dialog box adds an element.
* Choosing Color in the Element Properties dialog box opens the Select Color dialog box.
* Choosing Linetype in the Element Properties dialog box opens the Select Linetype dialog box.
Choosing OK in the Element Properties dialog box saves the style of the multiline element and exits the Element Properties dialog box.

9.4.1.1.2 Object function-based functional descriptive text

The Multiline Style button from the Object Properties toolbar or the Data menu under Windows opens the Multiline Styles dialog box.
The Multiline Style button from the Data menu under DOS and UNIX opens the Multiline Styles dialog box.
The Add button in the Element Properties dialog box adds an element.
The Color button in the Element Properties dialog box opens the Select Color dialog box.
The Linetype button in the Element Properties dialog box opens the Select Linetype dialog box.
The OK in the Element Properties dialog box saves the style of the multiline element and exits the Element Properties dialog box.

9.4.1.2 IMD Text 2: page 46

9.4.1.2.1 User actions-based functional descriptive text.

Choosing Polyline from the Polyline flyout on the Draw toolbar under Windows starts the PLINE command
Choosing Polyline from the Draw menu under DOS and UNIX starts the PLINE command
Entering a (when PLINE is / has been started) switches to Arc mode.
Entering l (when PLINE is / has been started) switches to Line mode.

9.4.1.2.2 Object function-based functional descriptive text

The Polyline button from the Polyline flyout on the Draw toolbar under Windows starts the PLINE command.
The Polyline button from the Draw menu under DOS and UNIX starts the PLINE command.
The a key (when PLINE is / has been started) switches to Arc mode.
The l key (when PLINE is / has been started) switches to Line mode.

9.4.1.3 IMD Text 3: page 58

9.4.1.3.1 User actions-based functional descriptive text.

Choosing 3 Points from the Arc flyout on the Draw toolbar under Windows starts the ARC command.

*Choosing Arc then choosing 3 Points from the Draw menu under DOS and UNIX start / starts (??) the ARC command.

*Entering endp and then selecting the line so the arc snaps to the endpoint of the line when the ARC command is / has been started specifies / specify the endpoint.

9.4.1.3.2 Object function-based functional descriptive text

The 3 Points button from the Arc flyout on the Draw toolbar under Windows starts the ARC command.

The Arc button and then the 3 Points button from the Draw menu under DOS and UNIX starts the ARC command.

(*Entering endp and then selecting the line so the arc snaps to the endpoint of the line when the ARC command is / has been started specifies / specify the endpoint.)

9.4.1.4 IMD Text 4: pages 48-49

9.4.1.4.1 User actions-based functional descriptive text.

*Choosing Multiline Style from the Data menu opens the Multiline Style dialog box.

*Choosing Multiline Properties in the Multiline Styles dialog box opens the Multiline Properties dialog box.

Selecting Display Joints in the Multiline Properties dialog box displays a line at the vertices of the multiline.

Selecting On under Fill in the Multiline Properties dialog box displays a background color.

Choosing OK in the Multiline Properties dialog box exits the Multiline Properties dialog box.

Selecting Add in the Multiline Style dialog box add a style to a drawing.

Selecting Save in the Multiline Style dialog box saves a style to a file.

Choose OK in the Multiline Style dialog box closes the dialog box.

9.4.1.4.2 Object function-based functional descriptive text

The Multiline Style button from the Data menu opens the Multiline Style dialog box.

The Multiline Properties button in the Multiline Styles dialog box opens the Multiline Properties dialog box.

The Display Joints button in the Multiline Properties dialog box displays a line at the vertices of the multiline.

The On button under Fill in the Multiline Properties dialog box displays a background color.

The OK button in the Multiline Properties dialog box exits the Multiline Properties dialog box.

The Add button in the Multiline Style dialog box adds a style to a drawing.

The Save button in the Multiline Style dialog box saves a style to a file.

The OK button in the Multiline Style dialog box closes the dialog box.

9.4.1.5 IMD Text 5: page 75

9.4.1.5.1 User actions-based functional descriptive text.

*Choosing Hatch from the Hatch flyout on the Draw toolbar under Windows opens the Boundary Hatch dialog box.
*Choosing Hatch from the Draw menu under DOS and UNIX opens the Boundary Hatch dialog box.
*Choosing Advanced under Boundary in the Boundary Hatch dialog box opens the Advanced Options dialog box.
Choosing Apply in the Boundary Hatch dialog box applies the hatch.

