Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document

D-00-01

IMPACTS in Natural Language Generation
NLG Between Technology and Applications

Workshop at Schloss Dagstuhl, Germany
July 26-28, 2000

Tilman Becker, Stephan Busemann (eds.)

July 2000

Deutsches Forschungszentrum für Künstliche Intelligenz

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210
E-Mail: info@dfki.uni-kl.de

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341
E-Mail: info@dfki.de

WWW: http://www.dfki.de

Deutsches Forschungszentrum für Künstliche Intelligenz

DFKI GmbH

German Research Center for Artificial Intelligence

Founded in 1988, DFKI today is one of the largest nonprofit contract research institutes in the field of innovative software technology based on Artificial Intelligence (AI) methods. DFKI is focusing on the complete cycle of innovation — from world-class basic research and technology development through leading-edge demonstrators and prototypes to product functions and commercialization.

Based in Kaiserslautern and Saarbrücken, the German Research Center for Artificial Intelligence ranks among the important "Centers of Excellence" worldwide.

An important element of DFKI's mission is to move innovations as quickly as possible from the lab into the marketplace. Only by maintaining research projects at the forefront of science can DFKI have the strength to meet its technology transfer goals.

DFKI has about 115 full-time employees, including 95 research scientists with advanced degrees. There are also around 120 part-time research assistants.

Revenues for DFKI were about 24 million DM in 1997, half from government contract work and half from commercial clients. The annual increase in contracts from commercial clients was greater than 37% during the last three years.

At DFKI, all work is organized in the form of clearly focused research or development projects with planned deliverables, various milestones, and a duration from several months up to three years.

DFKI benefits from interaction with the faculty of the Universities of Saarbrücken and Kaiserslautern and in turn provides opportunities for research and Ph.D. thesis supervision to students from these universities, which have an outstanding reputation in Computer Science.

The key directors of DFKI are Prof. Wolfgang Wahlster (CEO) and Dr. Walter Olthoff (CFO).

DFKI's six research departments are directed by internationally recognized research scientists:

- Information Management and Document Analysis (Director: Prof. A. Dengel)
- Intelligent Visualization and Simulation Systems (Director: Prof. H. Hagen)
- Deduction and Multiagent Systems (Director: Prof. J. Siekmann)
- Programming Systems (Director: Prof. G. Smolka)
- Language Technology (Director: Prof. H. Uszkoreit)
- Intelligent User Interfaces (Director: Prof. W. Wahlster)

In this series, DFKI publishes research reports, technical memos, documents (eg. workshop proceedings), and final project reports. The aim is to make new results, ideas, and software available as quickly as possible.

Prof. Wolfgang Wahlster
Director

IMPACTS in Natural Language Generation
NLG Between Technology and Applications

Tilman Becker, Stephan Busemann (eds.)

DFKI-D-00-01

Contents

Introduction

The tension between theoretical work and its implementation has often been considered fruitful. In the field of Natural Language Generation (NLG), it is now complemented by another tension, the one between technologies and applications:

- "I have invented a new technique for NLG!" – "What is its impact on applications?"
- "I have built a new NLG application!" – "What is its impact on the technology?"

Much of NLG technology is based on a theoretical understanding of the process of language generation, whereas the applications[1] strongly rely on practical requirements. The technology offered by the field does not match up the needs of application programmers or customers. Most theoretically well-motivated technologies that cannot straightforwardly be employed within specific applications. The expectations of the customers often reach beyond what information technology can offer. Plug and Play technology is inherently difficult if the input changes drastically from one application to another, as is the case for NLG.

Some NLG application developers find it thus preferable to not reuse existing technology. This is often due to the lack of solutions for the knowledge bottleneck and for the input formation bottleneck: NLG technology lacks the power of dealing with external conceptual lexical knowledge bases, and it also lacks standards of representing inputs at a suitable specificity.

The "IMPACTS" workshop contributes towards bridging the gap between technological progress and the suitability of NLG systems for use in applications. The workshop addresses researchers and developers in NLG, as well as current and potential users of NLG applications.

The Programme Committee includes the following persons (in alphabetical order):

> John Bateman, University of Bremen, Germany; Tilman Becker, DFKI Saarbrücken, Germany (Program Co-Chair); Stephan Busemann, DFKI Saarbrücken, Germany (Program Co-Chair); Robert Dale, Language Technology Pty Ltd and Macquarie University, Australia; Laurence Danlos, LORIA, France; Michael Elhadad, Ben-Gurion University, Israel; Eduard Hovy, ISI, University of Southern California, USA; Richard Kittredge, CoGenTex Inc, USA; Inderjeet Mani, Mitre Corporation, USA; David D. McDonald, Brandeis University, USA; Cecile Paris, CSIRO Mathematical and Information Sciences, Macquarie University, Australia; Owen Rambow, AT&T, USA; Ehud Reiter, University of Aberdeen, UK; Donia Scott, ITRI, University of Brighton, UK.

The workshop consists of three types of sessions: presentation of submitted papers, an invited talk and four invited "burning issues" sessions. The papers in this volume are unpublished research reports reviewed by the Programme Committee. We invited original and unpublished contributions from all areas of NL generation relating to the workshop theme.

[1] For this workshop, we adopt a broad notion of application by including pieces of software containing NLG technology that currently are used by others in order to solve real-world tasks.

An important issue that is central to first three of the paper contributions is the acquisition of the various kinds of knowledge needed for NLG. The paper by Gloria De Salve, Berardina De Carolis and Fiorella de Rosis, entitled *"Image Descriptions from Annotated Knowledge Sources"* describes how annotations of radiological images are designed and used as the basis for natural language descriptions in the system ARIANNA.

In their paper *"'Instructing by doing:' Interactive graphics- and knowledge-based generation of instructional text,"* Manfred Stede and Holmer Hemsen propose a novel way of acquiring the knowledge-base for instructional text: Sequences of events are entered into the system by simulating them in a virtual 3D world.

The paper *"Where Do Instructions Come From? Knowledge Acquisition and Specification for Instructional Text"* by Keith Vander Linden, Cecile Paris and Shijian Lu is also concerned with instructional text and knowledge acquisition. They implemented a system that enhances the automatic acquisition with a tool (TAMOT) to edit and configure the extracted knowledge.

The work presented by Paul Piwek, Roger Evans, Lynne Cahill, Neil Tipper in *"Natural Language Generation in the Mile System"* is concerned with a different domain: a query-answering system. By using WYSIWYM ("What you see is what you mean") for formulating queries, they apply NLG in generating the answers as well as the queries. This is a prime example of a new technology (WYSIWYM) that has an impact on applications, namely highly complex queries.

Finally, Chris Mellish addresses a very important practical aspect in *"Understanding Shortcuts in NLG Systems"*. Since shortcuts, i.e. bypassing entire modules of a typical NLG system architecture, can be helpful or even necessary in constructing practical systems, it is important to clarify how and where they currently are used and can be used.

The invited talk *"Parsing to Text Structure: the basis of a reversible natural language generation system"* is given by David D. McDonald (Brandeis University). He addresses the input formation bottleneck, suggesting the derivation of NLG input from human language text within an architecture that combines parsing and generation.

In order to implement our concept of a discussion-intensive event, we invited "burning issues" presentations taking up the main topic of the workshop. They initiate interesting discussions, possibly within smaller groups, and last two hours up to half a day. The topics deal with include

- the reusability of modules within NLG systems (chaired by Chris Mellish, University of Edinburgh). Steps towards interface standards and towards developing modules that can be used for a variety of tasks are a prerequisite for the quick and reliable development of applications.
- the opportunities and limits of statistics-based NLG (chaired by Eduard Hovy, ISI, Los Angeles). Statistics-based generation can possibly support solutions to the knowledge bottleneck with the help of machine learning technology.
- the relation between text summarization and NLG (chaired by Daniel Marcu, ISI, Los Angeles). Summarization is a highly topical application task, but has, until now, mostly

been dealt with at the text string level rather than by using NLG technology.

- methods of using corpora and criteria to evaluate NLG systems (chaired by Srinivas Bangalore, AT&T Research, Florham Park). Evaluating generation systems is urgently needed for customers to assess the available functionality and coverage, and for system comparison.

"IMPACTS" was approved by the Special Interest Group for NLG (SIGGEN) as a "SIGGEN NLG Workshop." SIGGEN (`http://alcweb.org/siggen/`) will support the permanent electronic publication of the workshop results.
The workshop's homepage is `http://impacts.dfki.de`.

Tilman Becker
Stephan Busemann

Workshop Program

Tuesday, 25 July 2000

18:00 **Arrival, Reception, Dinner**

Wednesday, 26 July 2000

08:30 **Introduction**

08:40–09:20 **Keith Vander Linden, Cecile Paris, and Shijian Lu:**
Where Do Instructions Come From? Addressing the Problem of Knowledge Acquisition in the Context of Instructional Text

09:20–10:00 **Manfred Stede, Holmer Hemsen:**
"Instructing by doing": Interactive graphics- and knowledge-based generation of instructional text

10:00 **Coffee Break**

10:30–11:10 **Gloria De Salve, Berardina De Carolis, Fiorella de Rosis:**
Image Descriptions from Annotated Knowledge Sources

11:10–11:50 **Paul Piwek, Roger Evans, Lynne Cahill, Neil Tipper:**
Natural Language Generation in the Mile System

11:50–12:30 **Chris Mellish:** *Understanding Shortcuts in NLG Systems*

12:30 **Lunch Break**

15:00–18:00 *Burning Issue Discussion, chaired by*
Chris Mellish, University of Edinburgh:
What are reusable modules for NLG?

Thursday, 27 July 2000

08:45–12:00 *Burning Issue Discussion, chaired by*
Eduard Hovy, USC/ISI, Los Angeles:
The Opportunities and Limits of Statistics-Based Generation

12:00 **Lunch Break**

14:30–15:30 *Invited Talk:*
David McDonald, Brandeis University:
Parsing to Text Structure: the basis of a reversible natural language generation system

15:30 **Coffee Break**

15:45–18:00 *Burning Issue Discussion, chaired by*
Daniel Marcu, USC/ISI, Los Angeles:
Summarization and Generation

Friday, 28 July 2000

08:45–10:45 *Burning Issue Discussion, chaired by*
Srinivas Bangalore, AT&T Labs - Research, Florham Park NJ:
Corpora, Evaluation and Generation

10:45 **Coffee Break**

11:15–12:30 **Burning Issues Panel Discussion**

12:30 **Lunch, Departure**

Where Do Instructions Come From? Knowledge Acquisition and Specification for Instructional Text

Keith Vander Linden[1], Cécile Paris[2], and Shijian Lu[2]

[1] Department of Computer Science, Calvin College,
Grand Rapids, MI 49546, USA
kvlinden@calvin.edu
[2] CSIRO, Mathematical and Information Sciences,
Locked Bag 17, North Ryde, NSW 1670, AU
{Cecile.Paris, Shijian.Lu}@cmis.csiro.au

Abstract. Instructional text, because it is a useful and relatively constrained sub-language, has been a popular target for research-oriented generation systems. This work has demonstrated that existing technology is adequate for generating draft instructions; the problem, as is typical of generation work in general, has been with the acquisition of domain and lexicogrammatical knowledge. This acquisition task is a formidable barrier to the practical use of generation technology. The Isolde project attempts to address this problem by extracting parts of the required knowledge from existing models and by building tools to tailor what is extracted into a form suitable for generation.

1 Introduction

A number of research systems have been successful at generating drafts of user-oriented, procedural instructions (e.g., Mellish/Evans [7], COMET [6], TechDoc [14], Drafter [9,10], WYSIWYM [11], WIP [16], SPIN [3]). It is, therefore, clear that current generation technology is adequate for generating drafts of the sorts of instructions found in simple user's documentation. These systems have not, however, worked their way into commercial applications. The basic reason for this is that the acquisition and specification of the input knowledge required by these systems is too difficult. In general, it may be more difficult than simply entering the text by hand, or in the case of multilingual instructions, of entering the text by hand and then translating it automatically.

Some of the systems mentioned above, particularly the earlier ones, expect the user to hand-code the domain knowledge in an internal format. Other more recent systems (e.g., Drafter, WYSIWYM) have developed user interfaces of various sorts that facilitate the input of this knowledge. While these interfaces have proven to be usable and useful, they have been hard to apply in practice. First of all, they require coding of

all the domain knowledge in a format that is specific to the generation system, and thus not useful for any purpose other than generation. While the representation could be reused when generating future drafts of the instructions, it is not useful for other purposes such as interface analysis or system design. This makes it harder to justify the effort required to code it in the first place. Second, these systems tend to assume that the relevant lexical knowledge is predefined. Thus, new lexical items must be entered when moving to a new domain.

In contrast, successful generation systems for other sub-languages (e.g., Météo [2]) operate in domains in which suitable input representations are available because they are already built for other reasons. These sub-languages also tend to have restricted lexicogrammatical resources that can be predefined. Neither of these is the case for the instructional sub-language. Pre-built models do exist, but they are neither complete nor universal, and the domains for instructions are not restricted, so while the grammatical resources can be predefined, the lexical resources cannot.

The Isolde project has attempted to identify those pre-built resources that do exist in the common practice of user interface design, and to extract as much of the necessary domain and lexical knowledge from them as possible. To support this we, we have:

- built tools that extract knowledge from a number of types of existing models, including those built using the Unified Modeling Language (UML [13]),
- built a tool that supports the construction of task models - These models serve both as a collection point for the knowledge we extract from other system models, and as representations useful in their own right to interface designers,
- configured an instruction generation system to produce text drafts, allowing the user to modify the knowledge base that underlies any incorrect expressions.

This paper will describe each of these functions in turn, in the context of an extended example. It will pay particular attention to the difficulties we have faced in reusing knowledge produced for other purposes, and will attempt a preliminary quantification of how useful the existing models are for generation. It will then draw some general conclusions for language generation applications.

2 Extracting the Knowledge Required for Instructions

This section will discuss the two basic types of knowledge required by an instruction generation system: domain knowledge and linguistic knowledge. The required content and structure of this knowledge is well-known. It is typically represented in a standard slot-filler knowledge base format, in which the actions are arranged in a procedural, plan-goal hierarchy, and the objects are linked to the actions as actors, actees, etc. (cf. Drafter). We focus here on identifying the commonly used system models from which part of this knowledge can be extracted.

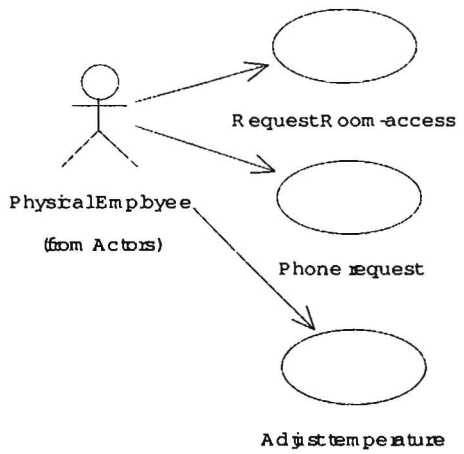

Figure 1. A portion of the UML Use Case diagrams for BMS

2.1 Domain Knowledge

The domain knowledge required for procedural instructions comprises: (1) the objects used in the domain of application (e.g., "rooms" are manipulated by building maintenance systems); (2) the graphic objects presented by the user interface (e.g., the "file menu" on a window application); and (3) the tasks that are performed on these objects (e.g., the user can "secure" a room). One source of this information is the UML system models produced software engineers [12], which provide:

- *use case diagrams* - A use case identifies a thread of potential use for the system to be constructed. It also specifies the appropriate user type for that case.
- *class diagrams* - These contain hierarchical descriptions of the classes that will be used in the software itself.
- *interaction diagrams* - These diagrams describe the sequence of interactions between objects that take place in the execution of a use case.

We will now describe how each of these models can contribute important elements for the knowledge base. We will illustrate them using BMS, a model for a building maintenance system developed by Esprit Systems (distributed freely by Rational).

UML Use Case Diagrams. Figure 1 shows three use cases for BMS. We can see that the object PhysicalEmployee (an instance of Actor or User) can achieve 3 high-level goals. Clearly, this sort of diagram is a source of user types and of high-level user goals with respect to the system (see domain knowledge type (3) above). The primary problem here is that the names of the actors and the spellings of the verbs

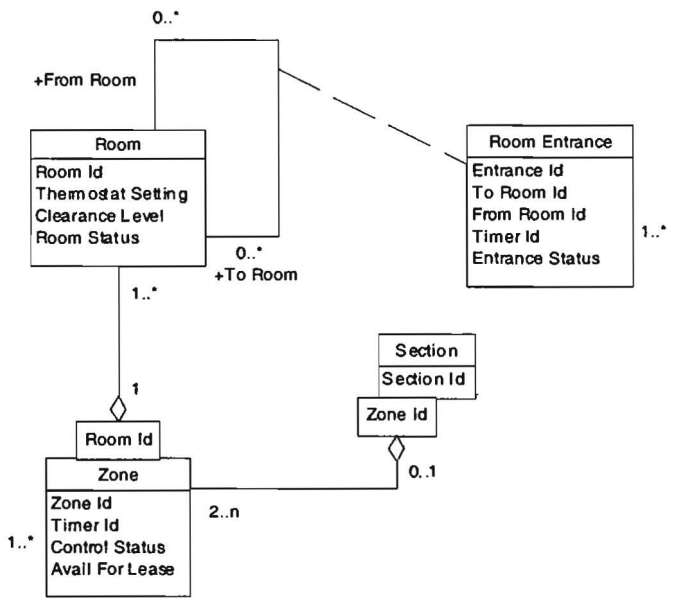

Figure 2. A portion of the UML Class diagrams for BMS.

and objects is dependent upon what the software engineer chooses to say in the diagram. We have little control over this. As a default, our extraction mechanism (implemented in Rose's internal scripting language) takes the exact spelling of the actor and produces an instance of the user concept from it. It also produces high-level tasks by parsing the names of the use cases, assuming that the action is the first word, the actee is the second, and the remainder of the words are other adjuncts.

UML Class Diagrams. Figure 2 shows a portion of the class diagram for the BMS system. In addition to being useful in building the BMS system, classes such as room and section are also destined to be objects referred to in the procedural instructions (knowledge type (1)). Our extraction mechanism, therefore, creates domain objects for each class and attribute using their identifier as a default lexical spelling.

UML Interaction Diagrams. Figure 3 shows the interaction diagram corresponding to the process of adjusting the temperature for a room with the BMS system. Here, the sequence of events that instantiate a use case are presented in temporal order (from top to bottom). Each event specifies the object or agent that initiates the event (the source of the arrows), and the object that is acted upon (the destination of the arrows). This diagram specifies the user and system actions that are typically expressed in instructions, but it also includes a number of internal programming details that are not relevant for the end user. For this information to be appropriate for instruction generation, we must filter out the inappropriate details. This filtering is done by a set of content selection heuristics which extract all user initiated actions, and the last system action in any sequence of system actions [5].

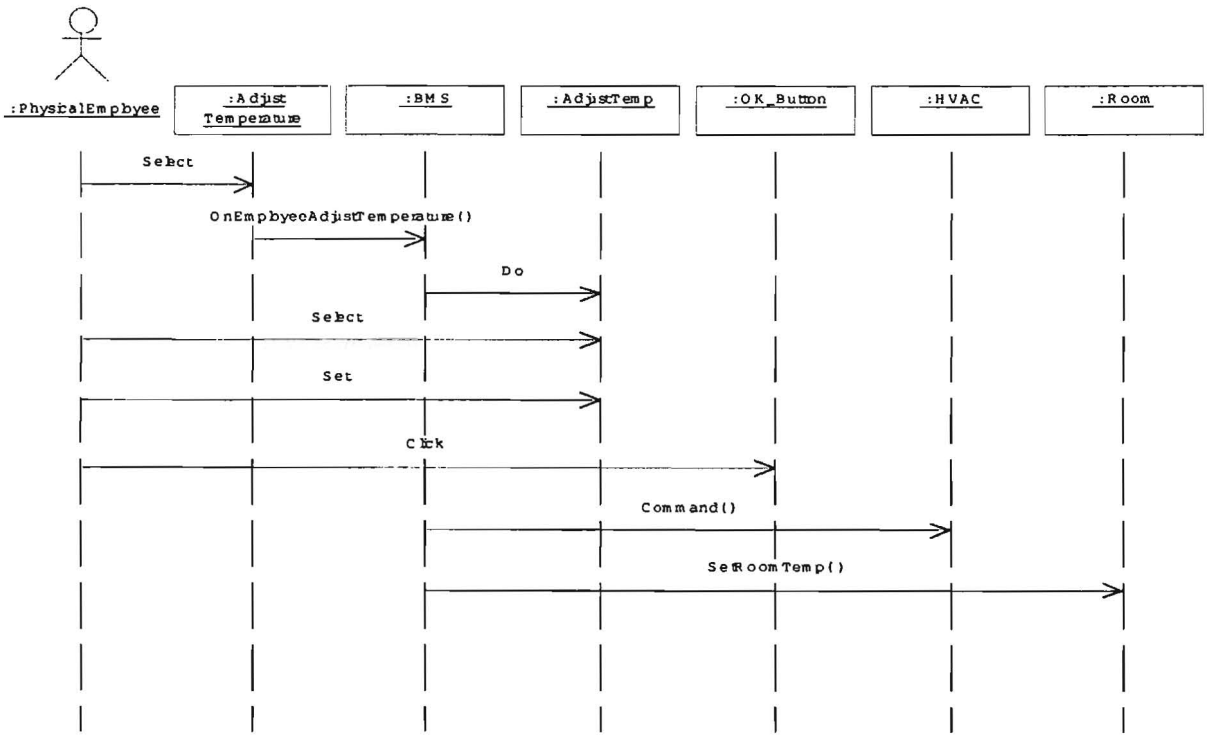

Figure 3. A portion of the UML Interaction Diagrams for BMS

As we will see below, these three UML diagrams provide the knowledge required to generate procedural instructions with very little intervention by either the interface designer or the technical author. We estimate that UML models can, *in principle*, be used to generate up to 70% of the user task model.[1] Unfortunately, this is due largely to the fact that we modified the original BMS model for generation purposes. The original BMS model used inappropriate names and included neither the GUI object knowledge (type (2)) nor the scenario diagrams (parts of type (3))[2] *In practice*, UML models can be counted on to provide only the user types and goals (from use-case diagrams) and the domain objects (from class diagrams).