9.4.1.5.2 Object function-based functional descriptive text

The Hatch button from the Hatch flyout on the Draw toolbar under Windows opens the Boundary Hatch dialog box.
The Hatch button from the Draw menu under DOS and UNIX opens the Boundary Hatch dialog box.
The Advanced button under Boundary in the Boundary Hatch dialog box opens the Advanced Options dialog box.
The Apply button in the Boundary Hatch dialog box applies the hatch.

9.4.2 Bulgarian

9.4.2.1 IMD Text 1: pages 47-48

9.4.2.1.1 User actions-based functional descriptive text.

При избиране на Multiline Style от функционалния ред Object Properties или менюто Data под Windows се отваря диалоговият прозорец Multiline Styles.
При избиране на Multiline Style от менюто Data под DOS и UNIX се отваря диалоговият прозорец Multiline Styles.
При избиране на Add в диалоговия прозорец Element Properties се прибавя елемент.
При избиране на Color в диалоговия прозорец Element Properties се отваря диалоговият прозорец Select Color.
При избиране на Linetype в диалоговия прозорец Element Properties се отваря диалоговият прозорец Select Linetype .
При избиране на OK в диалоговия прозорец Element Properties се запазват характеристиките на елемента на мултилинията и се излиза от диалоговия прозорец Element Properties .

9.4.2.2 IMD Text 2: page 46

9.4.2.2.1 User actions-based functional descriptive text.

При избиране на Polyline от плаващото меню Polyline на функционалния ред Draw под Windows се стартира командата PLINE.
При избиране на Polyline от менюто Draw под DOS и UNIX се стартира командата PLINE.
При въвеждане на **a** (когато PLINE е стартирана) се превключва в режим Arc.
При въвеждане на **l** (когато PLINE е стартирана) се превключва в режим Line.

9.4.2.3 IMD Text 3: page 58

9.4.2.3.1 User actions-based functional descriptive text.

При избиране на 3 Points от плаващото меню Arc на функционалния ред Draw под Windows се стартира командата ARC.

При избиране на Arc и (след това) избиране на 3 Points от менюто Draw под DOS и UNIX се стартира командата ARC.

При въвеждане на **endp** и (след това) посочване на линия, така че дъгата да се захване за крайната точка на линията, когато е стартирана командата ARC, се задава крайната точка.

9.4.2.4 IMD Text 4: pages 48-49

9.4.2.4.1 User actions-based functional descriptive text.

При избиране на Multiline Style от менюто Data се отваря диалоговият прозорец Multiline Style.

При избиране на Multiline Properties в диалоговия прозорец Multiline Styles се отваря диалоговият прозорец Multiline Properties.

При избиране на Display Joints в диалоговия прозорец Multiline Properties се появява линия във върховете на мултилинията.

При избиране на On в подменюто Fill в диалоговия прозорец Multiline Properties се появява основният цвят.

При избиране на OK в диалоговия прозорец Multiline Properties се излиза от диалоговия прозорец Multiline Properties.

При избиране на Add в диалоговия прозорец Multiline Style към чертежа се добавя стил.

При избиране на Save в диалоговия прозорец Multiline Style стилът се записва във файл.

При избиране на OK в диалоговия прозорец Multiline Style диалоговият прозорец/ прозорецът се затваря.

9.4.2.5 IMD Text 5: page 75

9.4.2.5.1 User actions-based functional descriptive text.

При избиране на Hatch от плаващото меню Hatch на функционалния ред Draw под Windows се отваря диалоговият прозорец Boundary Hatch.

При избиране на Hatch от менюто Draw под DOS и UNIX се отваря диалоговият прозорец Boundary Hatch.

При избиране на Advanced от подменюто Boundary в диалоговия прозорец Boundary Hatch се отваря диалоговият прозорец Advanced Options.

При избиране на Apply в диалоговия прозорец Boundary Hatch се получава щриховката.

9.4.3 Czech

9.4.3.1 IMD Text 1: pages 47-48

9.4.3.1.1 User actions-based functional descriptive text.

V operačním systému *Windows* se vybráním *Multiline Style* na nástrojovém panelu *Object Properties* nebo v menu *Data* otevírá dialogové okno *Multiline Styles*.