2.2 Linguistic knowledge

The linguistic knowledge required by instruction generation includes the lexical resources used to express the relevant objects and tasks, and the grammatical and dis-

[1] This is based informally on our experience with the BMS system

[2] A better source for some of this knowledge is the GUI object and event recorders commonly used in interface design work. We are currently incorporating one of these tools.

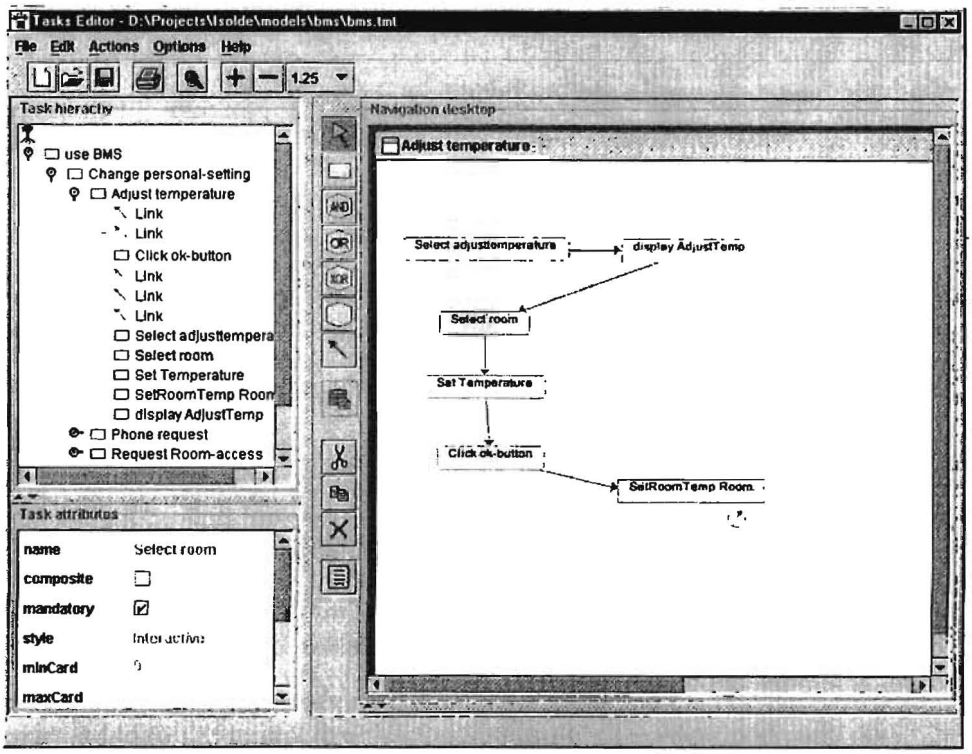

Figure 4. The adjust-temperature task decomposition in TAMOT

course knowledge used to fashion instructional sentences and texts. As is the case with typical instruction generation systems, we are able to hard-code this information because it tends to be constant from one instructional domain to the next. The lexical knowledge, on the other hand, does change from one domain to another. To help alleviate this, our knowledge extraction mechanism infers lexical spellings for the domain knowledge elements that it extracts. Although these spellings may be incorrect, they can be used to generate a first draft of the instructions, at which time the technical author can fix them up.

3 A Task Model Editor

The knowledge that we were able to extract from the UML models created in Rational Rose is useful, but, as we saw in the previous section, potentially incorrect or incomplete. We have, therefore, built TAMOT (see figure 4), a Java tool that allows a user to configure the extracted knowledge as an input for the generation system. This representation is not unlike those typical of instruction generation systems, except that it takes the form of an enhanced *task model*, represented in Diane+ [15]. Task models are procedural models of human tasks, goals and system responses. A key feature of

these models is that they can serve not only as adequate inputs to the instruction generation system, but also as useful representations for user interface design and analysis. Indeed, TAMOT is currently being used for this latter purpose by interface design consultants in our group. Thus, the construction of the task model is useful not only to drive the generation process, but also to aid in the design of the interface.

On the right-hand side of figure 4 we see a simple task decomposition for the user task of adjusting the temperature of a room. The oval-cornered boxes indicate user actions while the rectangular boxes indicate system actions. The arrows indicate the sequence in which the tasks must be executed. The interface designer is able manipulate the model by adding tasks to the Navigation Desktop, dragging them around, or by selecting them and editing their properties in the Task attributes window (on the lower left). The interface also presents an hierarchical view of all the tasks (on the upper left). The tasks shown in the figure were all derived automatically from the BMS model, though we did have to format the tasks in a more readable way and remove one system task that would not have been helpful to the user. The representation of each of these tasks is linked to the representation of the domain knowledge required to express the task in the generated instructions. This domain knowledge was also derived from the UML model based on a simple parsing of the UML object names (described above).[3]

As we will see in the next section, the task and domain knowledge, taken together, are capable of generating procedural user instructions. We estimate that, *in principle*, such knowledge can support from 50 to 100% of the procedural portions of user instructions, depending upon the complexity of the procedures.[4] *In practice*, the support for generation is better for task models than for UML models. Because interface designers tend to operate with a user-oriented view, the task models they produce are much more amenable to generating instructions. We estimate that with a more sophisticated parsing mechanism and more complete generation facilities we could generate useful instructions from nearly all of the elements of realistic task models.[5] The problems we've had in generating from the task models are with inconsistent naming conventions, the use of preconditions/feedback (which we currently do not support), and with occasional complicated task representations that are hard to linearize into text. With the upgrades to the system we have mentioned, we believe that we can approach the "in principle" level of 50-100% support for procedural instructions from realistic task models. We also believe that we can support much of the remainder of the text using the canned text facility supported by the Isolde generator.

---

[3] We are currently investigating the use of a more sophisticated controlled language parser.

[4] This is based on a corpus study of 43 pages of documentation for an on-line phone application from Ericsson. Approximately one-half of this documentation was procedural.

[5] This is based on an analysis of 5 task models. The task models were created using TAMOT by 2 interface designers for 4 different applications. The models included a total of 279 tasks.

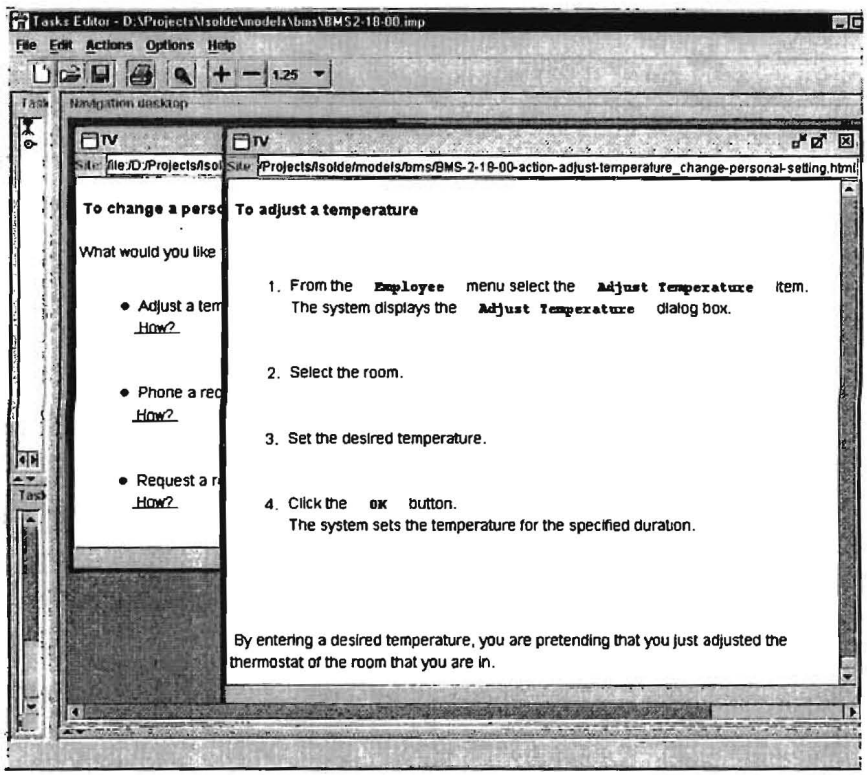

Figure 5. A portion of the hypertext output as displayed in TAMOT

4 Instruction Generation

The generation system is configured as a separate server (implemented in Lisp). It includes: (1) the Moore and Paris text planner [8]; (2) a new sentence planner implemented with extensions to the text planner; and (3) the KPML development environment for tactical generation [1]. We are currently attempting to quantify the usefulness of these "deep" generation tools in this domain. Our primary concern in this paper is with the errors that crop up in the output text due to the fact that the original knowledge sources were not hand-crafted for generation. When there are problems, such as an infelicitous name retained from the UML model, or perhaps some problem with parsing the short texts produced by the technical authors in the task model, the technical author will see them in the draft texts produced by the generator and will be allowed to modify the domain/task knowledge from which they came.

We implement this "fix it when it's broken" approach using a mouse-sensitive, hypertext display buffer (see figure 5) which operates just like a hypertext browser ex-

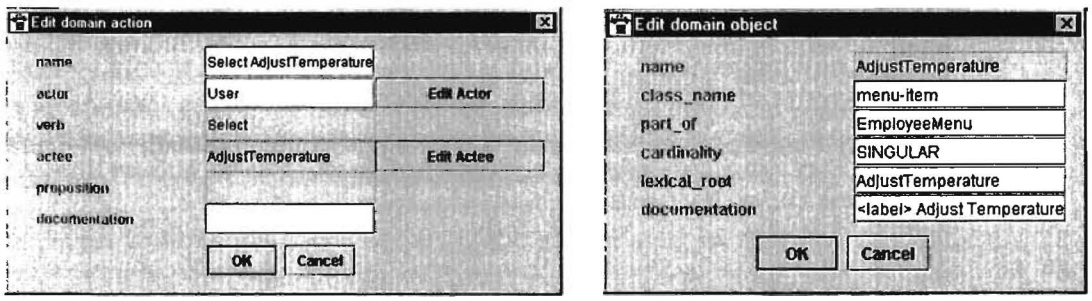

Figure 6. Domain Action and Domain Object Editing Dialog boxes

cept that all the expressions are mouse sensitive. When the technical author finds an expression that is incorrect, they click on it to get a set of dialog boxes that allow them to edit the domain model entities from which the text was generated (see figure 6). The domain action, shown on the left of figure 6 is linked to its case role fillers via the "Edit Actor" and "Edit Actee" buttons. The editing dialog box for the actee is shown on the right. The technical author is able to use these dialog boxes to modify the domain and lexical knowledge used to drive the generation process. They can, for example, change the lexical spelling of the verb or change the actee altogether, and then regenerate the text, iterating until the text is what they want. A portion of the final output of our example is shown in Figure 5.