V operačních systémech *DOS a UNIX* se vybráním *Multiline Style* v menu *Data* otevírá dialogové okno *Multiline Styles.*

Vybráním *Add* v dialogovém okně *Element Properties* se přidává element.

Vybráním *Color* v dialogovém okně *Element Properties* se otevírá dialogové okno *Select Color.*

Vybráním *Linetype* v dialogovém okně *Element Properties* se otevírá dialogové okno *Select Linetyper*.

Vybráním *OK* v dialogovém okně *Element Properties* se ukládá styl elementu multičáry a opuští se dialogové okno *Element Properties*.

9.4.3.1.2 Object function-based functional descriptive text

V operačním systému *Windows* příkaz *Multiline Style* na nástrojovém panelu *Object Properties* nebo v menu *Data* otevírá dialogové okno *Multiline Styles.*

V operačních systémech *DOS a UNIX* příkaz *Multiline Style* v menu *Data* otevírá dialogové okno *Multiline Styles*.

Příkaz *Add* v dialogovém okně *Element Properties* přidává element.

Volba *Color* v dialogovém okně *Element Properties* otevírá dialogové okno *Select Color.*

Volba *Linetype* v dialogovém okně *Element Properties* otevírá dialogové okno *Select Linetyper.*

Tlačítko *OK* v dialogovém okně *Element Properties* ukládá styl elementu multičáry a opuští dialogové okno *Element Properties*.

9.4.3.2 IMD Text 2: page 46

9.4.3.2.1 User actions-based functional descriptive text.

V operačním systému *Windows* se vybráním *Polyline* v plovoucím ikonovém menu *Polyline* na nástrojovém panelu *Draw* spouští příkaz *PLINE.*

V operačních systémech *DOS* a *UNIX* se vybráním *Polyline* v menu *Draw* spouští příkaz PLINE.

Zadáním *a* (když je/byl spuštěn příkaz *PLINE*) se přepíná do režimu kreslení oblouků.

Zadáním *l* (když je/byl spuštěn příkaz *PLINE*) se přepíná do režimu kreslení čar.

9.4.3.2.2 Object function-based functional descriptive text

V operačním systému *Windows* příkaz *Polyline* v plovoucím ikonovém menu *Polyline* na nástrojovém panelu *Draw* spouští příkaz *PLINE.*

V operačních systémech *DOS* a *UNIX* příkaz *Polyline* v menu *Draw* spouští příkaz PLINE.

(i) Když je/byl spuštěn příkaz *PLINE,* přepíná klávesa *a* do režimu kreslení oblouků.

(ii) Klávesa *a* přepíná do režimu kreslení oblouků.

(i) Když je/byl spuštěn příkaz *PLINE,* přepíná klávesa *l* do režimu kreslení čar.

(ii) Klávesa *l* přepíná do režimu kreslení čar.

9.4.3.3 IMD Text 3: page 58

9.4.3.3.1 User actions-based functional descriptive text.

V operačním systému *Windows* se vybráním *3 Points* v plovoucím ikonovém menu *Arc* na nástrojovém panelu *Draw* spouští příkaz *ARC*.

V operačních systémech *DOS* a *Unix* se vybráním *Arc* v menu *Draw* a následným vybráním *3 points* spouští příkaz *ARC*.

Když je/byl spuštěn příkaz *ARC*, zadáním *endp* a následným vybráním čáry se určuje koncový bod oblouku.

9.4.3.3.2 Object function-based functional descriptive text

V operačním systému *Windows* volba *3 Points* v plovoucím ikonovém menu *Arc* na nástrojovém panelu *Draw* spouští příkaz *ARC*.

V operačních systémech *DOS* a *Unix* příkaz *Arc* v menu *Draw* a následným vybráním *3 points* spouští příkaz *ARC*.

Když je/byl spuštěn příkaz *ARC*, *endp* a následné vybrání čáry určuje koncový bod oblouku.

9.4.3.4 IMD Text 4: pages 48-49

9.4.3.4.1 User actions-based functional descriptive text.

Vybráním *Multiline Style* v menu *Data* se otevírá dialogové okno *Multiline Style*.

Vybráním *Multiline Properties* v dialogovém okně *Multiline Styles* se otevírá dialogové okno *Multiline Properties.*

Vybráním *Display Joints* v dialogovém okně *Multiline Properties* se zobrazují čáry ve vrcholech multičáry.

Vybráním *On* pod *Fill* v dialogovém okně *Multiline Properties* se zobrazuje barva pozadí.

Vybráním *OK* v dialogovém okně *Multiline Properties* se opouští dialogové okno *Multiline Properties.*

Vybráním *Add* v dialogovém okně *Multiline Style* se styl přidává k výkresu.

Vybráním *Save* v dialogovém okně *Multiline Style* se styl ukládá do souboru.

Vybráním *OK* v dialogovém okně *Multiline Style* se dialogové okno zavírá.

9.4.3.4.2 Object function-based functional descriptive text

Příkaz *Multiline Style* v menu *Data* otevírá dialogové okno *Multiline Style*.

Příkaz *Multiline Properties* v dialogovém okně *Multiline Styles* otevírá dialogové okno *Multiline Properties.*

Volba *Display Joints* v dialogovém okně *Multiline Properties* zobrazuje čáry ve vrcholech multičáry.

Vokba *On* pod *Fill* v dialogovém okně *Multiline Properties* zobrazuje barva pozadí.

Tlačítko *OK* v dialogovém okně *Multiline Properties* opouští dialogové okno *Multiline Properties.*

Příkaz *Add* v dialogovém okně *Multiline Style* přidává styl k výkresu.

Příkaz *Save* v dialogovém okně *Multiline Style* ukládá styl do souboru.

Tlačítko *OK* v dialogovém okně *Multiline Style* zavírá dialogové okno.

9.4.3.5 IMD Text 5: page 75

9.4.3.5.1 User actions-based functional descriptive text.

V operačním systému *Windows* se vybráním *Hatch* v ikonovém menu *Hatch* na nástrojovém panelu *Draw* otevírá dialogové okno *Boundary Hatch*.

V operačních systémech *DOS* a *Unix* se vybráním *Hatch* v menu *Draw* otevírá dialogové okno *Boundary Hatch.*

Vybráním *Advanced* pod *Boundary* v dialogovém okně *Boundary Hatch* se otevírá dialogové okno *Advanced Options.*

Vybráním *Apply* v dialogovém okně *Boundary Hatch* se aplikuje šrafování.

9.4.3.5.2 Object function-based functional descriptive text

V operačním systému *Windows* příkaz *Hatch* v ikonovém menu *Hatch* na nástrojovém panelu *Draw* otevírá dialogové okno *Boundary Hatch.*

V operačních systémech *DOS* a *Unix* příkaz *Hatch* v menu *Draw* otevírá dialogové okno *Boundary Hatch.*

Volba *Advanced* pod *Boundary* v dialogovém okně *Boundary Hatch* otevírá dialogové okno *Advanced Options.*

Příkaz *Apply* v dialogovém okně *Boundary Hatch* aplikuje šrafování.

9.4.4 Russian

9.4.4.1 IMD Text 1: pages 47-48

9.4.4.1.1 User actions-based functional descriptive text.

Нажатием кнопки Multiline Style на панели инструментов Object Properties или в меню Data под Windows открывается диалоговое окно Multiline Styles.

Нажатием кнопки Multiline Style в меню Data под DOS and UNIX открывается диалоговое окно Multiline Styles.

selectAddprocНажатием кнопки Add в диалоговом окне Element Properties добавляется элемент.

Нажатием кнопки Color в диалоговом окне Element Properties открывается диалоговое окно Select Color.

Нажатием кнопки Linetype в диалоговом окне Element Properties открывается диалоговое окно Select Linetype.

Нажатием кнопки OK в диалоговом окне Element Properties сохраняется стиль элемента мультилинии и закрывается диалоговое окно Element Properties.

9.4.4.1.2 Object function-based functional descriptive text

Кнопка Multiline Style на панели инструментов Object Properties или в меню Data под Windows открывает диалоговое окно Multiline Styles.

Кнопка Multiline Style в меню Data под DOS and UNIX открывает диалоговое окно Multiline Styles.

selectAddproc Кнопка Add в диалоговом окне Element Properties добавляет элемент.

Кнопка Color в диалоговом окне Element Properties открывает диалоговое окно Select Color.

Кнопка Linetype в диалоговом окне Element Properties открывает диалоговое окно Select Linetype.