5 Conclusions

This paper has discussed some practical issues involved in fielding instructional text generation systems. Current generation technology is adequate for generating instructions, but the knowledge resources required as input to the process must be extracted, as much as possible, from existing sources. A PC-based implementation of the Isolde system was discussed as an example of an approach to this problem. Some preliminary estimates of system coverage were given. Though not conclusive, they do suggest that such a system could indeed be a practical way to produce portions of user-directed, procedural instructions.

We identify two basic conclusions from this continuing work. First, to be practical, a generation project must take pains to find readily available sources of input knowledge. Assuming that an author will be willing to manually input all or most of the resources by hand is probably unacceptable. This paper presents one approach to doing this in the context of instructions. Second, given that external resources will probably not be enough to drive the generation process completely, one must build a tool that allows an author to configure the resources that are extracted. It would be best if this tool supported a generally useful modeling language rather than a generation specific one. TAMOT is an example of such a tool.

Acknowledgements. This work is supported by ONR grants N00014-99-1-0906 and N00014-97-0064, Calvin College, and the CSIRO. The authors acknowledge a debt to the legacy of the Drafter project undertaken at the ITRI, University of Brighton. They also acknowledge the involvement of Sandrine Balbo, Michael Brasser, Thomas Lo, and Nadine Ozkan.

References

1. Bateman J. A.: Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, 3(1) (1997) p. 15-55
2. Kittredge, R., Polguere, A.: Generating extended bilingual texts from application knowledge bases, *Proceedings of the International Workshop on Fundamental Research for the Future Generation of Natural Language Processing,* Kyoto, Japan (1991) 147-160
3. Kosseim L. and Lapalme G. : Content and rhetorical status selection in instructional text. In *Proc. of the 7th Int. Workshop on NLG,* Kennebunkport, ME (1994)
4. Lavoie, B., Rambow, O., Reiter, E.: Customizable Descriptions of Object-Oriented Models. In *Proceedings of the Fifth Conference on Applied Natural Language Processing,* Washington, DC, (1997) pp. 265-268.
5. Lu, Shijian, Paris, C., Vander Linden, K.: Towards the Automatic Construction of Task Models from Object Oriented Diagrams, in S. Chatty and P. Dewan (eds.), Proceedings of the IFIP Working Conference on Engineering for Human-Computer Interaction (1998)
6. McKeown, K., Elhadad, M., Fukumoto, Y., Lim, J., Lombardi, C., Rogin, J., Smadja, F.: Natural language generation in COMET, *Current Research in Natural Language Generation,* chapter 5, Academic Press (1990)
7. Mellish, C., Evans, R., Natural language generation from plans, *Computational Linguistics,* 15(4) (1989) 233-249
8. Moore J. and Paris C.: Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*}, 19(4) (1993) 651--694
9. Paris C. and Vander Linden K.: Drafter: An interactive support tool for writing multilingual instructions. *IEEE Computer,* 29(7) (1996) 49-56, July
10. Paris, C., Vander Linden, K., Fischer, M., Hartley, A., Pemberton, L., Power, R., and Scott, D.: A support tool for writing multilingual instructions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence,* August 20-25, Montreal, Canada, (1995) Pages 1398-1404.
11. Power, R., Scott, D., and Evans, R.: What You See is What You Meant: direct knowledge editing with natural language feedback, *Proceedings of the 13th European Conference on Artificial Intelligence,* ECAI'98, Brighton, UK, August 1998
12. Pressman, R: *Software Engineering* A Practitioner's Approach, 4th ed, McGraw Hill (1997)
13. Rumbaugh, J., Jacobson, I., Booch, G.: *The Unified Modeling Language Reference Manual,* Addison-Wesley, Reading, MA, (1999)
14. Rösner and M. Stede: TECHDOC: A system for the automatic production of multilingual technical documents.In *Proc of KONVENS-92,* Berlin. Springer (1992)
15. Tarby, J-C. and Barthet, M-F. The DIANE+ Method. In *Proceedings. of the 2nd International Workshop on Computer-Aided Design of User Interfaces* (1995)
16. Wahlster, W, Andre, E, Finkler, W, Profitlich, H, Rist, T:"Plan-based integration of natural language and graphics generation",*Artificial Intelligence,* 63(1-2) (1993) 387-428

"Instructing by doing:" Interactive graphics- and knowledge-based generation of instructional text

Manfred Stede and Holmer Hemsen

Technische Universität Berlin, FB Informatik, Projektgruppe KIT,
Franklinstr. 28/29, 10587 Berlin/Germany,
{stede|hemsen}@cs.tu-berlin.de

Abstract. Much research has been conducted on applying natural language generation to the creation of technical documentation. A critical issue for such applications is supplying the input: How does the technical writer interact with a system to produce representations that can be processed by a generator? In this paper, we explore the possibility of interaction with a virtual reality as a means to produce the kernel of such representations; this needs, however, to be augmented with other techniques in order to account for those portions of instructional text that do not directly relate to physical actions.

1 Introduction and related work

Automatically generating instructional text (e.g., as a part of technical manuals) has become a popular application for text generation, as it offers a number of distinct advantages: text can be produced in multiple languages; regular updates can be created without manual re-writing and re-translating; existing data and knowledge sources can possibly be integrated into the document production process; last but not least, the language found in technical documents is typically not too complicated to impair their automatic generation. The critical issue, however, is in supplying the input to such a system: In what way does the human (co-) author of the technical document interact with the system to produce the desired text in an effective manner? Previous research has suggested menu-based interfaces (e.g., in TECHDOC [Rösner, Stede 1994] and DRAFTER [Hartley, Paris 1997]) and incremental text-template filling (WYSIWYM, [Power, Scott 1998]). In this paper, we explore a new option: interactive manipulation in a 3D graphical environment. While mixing text and graphics in the instruction generation *output* has been realized in several systems (e.g., in WIP/PPP [André 1997] or VISDOK [Hartmann et al. 1998]), graphics has to our knowledge not yet been applied on the *input* side.

The idea of our approach is that the 'author' manipulates objects on the screen using the mouse (for the time being), puts them together to create composite objects, etc. A symbolic knowledge base monitors the graphical activities and classifies them as conceptual 'actions'. In an aggregation step, individual

actions are joined to form complex action representations. These are the input to the verbalization component, which maps the conceptual representations first to sentence-semantic specifications, and then to linguistic utterances. — This scenario is not unlike those of systems producing descriptions of image data, such as NAOS [Novak 1987] or SOCCER [André et al. 1988]. They also employ domain knowledge to identify elementary actions and "chunk" them into linguistic descriptions. In contrast to these systems, however, our input is the concrete manipulation data; extracting relevant changes in image data is thus not a primary concern. As another point of contrast, we are interested in multilingual output and, eventually, in combining graphics-input with other modes of user interaction in the production of instructional text.

At present, we have implemented a first prototype intended as "proof of concept", which thus illustrates the basic functionality. Its architecture is described in section 2. We illustrate the approach with our implemented pilot application, a construction kit, in section 3. Specifically, we describe in detail the events leading to the construction of an axle; this would in a more complete implementation be a part of building some kind of vehicle. In section 4, we discuss the advantages and disadvantages of our approach, explore possibilities for extending it to more complicated texts, and hint at some directions for practical applications of the approach. In particular, we suggest to fuse our approach with the WYSIWYM method in order to account for passages of instructional text that go beyond descriptions of physical activities, and thus are not immediately amenable to graphical input. Ultimately, we therefore view our approach as one part of an "author's workbench", where interactive graphics can help producing the raw text that needs to be further processed with appropriate tools.

2 System architecture

The architecture of the prototype is shown in figure 1. While the user manipulates objects in the 3D environment, a stream of *elementary events* is produced and written to a file. The events are in the format of assertions in the description logic LOOM [MacGregor 1991]. When the user initiates text generation, a segmentation module reads the event-file, performs aggregations and maps it to a sequence of action representations in the format of 'SitSpecs' [Stede 1999]. This process is driven by the LOOM knowledge base (KB), which holds the knowledge about the level of abstraction desired for verbalizing the activities. The SitSpecs are converted to sentence-semantic specifications (SemSpecs) using the MOOSE module [Stede 1999], and SemSpecs are finally turned into English or German sentences by the KPML generator [Bateman 1997].

2.1 Interactive graphics

The graphics module is implemented in Java-3D [Sowizral et al. 1998]. With the mouse, the user can

1. enter a new object into the world, choosing it from a menu,

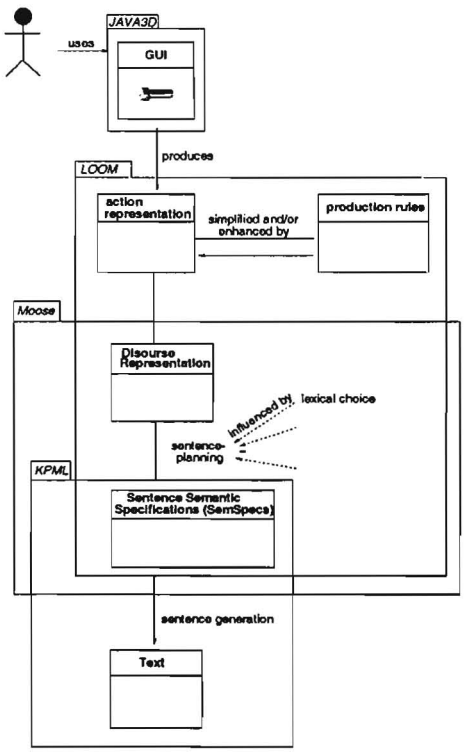

Fig. 1. System architecture

2. move an object to a new location,
3. turn an object around,
4. connect an object to another one.

The set of objects that can be introduced to the world (1) is determined by a menu, and thus the type of each new object in the world is fixed. This provides the link to the Loom KB: For a new object, a LOOM instance of the respective type is created.

Items (2) and (4) go beyond the level of straightforward graphical manipulation: (2) needs to detect collisions, i.e., ensure that an object is not moved "through" some other object. (4) needs to check whether the two objects involved can indeed be connected, i.e., whether they are nut and bolt or some other suitable pair. We decided to handle both tasks by the same mechanism. For a start, Java-3D offers a "built-in" collision detection, which notices a topological overlap between two objects (or, alternatively, their bounding boxes). Whenever this condition is triggered, we perform a deeper analysis of the topological relationship between the two objects in order to determine whether the user is likely to intend a connection between them. The conditions depend on the specific pair; for instance, if the tip of a screw collides with a wheel close

to the hole in the middle, and the angle between wheel and screw is close to 90 degrees, we surmise that the user intends to move the wheel over the screw. As soon as the collision as well as the additional conditions have been detected, the user is prompted to either confirm or reject the connection (in case the collision was not on purpose). Upon confirmation, the system completes this move and arranges the parts in their final position. For illustration of the results of the graphical manipulations, see figure 3, which will be explained in section 3.

2.2 Knowledge base

A key idea in our approach is to realize a close connection between the graphical representation of the "world" on the one hand and a symbolic representation of this world within a description logic on the other hand, and to exploit the power of automatic concept classification. For these purposes, we use the LOOM language and classifier [MacGregor 1991].

The terminological part of the knowledge base (Tbox) holds the knowledge about the various types of objects and their properties, and the possibilities for connecting them: Nuts can be connected to bolts, liquids can be put into a container, etc. The assertional part (Abox) is a symbolic representation of the state of the world and the changes that occur; there is an instance for each object in the world, and the connections between objects are modelled via LOOM relations. These are explicitly asserted when the user performs a connect-event in the 3D world. Importantly, as a result of a new connection-role, the LOOM classifier can automatically determine new type information. An example involving the classification of an assembled axle follows below. In other scenarios, re-classification can also occur when some other attribute of an object changes, e.g., when the user flips a switch or turns a knob.

In analogy, a sequence of actions can be automatically classified as a "meaningful" macro-action. For instance, the sequence of elementary actions shown in figure 2 can be classified as a 'connect' macro action with the connector being the screw and the connectee the ring; the concept definition of the macro action contains a sequence of elementary actions of appropriate types (which abstract from the topological details that are not relevant for the classification).

While some aggregations can be performed by the classifier automatically, others do not lend themselves to being formulated as a complex concept; for these cases we use LOOM production rules to trigger additional inferences.

2.3 Text generation

Supported by the automatic classifier as just described, the first step of the text generation process consists of "chunking" the sequence of elementary actions into a text plan, which involves the well-known task of aggregation (e.g., [Dalianis 1996]). At the moment, we are using only a fairly simple aggregation module that is geared to the scenario of our pilot application, to be explained in the next section.

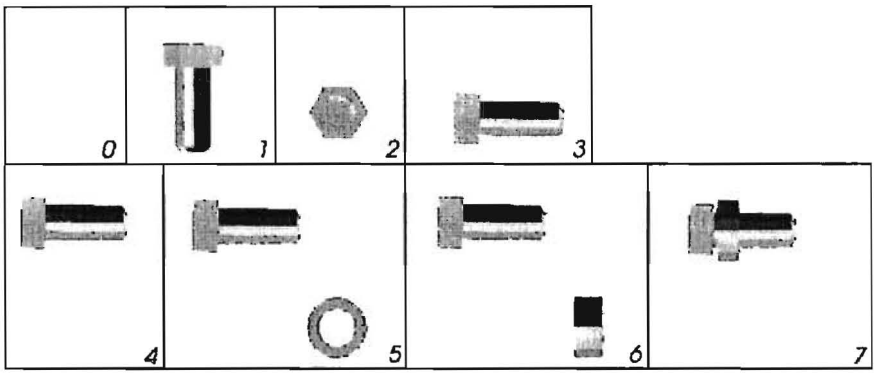

Fig. 2. Sequence of elementary actions in the Java-3D world

The text plan is also represented in LOOM and follows the format of 'SitSpecs' as used in the MOOSE generator [Stede 1999]. MOOSE converts the SitSpec into a sequence of sentence-semantic specifications; these are language-specific, lexicalized structures that are in the final step converted to linguistic utterances (in either English or German) by KPML [Bateman 1997]. MOOSE, originally a single-sentence generator, is currently being upgraded to produce complete paragraphs of text; thus it will accomplish sentence planning tasks such as determining sentence boundaries and structure, and choice of referring expressions. The input to the system can therefore be either an individual SitSpec, or a rhetorical tree (in the spirit of RST [Mann, Thompson 1988]). As the following section will show, however, the sentence planner at present is still in a very preliminary stage.

3 Example: Constructing an axle

Our pilot implementation of the framework (documented in [Hemsen 2000]) deals with a 'construction kit' with a set of parts that can be assembled into various mechanical objects. (This scenario was also used by [Wachsmuth, Jung 1996]). In the following, we describe the example of construcing a vehicle axle, composed of two wheels, rings and screws, and a cube holding them together.

In the Java-3D world, the user enters the various objects and moves them close together so that the system can infer the intended connections. The sequence of elementary actions shown in figure 2 is one possible beginning of the activity. Abstracting from the movement events, the initial sequence of elementary actions is as follows. Two objects are introduced, which prompts the creation of two Loom instances (recall that their type is associated with the menu options):

```
(createm 'HexaScrew@11d1c62 'hexagon-headed)
(createm 'Ring@1778fcd 'ring)
```

When the ring has been moved to the screw, and the system detects that the preconditions for a connection are fulfilled, the corresponding event instance is created, together with two location states, which are linked to the event as pre-state and post-state, respectively:

```
(createm 'connect1 'event)
(createm 'location-state2 'loc-state)
(createm 'location-state3 'loc-state)
(tellm (has-locst-locatum location-state2 Ring@1778fcd))
(tellm (has-locst-location location-state2 'somewhere))
(tellm (has-locst-locatum location-state3 Ring@1778fcd))
(tellm (has-locst-location location-state3 HexaScrew@11d1c62))
(tellm (has-locst-localizer location-state3 'onto))
(tellm (has-ev-activity connect1 indefact2))
(tellm (has-ev-pre-state connect1 location-state2))
(tellm (has-ev-post-state connect1 location-state3))
```

This process continues until the axle is complete, which the LOOM classifier notices automatically. Here is the definition of the concept:

```
(defconcept axle
    :is (:and cube-with-parts
              (:exactly 2 has-connectee-part)
              (:all has-connectee-part axle-part)
              (:satisfies (?y) (Sum (has-connectee-pos ?Y) 7))))
```

'Axle-part' is in the same way defined as a screw with a ring and wheel connected to it. The 'satisfies' clause in the concept ensures that the two axle-parts are indeed mounted to opposite sides of the cube (otherwise, all the parts would be there and connected, but not to the effect of a functioning axle). The resulting object in Java-3D is shown in figure 3.

The text planning module, in charge of building a rhetorical graph structure, consults the knowledge base to determine that the concept 'axle' is a sub-concept of 'integral-part', which denotes an integral constituent of some higher-level entity (here, some vehicle). Accordingly, it infers that constructing the axle was indeed the purpose of the action sequence, and thus constructs a structure that can be abbreviated as follows:

```
(PURPOSE (has-nucleus (construct-axle ...))
         (has-satellite (SEQUENCE (take screw ...)
                                  (take ring ...)
                                  (put ring screw ...))))
```

Using only one aggregation rule (skip the second 'take' action), a straightforward English version of the text produced by our system is

> "Take a hexagon bolt, and put a ring onto the hexagon bolt, and put a wheel onto the hexagon bolt, and fasten the hexagon bolt to a thread-cube, and take a hexagon bolt, and put a ring onto the hexagon bolt, and put a wheel onto the hexagon bolt, and fasten the hexagon bolt to the thread-cube, in order to construct an axle."

In our ongoing work on the sentence planner, this text is to be improved by various additional aggregation rules. In contrast to "chunking" the elementary events from the input data stream, we are now concerned with aggregation on the *text* level: Sentence boundaries have to be introduced, which requires a different signal of the PURPOSE relation (some appropriate adverbial rather than a conjunction), and referring expressions can be improved. Furthermore, the fact that the two halfs of the axle are assembled in exactly the same way should be reflected in the text (which is to be recognized on the level of action aggregation rather than text aggregation, though). One possible target text incorporating these improvements is:

> "In order to construct an axle, take a hexagon bolt, put first a ring and then a wheel onto the hexagon bolt, and fasten this bolt to a thread-cube. Take another hexagon bolt and again put a ring and a wheel onto it. After that, fasten this bolt to the thread on the opposite side of the thread-cube."

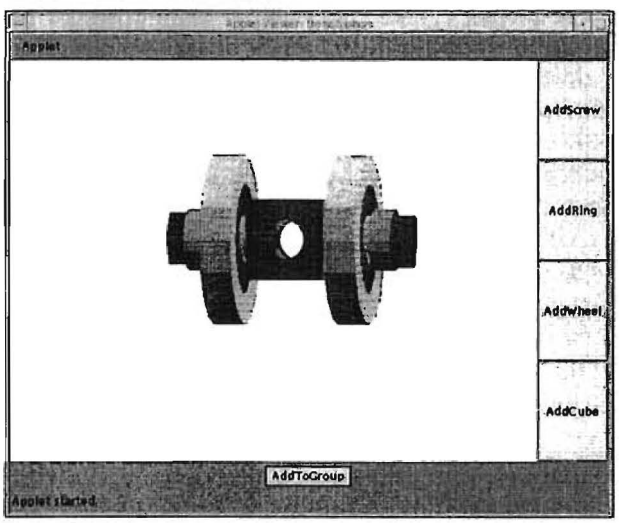

Fig. 3. Screenshot Java-3D: assembled axle

4 Perspectives

4.1 Directions for extensions

The pilot application is merely a first "proof of concept" for the scenario, which can now be enhanced into various directions. As mentioned above, an improved sentence planning module is currently under development. Another step that

needs to be improved for a larger-scale application is constructing the input to the text planner: Recall that at present, we use simple annotations to KB concepts in order to determine whether the creation of some object was made "on purpose" and is to be verbalized as such. In general, though, it is necessary to map the (partly aggregated) stream of elementary actions first to a (pre-verbal and pre-RST) plan structure that explicitly reflects what actions are parts of other actions, and what goals are being followed [Mellish, Evans 1989]. This structure can then be mapped to an RST-inspired text plan, as it was for instance done in TECHDOC [Rösner, Stede 1994].

Improvements can also be envisioned on the side of the graphics input. For example, instruments and tools that are necessary for certain actions can appear as clickable icons, so the user can indicate that it is required for some activity ("remove the wheel with a screwdriver"). Furthermore, the graphical world and the knowledge base should be coupled more closely to the effect that graphics activities by the user are immediately checked by the KB for their possible consequences. For example, some rules of physics can be implemented so that moving a liquid into a container has a different effect than moving it to the outside of a container; this kind of knowledge does not belong to the Java-3D model but to the symbolic knowledge. Also, visible consequences of user's actions (e.g., a light turning on in response to moving a switch) need to be computed in the KB and propagated back to the visual scene.

Java-3D offers the advantage that the applications can be run over the web. The associated disadvantage, however, is that the 3D models as well as the possible modes of interaction are relatively impoverished when compared to state-of-the-art virtual reality environments. On the other hand, 3D web browsers, coupled with more sophisticated input devices (3D mouse, data glove), will soon become available and commonplace. Then, the task of creating verbal protocols of user's activities in the virtual reality will of course be much more complex than in our example presented above, but it offers many applications, not only for producing instructions but also for other purposes.