Кнопка OK в диалоговом окне Element Properties сохраняет стиль элемента мультилинии и закрывает диалоговое окно Element Properties.

9.4.4.2 IMD Text 2: page 46

9.4.4.2.1 User actions-based functional descriptive text.

Нажатием кнопки Polyline в палитре Polyline flyout на панели инструментов Draw под Windows запускается команда PLINE.

Нажатием кнопки Polyline в меню Draw под DOS and UNIX запускается команда PLINE.

Нажатие клавиши a (когда запущена команда PLINE) переключает в режим Arc.

Нажатие клавиши l (когда запущена команда PLINE) переключает в режим Line.

9.4.4.2.2 Object function-based functional descriptive text

Кнопка Polyline в палитре Polyline на панели инструментов Draw под Windows запускает команду PLINE.
Кнопка Polyline в меню Draw под DOS and UNIX запускает команду PLINE.
Клавиша a (когда запущена команда PLINE) переключает в режим Arc.
Клавиша l (когда запущена команда PLINE) переключает в режим Line.

9.4.4.3 IMD Text 3: page 58

9.4.4.3.1 User actions-based functional descriptive text.

Нажатием кнопки 3 Points в палитре Arc на панели инструментов Draw под Windows запускается команда ARC.
Нажатием кнопки Arc и затем нажатием кнопки 3 Points в меню Draw под DOS and UNIX запускается команда ARC.
Вводом endp (когда запущена команда ARC) и заданием линии, к конечной точке которой привязана дуга, задается конечная точка дуги.

9.4.4.3.2 Object function-based functional descriptive text

Кнопка 3 Points в палитре Polyline flyout на панели инструментов Draw под Windows запускает команду ARC.
Кнопка Arc и кнопка 3 Points в меню Draw под DOS and UNIX запускает команду PLINE.

9.4.4.4 IMD Text 4: pages 48-49

9.4.4.4.1 User actions-based functional descriptive text.

Нажатием кнопки Multiline Style в меню Data menu открывается диалоговое окно Multiline Style.
Нажатием кнопки Multiline Properties в диалоговом окне Multiline Styles dialog box открывается диалоговое окно Multiline Properties.
Нажатием кнопки Display Joints в диалоговом окне Multiline Properties линия отображается у вершин мультилинии.
Нажатием кнопки On в окне Fill в диалоговом окне Multiline Properties показывается цвет фона.
Нажатием кнопки OK в диалоговом окне Multiline Properties закрывается диалоговое окно Multiline Properties.
Нажатием кнопки Add в диалоговом окне Multiline Style стиль добавляется к рисунку.
Нажатием кнопки Save в диалоговом окне Multiline Style стиль сохраняется в файле.
Нажатием кнопки OK в диалоговом окне Multiline Style закрывается диалоговое окно.

9.4.4.4.2 Object function-based functional descriptive text

Кнопка Multiline Style в меню Data menu открывает диалоговое окно Multiline Style.

Кнопка Multiline Properties в диалоговом окне Multiline Styles dialog box открывает диалоговое окно Multiline Properties.

Кнопка Display Joints в диалоговом окне Multiline Properties отображает линию у вершин мультилинии.

Кнопка On в окне Fill в диалоговом окне Multiline Properties показывает цвет фона.

Кнопка OK в диалоговом окне Multiline Properties закрывает диалоговое окно Multiline Properties.

Кнопка Add в диалоговом окне Multiline Style добавляется стиль к рисунку.

Кнопка Save в диалоговом окне Multiline Style сохраняет стиль в файле.

Кнопка OK в диалоговом окне Multiline Style закрывает диалоговое окно.

9.4.4.5 IMD Text 5: page 75

9.4.4.5.1 User actions-based functional descriptive text.

Нажатием кнопки Hatch в палитре Hatch на панели инструментов Draw под Windows открывается диалоговое окно Boundary Hatch.

Нажатием кнопки Hatch в меню Draw под DOS и UNIX открывается диалоговое окно Boundary Hatch.

Нажатием кнопки Advanced в пункте Boundary в диалоговом окне Boundary открывается диалоговое окно Advanced Options.

Нажатием кнопки Apply в диалоговом окне Boundary Hatch применяет штриховку.

9.4.4.5.2 Object function-based functional descriptive text

Кнопка Hatch в палитре Hatch на панели инструментов Draw под Windows открывает диалоговое окно Boundary Hatch.