4.2 The role of the knowledge base

As we have stressed the role of automatic classification in the process leading to verbalizing the user's activities, it is clear that a comprehensive domain model must provide the detailed concept representations enabling these classifications. For our pilot implementation, the domain model was built by hand, but it reused significant portions of the ontology and domain model that were developed earlier for the MOOSE system. Re-usability is indeed a key factor for scaling up the prototype to a practical application: When models are built in such a way that the upper ontology as well as general knowledge about types of technical objects can be carried across domains, the prohibitive costs of manually building domain models can be reduced. Furthermore, it can be expected that the ongoing efforts in standardizing knowledge representation formats and in sharing knowledge bases will lead to the availability of standard modules that can be used as a *basis for the knowledge sources required for generation.*

4.3 Toward an author's workbench

Focusing now again on instructional text, we notice that extending the graphical environment into a full-fledged virtual reality will, at any rate, cover only one side of the coin. While descriptions of sequences of physical activities are a central ingredient of instructional text, there are additional elements that also need to be accounted for, and that are not easily accomplished with graphical means.

Consider a somewhat more complicated instructional text from a car manual. We have divided it into 'minimal units' and marked them with square brackets.

> [Wait]1 until [the engine is cool]2, then [turn the radiator cap clockwise]3 until [it stops]4. [DO NOT PRESS DOWN WHILE TURNING THE CAP]5. After [any remaining pressure has been relieved]6, [remove the cap]7 by [pressing down]8 and [again turning it counterclockwise]9. [Add enough coolant]10 to [fill the radiator]11, and [reinstall the cap]12. [Be sure to tighten it securely]13. [Fill the reserve tank up to the max mark]14 with [the engine cold]15.

Using the labels of the minimal units, the text can be assigned the following RST analysis (notation: (RELATION NUCLEUS SATELLITE)):

```
(SEQUENCE (UNTIL 1 2)
          (CIRCUMSTANCE (UNTIL 3 4)
                        5)
          (PRECONDITION (PURPOSE (SEQUENCE 8 9)
                                 7)
                  6)
          (PURPOSE 10 11)
          12
          13
          (PRECONDITION 14 15))
```

Several portions of this text cannot be inferred from actions in the graphics environment. First, the UNTIL–relation cannot be read off directly from an action sequence; and in particular, 'waiting' is not an action that is easily demonstrated in a virtual world. Second, both PRECONDITIONs are problematic: noticing that all pressure has been relieved would require a highly sophisticated simulation; the engine being cold might be visualized in some way or another, but the fact that it is a precondition for something else might not. Third, CIRCUMSTANCES typically convey information that *accompanies* an activity and is thus difficult to simulate; here it is even more problematic since it is an instruction *not* to do something. Finally, the "be sure..." sentence represents a cognitive activity rather than a physical one.

To account for such problems, additional mechanisms are needed. While it is not impossible that the user in between actions clicks on some buttons with coherence relations on them, this would only be a partial answer. In general, it seems more reasonable to open up the possibility of linguistic interaction in addition to graphics interaction. Here, fusing our approach with the WYSIWYM method proposed by Power and Scott [1998] seems to be a viable option. The

graphics component would produce the "backbone text" that the user can further augment by clicking on the text rather than on the image. For instance, text segments spanned by a coherence relation can be marked and the relation chosen from a menu, whereupon the system would alter the text to include a signal for the relation. Similarly, new propositions can be inserted, such as cognitive activities or circumstances of actions.

Assuming that our approach is developed into the directions just sketched, it can address a significant problem in the production of technical documentation: the knowledge gap between engineer and technical writer. Nowadays, the technical writer typically receives a more or less precise specification from the engineer and strives to produce a readily understandable text from it. This can require feedback from the engineer, which is not always available, though. The resulting instruction manuals often reflect this problem. When the engineer can through virtual-reality interaction provide a detailled formal representation of the instruction content, the gap may be narrowed: The generation system turns the specification into a clear and unambiguous — yet raw — text, and the technical writer can interactively polish it to the effect that a well-written and understandable text results.

References

[André 1997] E. André. "WIP and PPP: A Comparison of two Multimedia Presentation Systems in Terms of the Standard Reference Model." *Computer Standards and Interfaces* 18(6-7), 1997.

[André et al. 1988] E. André, G. Herzog, T. Rist. "On the Simultaneous Interpretation of Real World Image Sequences and their Natural Language Description: The System SOCCER." In: *Proceedings of the 8th ECAI*, München, 1988.

[Bateman 1997] J. Bateman. "Enabling Technology for Multilingual Natural Language Generation: The KPML Development Environment." In *Journal of Natural Language Engineering*, 3(1), 15-55, 1997.

[Dalianis 1996] H. Dalianis. *Concise Natural Language Generation from Formal Specifications.* Dissertation, The Royal Institute of Technology and Stockholm University, Stockholm, 1996.

[Hartley, Paris 1997] A. Hartley, C. Paris. "Multilingual Document Production: From Support for Translating to Support for Authoring." In: *Machine Translation* 12, pp. 109–128. 1997.

[Hartmann et al. 1998] K. Hartmann, R. Helbing, D. Rösner, T. Strothotte. "Visdok: Ein Ansatz zur interaktiven Nutzung von technischer Dokumentation." In P. Lorenz and B. Preim, editors, *Simulation und Visualisierung '98*, SCS-Society for Computer Simulation Int., pp. 308-321, Erlangen, 1998.

[Hemsen 2000] H. Hemsen. "Generierung mehrsprachiger Instruktionstexte aus der Interaktion mit einer virtuellen Welt." Diploma thesis, Dept. of Computer Science, TU Berlin, April 2000.

[MacGregor 1991] R. MacGregor. "Using a Description Classifier to Enhance Deductive Inference." In: Proc. of the Seventh IEEE Conference on AI Applications, 1991.

[Mann, Thompson 1988] W. Mann, S. Thompson. "Rhetorical structure theory: Towards a functional theory of text organization." In: *TEXT*, 8:243-281, 1988

[Mellish, Evens 1989] C. Mellish, R. Evans. "Natural language generation from plans." In: *Computational Linguistics* 15(4), pp. 233–249, 1989.

[Novak 1987] H.-J. Novak. *Textgenerierung aus visuellen Daten: Beschreibung von Straßenszenen.* Heidelberg: Springer Verlag, 1987.

[Power, Scott 1998] R. Power, D. Scott. "Multilingual authoring using feedback texts." In: Proceedings of COLING-ACL '98, pp. 1053–1059, Montreal, 1998.

[Rösner, Stede 1994] D. Rösner, M. Stede. "Generating multilingual documents from a knowledge base: The TECHDOC project." In: *Proceedings of the International Conference on Computational Linguistics (COLING).* Kyoto, 1994.

[Sowizral et al. 1998] H. Sowizral, K. Rushforth, M. Deering. *The Java 3D API Specification.* Amsterdam: Addison-Wesley Longman, 1998.

[Stede 1999] M. Stede. *Lexical semantics and knowledge representation in multilingual text generation.* Dordrecht/Boston: Kluwer, 1999.

[Wachsmuth, Jung 1996] I. Wachsmuth und B. Jung. "Dynamic Conceptualization in a Mechanical-Object Assembly Environment." *Artificial Intelligence Review* 10, pp. 345–368, 1996.

Image Descriptions from Annotated Knowledge Sources

Gloria De Salve, Berardina De Carolis and Fiorella de Rosis

Intelligent Interfaces, Department of Computer Science,

University of Bari, Italy

{desalve,decarolis,derosis}@di.uniba.it

Chiara Andreoli, M.Luisa De Cicco and V Cavallo

Department of Experimental Medicine and Pathology,

University "La Sapienza", Rome, Italy

Abstract. We present the first results of a research aimed at generating image descriptions from annotated knowledge sources. In particular, we discuss the role of annotation in the generation process and the approach we adopted in annotating the data and the discourse plan, by showing examples from an application in the medical setting: the application concerns description of radiological images, either individually or by comparison with 'reference' images, in the context of dynamically generated hypermedia guidelines.

1 Introduction

The amount of information in the WWW is growing exponentially; this growth makes it increasingly difficult to find, access, present and maintain information. From research about how to make these tasks easier, methods for making machine understandable the information available in the WWW have emerged: these methods require associating semantics to information, through domain-specific annotation languages. An annotation can be loosely defined as "any object that is associated with another object by some relationship" (from the W3C Annotation Working Group). In particular, XML is a standard, proposed by the W3C, to create mark-up languages for annotating documents in a wide variety of application domains; developing such languages brings the advantage of favouring re-use and share of resources [11].

We are investigating how annotations could be used in NLG and, in particular, in generating explanations from concept ontologies, to examine the advantages this approach might offer and the efforts it requires. Introducing annotations in a NLG system requires two main steps:

1. *defining annotations for knowledge sources* in the application domain and for the intermediate results of the generation process; whenever possible, already exist-

ing and shared annotation languages should be employed (especially as far as application domain data are concerned);
2. *revising the NLG algorithms* so as to enable every generation module to read annotated data and to produce annotated results.

Annotating resources allows, in general, knowledge sharing and makes their semantics machine understandable. In particular, annotating the steps of the generation process (for instance, the discourse plan) enforces a distributed vision of the process and enables rendering the final output as a function of the device through which the User interacts.

In this paper, we illustrate how we applied this approach in a particular context: generating explanations about radiological images in ARIANNA [3], a system that is aimed at dynamically generating user adapted hypermedia presentations of medical guidelines. Our medical partners envisage using this system to instruct students and to spread guidelines among general practitioners and specialists. In addition to the guideline, the prototype is able to dynamically generate user adapted explanations of concepts involved in the clinical decision process: in this context, the User may ask to see some example about the explained concept, to better understand it; this example may be described either individually or by comparison with other cases, that the User is presumed to already know. As we work in the radiological domain, most of the examples are illustrated by images; therefore, our goal is to automatically generate context-dependent image descriptions, and we need "understanding" images to this purpose.

Since we are not interested in automatic image recognition, we build and use metadata by annotating every item in the image database by means of a image annotation tool. To this purpose, we defined a XML-based mark-up language for radiological images and we developed an algorithm for interpreting its semantics. Starting from an annotated image to be described and a given communicative goal that formalises the User request, a *discourse plan* is produced. This plan is built, as usual, by taking into account the User's information needs and her background knowledge, and specifies the information content and the structure of the description text [3,10]: it is written as an XML-structure, according to a mark-up language that we defined for this purpose. The annotated plan is the input of a *Surface Generator* that, according to the interaction context and to the User characteristics, decides how to render it. In the following Sections, we will describe this method in more detail, by focusing, in particular, on how we use the annotation in the NLG process and by discussing the impact that an XML-based annotation may have on this process.

2 Generation of Image Descriptions

The explanation facility of ARIANNA uses two main strategies to generate the concept description that is appropriate in a given context [3]: the concept position in a taxonomy of medical concepts and its relation with "similar" concepts that the User knows. If the User does not know other "similar" concepts, the generated text

provides a complete description of the concept itself, in which its position in the taxonomy is specified by describing the relations with its ancestors. If, on the contrary, the User knows other concepts in the taxonomy (for instance because she has just seen their description), an explanation by comparison with the most similar of them is provided. To select the reference concept, a 'degree of similarity' between concepts is measured, by considering the attributes they have in common; the comparison then includes the 'commonalities' and the 'alignable' and 'non alignable' differences [9]. Only properties appropriate to the User level of knowledge are mentioned in the text: commonalities are presented first, alignable differences second and non-alignable differences at the end. This strategy corresponds to what we consider a systematic description of concepts, which is typical of learning tasks, as opposed to information-seeking ones [7].

As we mentioned in the Introduction, in the context of these explanations, the User may ask to see an example; in the majority of cases, these examples are in the form of radiological images, that have to be illustrated through some natural language text. In our first prototype of ARIANNA, image descriptions were pre-stored comments; this required our radiologists to provide a text for every example image and did not allow us to tailor it to the context. We therefore thought about applying, to produce image descriptions, strategies similar to those we applied in the case of concept explanations, so as to generate automatically texts by also taking into account adaptivity to the User knowledge. However, this goal required that our generator be able to "understand" images: let's see how we did it.

3. Understanding the Image

Understanding a image means extracting the features that characterize the information needed for its description: typically, these features are regions with their shape, texture, edges and so on. Since we do not use automatic image recognition techniques to extract these features, we use metadata to describe the image components, their attributes and the relationships among them. To build these metadata, we use a tool in Java (Inote [8]) that is available on line and provides a way of annotating images with a XML-based mark-up language. Inote allows the User to attach textual annotations to a image and to store them in a text file as XML data. With this tool, our medical partners can mark-up a digital radiological image by directly "writing on it" and without altering it; once a image has been loaded, the borders of one or more regions in the image may be outlined interactively, and a number of attributes may be associated with each region. Regions are called "details" and attributes "annotations", and may be given a name; a text may be associated with every annotation of every detail, by filling a text field. The details may be organized into as many "overlays" as needed. Inote's mark-up language is very general, and may be applied to every kind of image. To tailor it to radiological images, we defined an ad hoc markup language that allows us to identify overlays and details in our images, with their attributes, in a univoque and unambiguously interpretable way. A radiological image has some "General Properties" that identify it: the technique with

which the image was produced, the body region on which the exam was performed and the diagnosis. Its main information content then consists in a list of details that correspond to the regions of interest (anatomic structures); a set of attributes (morphology, density, etc.) is associated with each of them.

```
<overlay>
        <title>parenchymal organs</title>
        <detail>
            <title>liver</title>
            <annotation>
                <title>position</title>
                <text>left</text>
            </annotation>
            <annotation>
                <title>rel_position</title>
                <text>medial-part(abdomen)</text>
            </annotation>
            <annotation>
                <title>morphology</title>
                <text>ellipsoidal</text>
            </annotation>
            <annotation>
                <title>volume</title>
                <text>normal</text>
            </annotation>
            <annotation>
                <title>margins</title>
                <text>regular</text>
            </annotation>
        </detail>
</overlay>
```

Fig. 1. An example of XML structure produced by Inote.

The first overlay in the Inote file then defines the "General Properties" of the image; it is followed by other overlays, representing groups of visible details. For instance, in the CT-scan (Computerised Tomography) of abdominal organs, the following overlays may be defined:

- parenchymal organs
- hollow organs
- vascular structures
- muscular structures
- skeletal structures

The overlay named 'parenchymal organs' includes, as details, the organs in the image that belong to this category: the liver, the spleen and the lung parenchyma.

For each organ or detail, the following attributes may be specified: position in the image, relation with other parts, morphology, volume, density and margins. Each of them corresponds to an annotation. The example in Fig. 1 is a portion of the XML structure that was produced for a CT-scan of the abdomen: one can notice that the attributes 'relation with other parts' and 'density' are omitted in this case, while the attribute 'position' takes the value 'left', ... and so on. Fig 2 shows how this information was introduced, with Inote's graphical interface: the radiologist, after outlining with the mouse the border of a 'detail' (in the example, the 'liver'), enters the annotation of every attribute he/she wants to define for that detail (in the example, the 'morphology').

4 Planning the Image Description

The XML structure produced by Inote represents the knowledge base for our description generator. Before generating a text, our XML-application has to interpret the Inote tags and the detail and the overlay to which every annotation belongs. The

algorithm first establishes the discourse plan that corresponds to the given communicative goal.

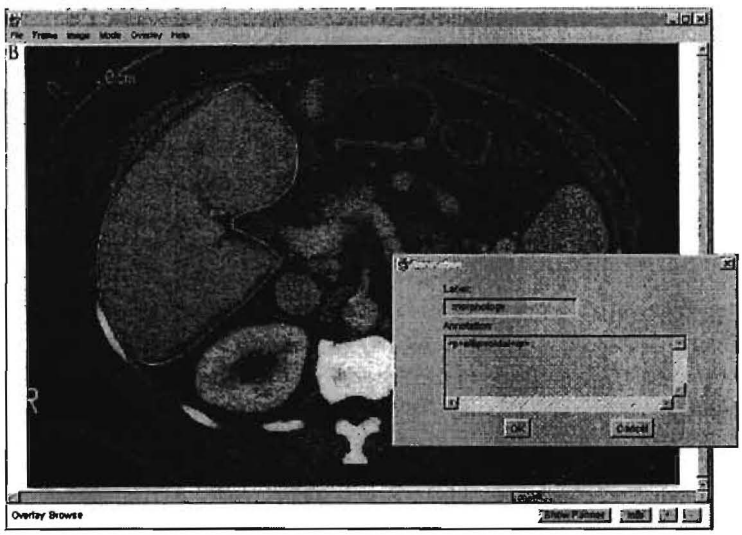

Fig.2: An example of a CT-scan annotation with Inote

According to this goal and to the User characteristics, a presentation plan is selected from a library of non-instantiated plans that are represented as XML structures too; the generic plan is, then, instantiated by filling the slots of its leaves with data in the XML-domain-file that is associated with the image to describe. The DTD definition of our Discourse Plan Markup Language is shown in Fig.3: a discourse plan is identified by its *name*; its main components are the *nodes*, each identified by a name. Mandatory attributes of nodes describe the communicative *goal* and the rhetorical elements: *role* of the node in the RR associated with its father (nucleus or satellite) and *RR name*. The 'info' element, that is not mandatory, may add other information, such as the discourse *focus* of the and the *complexity* of the sub-tree departing from the node. We do not employ this optional information in image descriptions, but we do it in other application domains [4, 5].

```
DPML 1.0 - Discourse Plan Markup Language
<!DOCTYPE d-plan[
<!ATTLIST d-plan name CDATA #REQUIRED>
<!ELEMENT node (node*, info*)>
 <!ATTLIST node name CDATA #REQUIRED goal  CDATA #REQUIRED
            role (root|nucleus|sat) #REQUIRED RR CDATA #IMPLIED>
<!ELEMENT info EMPTY)>
<!ATTLIST info focus  CDATA #REQUIRED compl (H|M|L) #REQUIRED >
]>
```

Fig3. Discourse Plan Markup Language DTD.

The XML-based annotation of the discourse plan is motivated by two reasons: the first one is that, in this way, a library of standard explanation plan may be built, that can be instantiated when needed and can be used by different applications, in several contexts. The second one is that XML is a standard interface between all the components of our Generator, that favours the distribution of resources and increases the computation speed.

A small portion of the XML-Instantiated-Plan that was produced for describing the C.T. scan of the abdomen in Figures 2 and 3 is shown in Fig. 4. In this case, the plan has been instantiated according to the information relative to 'img1.xml' (see the goal 'Explain(image, img1.xml)' associated with n1). Node n9 describes the overlay 'parenchimal organs'; node n10 the detail 'liver' with its attributes, in an appropriate order.

```
<d-plan name="CT-abdomen.xml">
  <node name="n1" goal="Explain(Image, img1.xml)" role="root" RR="Sequence">
    <node name="n2" goal="Describe(General Features, image)" role="nucleus" RR="ElabGenSpec">
      <node name="n4" goal="Inform(diagnosis,normal liver)" role="nucleus" RR="null"/>
      <node name="n5" goal=" Describe(Exam, C.T.)" role="sat" RR="Joint">
          <node name="n6" goal="Inform(name, C.T. Abdomen)" role="nucleus" RR="null"/>
          <node name="n8" goal="Inform(level, spleen)" role="nucleus" RR="null"/>
      </node>
    </node>
    <node name="n3" goal="Describe(Specific Features, image)" role="nucleus" RR="OrdinalSequence">
      <node name="n9"goal="Describe(ComplexStructure-1, parenchymal_organ)" role="nucleus"
      RR="OrdinalSequence">
          <node name="n10" goal="Describe(detail,liver)" role="nucleus" RR="ElabGenSpec">
            <node name="n12" goal="Describe(attribute,liver)" role="sat" RR="Joint">
              <node name="n13" goal="Inform(position,left)" role="nucleus" RR="null"/>
              <node name="n16" goal="Inform(rel_position,medialpart_abdomen)" role="nucleus" RR="null"/>
              <node name="n17"goal="Inform(morphology,ellipsoidal)"role="nucleus" RR="null"/>
              <node name="n18" goal="Inform(volume,normal)" role="nucleus" RR="null"/>
              <node name="n19" goal="Inform(margins,regular)" role="nucleus" RR="null"/>
            </node>
              <node name="n11" goal="Inform(name,liver)" role="nucleus" RR="null"/>
          </node>
          ...
</d-plan>
```

Fig. 4. An example of XML-Instantiated-Plan.

5 Rendering the Image Description

This functionality of our Image Describer is very simple; the XML-Instantiated-Plan is the input of a Surface Realizator that, using flexible templates, produces the image explanation as an HTML file. This process is mainly driven by the RRs between portions of the plan. The plan is explored in a depth-first way; for each node, a linguistic marker is placed between the text spans that derive from its children, according to the RR that links them. For instance, the sentence: "Inside the parenchyma, tubular shaped, hyperdense and white images are visible (the superhepatic veins)", in Fig 5, is obtained from a template for the *ElabGenSpec* RR in which the

satellite (a *Joint* template to the attributes 'position', 'shape', 'density' and 'colour'), is followed by the nucleus stating the name of the object in focus (the superhepatic veins, in this case), that is put in parentheses. The first sentence ('On the left side, there is an organ....') is generated from data shown in figures 1 and 4. We defined the templates' structure after an analysis of a corpus of explanations produced written by the radiologists of the team.