Кнопка Hatch в меню Draw под DOS и UNIX открывает диалоговое окно Boundary Hatch.

Кнопка Advanced в пункте Boundary в диалоговом окне Boundary открывает диалоговое окно Advanced Options.

Кнопка Apply в диалоговом окне Boundary Hatch применяет штриховку.

10. Overviews

10.1 English

10.1.1 Random ordering

The system enables you to create a multiline style, to specify the properties of a multiline, to draw a line and arc combination polyline, to draw an arc by specifying three points, and to define a boundary set in a complex drawing

10.1.2 Aggregation according to actee/action

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.

10.2 Bulgarian

10.2.1 Random ordering

10.2.1.1 Personal style

Системата Ви позволява да създадете стил на мултилиния, да зададете характеристиките на мултилиния, да начертаете полилиния, съставена от отсечки и дъги, да начертаете дъга по три точки, и да дефинирате област за щриховане в сложен чертеж.

10.2.1.2 Impersonal style

Системата позволява да се създаде стил на мултилиния, да се зададат характеристиките на мултилиния, да се начертае полилиния, съставена от отсечки и дъги, да се начертае дъга по три точки, и да се дефинира област за щриховане в сложен чертеж.

10.2.2 Aggregation according to actee/action

10.2.2.1 Personal style

Системата Ви позволява да създадете стил на мултилиния, и да зададете характеристиките на мултилиния, Вие можете също да начертаете полилиния, съставена от отсечки и дъги, и дъга по три точки.

10.2.2.2 Impersonal style

Системата позволява да се създаде стил на мултилиния, и да се зададат характеристиките на мултилиния, Възможно е също да се начертае полилиния, съставена от отсечки и дъги, и дъга по три точки.

10.3 Czech

10.3.1 Random ordering

10.3.1.1 Personal style

Systém Vám umožňuje, abyste vytvořili styl multičáry, (abyste) určili vlastnosti multičáry, (abyste) nakreslili oblouk určením tří bodů, (abyste) nakreslili úsečku složenou z čar a oblouku a (abyste) definovali hraniční množinu v komplexním výkrese.

10.3.1.2 Impersonal style

Systém umožňuje, vytvářet styly multičár, určovat vlastnosti multičár, kreslit oblouk určením tří bodů, kreslit úsečku složenou z čar a oblouku a definovat hraniční množinu v komplexním výkrese.

10.3.2 Aggregation according to actee/action

10.3.2.1 Personal style

Systém Vám umožňuje, abyste vytvořili styl multičáry a určili její vlastnosti. Můžete také nakreslit oblouk určením tří bodů, nakreslit úsečku složenou z čar a oblouku nebo definovat hraniční množinu v komplexním výkrese.

10.3.2.2 Impersonal style

Systém umožňuje, vytvářet styly multičá a určovat jejich vlastnosti. Dále je možné kreslit oblouky určením tří bodů, kreslit úsečky složené z čar a oblouku nebo definovat hraniční množinu v komplexním výkrese.

10.4 Russian

10.4.1 Random ordering

10.4.1.1 Personal style

Система позволяет Вам создавать стиль мультилинии, задавать свойства мултилинии, рисовать полилинии, состоящие из отрезков прямых и дуг, рисовать дугу по трем точкам, определять набор границ в сложном рисунке.

10.4.1.2 Impersonal style

Система позволяет создавать стиль мультилинии, задавать свойства мултилинии, рисовать полилинии, состоящие из отрезков прямых и дуг, рисовать дугу по трем точкам, определять набор границ в сложном рисунке.

10.4.2 Aggregation according to actee/action

10.4.2.1 Personal style

Система позволяет Вам создавать стиль мультилинии, и задавать свойства мултилинии. Вы можете также рисовать полилинии, состоящие из отрезков прямых и дуг, и дугу по трем точкам.

10.4.2.2 Impersonal style

Система позволяет создавать стиль мультилинии, и задавать свойства мултилинии. Можно также рисовать полилинии, состоящие из отрезков прямых и дуг, и дугу по трем точкам.

11. Table of Contents

11.1 English

TABLE OF CONTENTS

11.2 Bulgarian

СЪДЪРЖАНИЕ

11.3 Russian

СОДЕРЖАНИЕ

11.4 Czech

OBSAH