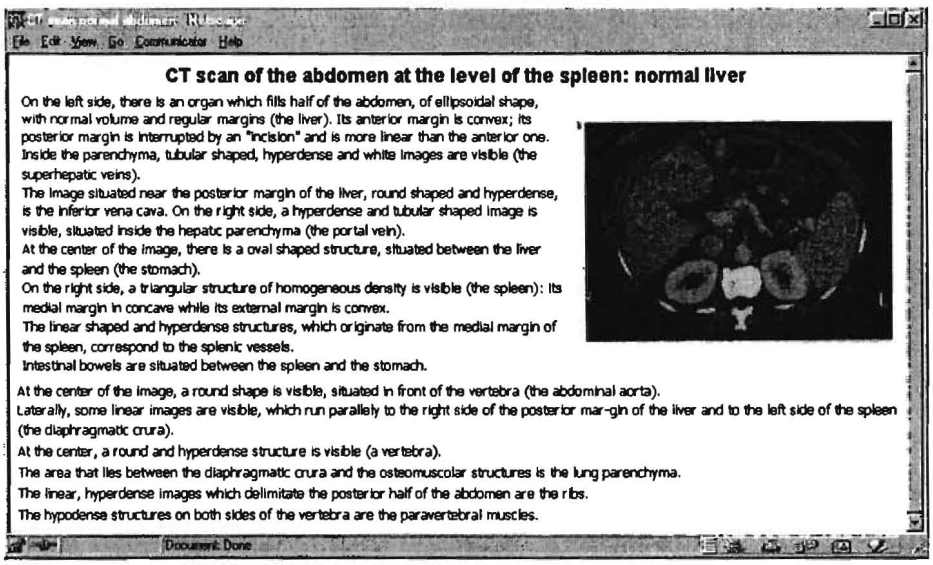

Fig 5. An example of image description.

At present, we generate the text in HTML; however, our approach is general enough to produce descriptions in different formats and, therefore, for different interaction contexts. It is also domain independent, since it is only driven by the communicative goal and the Rhetorical structure of the discourse plan, as in Marcu [12].

6. Comparing Images

Let's now see how we generate the description of a image by comparing it with a reference image. The general strategy we apply is similar to the one we applied to compare concepts in ARIANNA. For every detail in a overlay, we mention first commonalities, second alignable differences and finally non-alignable differences. In the case of image descriptions, we distinguish, at present, three types of comparisons, that depend on what the User already knows and on the images she has already seen. Given a Image I to be described to a User U and a Reference-Image RI, three different comparison plans may be activated:

Comparison 1. KnowAbout(U, RI) AND Remember(U, RI) ⇒ Exec(S, cplan_1);
If the user, according to its background knowledge, profession and level of expertise or according to what she has already seen, knows RI and is presumed to remember its description, the first comparison plan (cplan_1) is applied. This plan corresponds to the following strategy: for each overlay and for each detail, only the attribute values of I that are different from the ones in RI are mentioned (alignable differences). After them, the values of the attributes that are not present in RI are presented (non-alignable differences). This plan is applied, for instance, to describe pathological cases to radiologists.

Comparison 2. KnowAbout(U,RI) AND ¬Remember(U,RI) ⇒ Exec(S, cplan_2);
If the user knows RI but is presumed to not remember it in all its details, the second comparison plan (cplan_2) is applied. This plan corresponds to the following strategy: for each overlay and for each detail, the attributes of I that take different values from those of RI are mentioned, by describing both values (for I and for RI). After them, also in this case, non-alignable differences are presented. This plan is applied, for instance, to general practitioners.

Comparison 3. ¬KnowAbout(U,RI) ⇒ Exec(S, cplan_3);
If the user does not know RI, the third comparison plan (cplan_3) is applied. In this plan, for each overlay and for each detail, all attributes in the two images are described, by emphasizing commonalities, alignable and not-alignable differences. This plan is applied, for instance, to students.

Let us see some examples of comparisons that were generated with our system: in all these examples, the reference image is a CT scan of the abdomen for a 'non-pathological' case, while the image to be described is a case of hepatic cirrhosis, obtained with the same technique. The first text is generated by cplan_3: alignable differences are emphasized in italics, while there are no 'non alignable differences' between the two images; only the first part of the text is shown, for space reasons.

CT scan of the abdomen at the level of the spleen: hepatic cirrhosis.

As in the non-pathological case, the liver is the organ situated on the left side of the image, which fills half of the abdomen, of ellipsoidal shape. *In this case, however, its volume is reduced, its density is inhomogeneous and its margins, instead of being regular, are lobulated.* Like in the normal case, the anterior margin is convex while the posterior one is more linear and is interrupted by an incision. Superhepatic veins are visible inside the parenchyma; they are tubular shaped, hyperdense and white, like in the normal case.

As in the normal vein, the inferior vena cava is situated near the posterior margin of the liver, round shaped and hyperdense. The *portal vein* lies inside the hepatic parenchyma: it is hyperdense and tubular shaped like in the normal case, *but is enlarged.*

As in the normal case, the stomach is visible at the center of the image, between the liver and the spleen, and is oval-shaped.

The spleen is visible on the right side; it is triangular and has a homogeneous density, like in the normal case, *but it is enlarged.* Also in this case, its medial margin is concave while the external one is convex.

The splenic vessels, which originate from the medial margin of the spleen, are linear and hyperdense, like in the normal case.

Also in this image, between the spleen and the stomach, intestinal bowels are visible. .....

If c_plan2 is applied to the same case, the following text is obtained:

CT scan of the abdomen at the level of the spleen: hepatic cirrhosis.

If compared with a non-pathological case, the volume of the liver in this image is reduced, its density is inhomogeneous and its margins, instead of being regular, are lobulated. The portal vein is enlarged and the spleen is enlarged too.

.....

---

7 Conclusions and Future Work

In this paper, we presented the first prototype of Image Descriptor, a software to generate image descriptions from annotated knowledge sources: this prototype was built in Java using the IBM-XML4J parser and will be integrated in a system (ARIANNA) that dynamically generates hypermedia presentations of clinical guidelines; ARIANNA is already in use and an experimental evaluation study has been performed, to check how physicians react to it. The methods and the techniques we employed for generating image descriptions aim at favouring sharing and re-use of information. In particular, annotating images has several advantages: first of all, it enables retrieving images from Web databases according to their information content; in addition, once a image has been retrieved, it may be described in a natural language text whose content, structure, and style may be adapted to the context in which retrieval was made.

The annotation of linguistic resources favours, in general, their re-use and distribution: their semantics can be interpreted and rendered in different ways according to the interaction context; for instance, plain text, HTML or WML. Our research efforts go in this direction: we plan to introduce, in ARIANNA, a Conversational Agent with the role of an "Explainer" that supports the User at different levels; we already developed a similar Agent in another context, the generation of 'Animated User Manuals' for software applications [4]. In passing from hypertexts to Animated Agents, most of the techniques described in this paper will not change: for instance, the DTD for representing discourse plans is the same, and therefore also the planning component remains invaried; we only add a 'Sentence Planner' to revise the XML-plan files, and substitute the Surface Text Generator with a module that generates what we call the "Agent's behaviours" (a mixture of gestures, face expressions and speech).

The research Project with which our Image Descriptor has more relation is Cawsey and colleagues' work on 'resource description' [1,2]. One of the differences with this work is that our Surface Generator uses templates specified as Java classes, that can be instantiated according to the discourse plan portion that they need to render. We did not use an existing standard technique for specifying our templates (like, for instance, Cawsey's et al XSLT stylesheet templates), because they did not allow us to produce, at the same time, complex textual descriptions of individual images or comparisons of couples of images according to context-based criteria

As a final consideration: we claim that establishing standards in the NLG field is a promising approach to enable sharing of resources and methods among various research centers and to produce outputs in context and application-dependent forms. This may foster re-use of methods in different applications and settings: let's think about new UMTS phones or wearable computers, whose particular graphical interface will require revising the generation methods that many of us developed so far. The work described in this paper is a step in this direction.

Acknowledgments

This work was partially founded by the CNR grant 21.15.01 on: "Digital Processing of Radiological Images" and by the National Co-founded Project on "Intelligent Agents: Knowledge Acquisition and Interaction".

References

1. Cawsey, A. Presenting tailoed resource descriptions: Will XSLT do the job? In Proceedings of the 9th International WWW Conference, (2000).
2. Cawsey A., Bental D., Bruce E. and McAndrew, P.: Generating resource descriptions from metadata to support relevance assessments in retrieval. Proceedings of RIAO 2000.
3. De Carolis, B., de Rosis, F., Andreoli, C., Cavallo, V. and De Cicco, M.L.: The dynamic Generation of Hypertext Presentations of Medical Guidelines. The New Review of Hypermedia and Multimedia, 67-88 (1998).
4. De Carolis, B., de Rosis, F., Pizzutilo, S.: Generating User-Adapted Hypermedia from Discourse Plans. Fifth Congress of the Italian Association of Artificial Intelligence (AI*IA 97), Roma , (1997).
5. B. De Carolis, C. Pelachaud, I. Poggi, Verbal and non verbal discourse planning. Workshop on Achieving Human-like Behaviors. Autonomous Agents 2000. ACM Press.
6. de Rosis, F., De Carolis, B., Pizzutilo, S.: Automated Generation of Agent's Behavior from Formal Models of Interaction. To appear in proceedings of AVI 2000, Palermo, Italy (2000).
7. Hammond, N. and Allinson, L.: Extending Hypertext for Learning: an Investigation of Access and Guidance Tools. People and Computers V, HCI 89, Cambridge University Press (1989).
8. Inote: Image Annotation Tool. http://jefferson.village.edu/iath/inote.html.
9. Markman., A.B. and Gentner., D.: Commmonalities and Differences in Similarity Comparisons. Memory and Cognition, 24, 2 (1996).
10. Moore, J., D. Participating in Explanatory Dialogues. Interpreting and Responding to Question in Context. ACL-MIT Press series in NLP, (1995).
11. W3C: eXtensible Markup Language (XML). http://www.w3.org/xml/
12. Marcu, D.: Extending a Formal and Computational Model of Rhetorical Structure Theory with Intentional Structures à la Grosz and Sidner. The 18th International Conference on Computational Linguistics COLING'2000, Luxembourg, (2000).

Natural Language Generation in the MILE System

Paul Piwek, Roger Evans, Lynne Cahill and Neil Tipper

ITRI – University of Brighton
Watts Building, Moulsecoomb, Brighton BN2 4GJ, UK
email: Firstname.Lastname@itri.brighton.ac.uk

Abstract. We describe how Natural Language Generation (NLG) Technology is used in the MILE system. MILE is a web-based system for accessing maritime rules and regulations. We explain how the architecture of the system was derived from a set of user requirements and focus on the role of NLG in this architecture. More specifically, we describe how multilingual generation of answers to queries and the use of the NLG-based WYSIWYM-technology for multilingual query formulation fit into the architecture. The architecture is different from conventional dialogue systems in that it is centered around a dialogue database which allows the user to store, retrieve and manipulate so-called dialogue histories (i.e., language-independent records of the dialogue between the user and the system).

1 Introduction

As part of the CLIME (Computerised Legal Information Management and Explanation) project[1], a natural language interface has been developed at ITRI in the University of Brighton for accessing legal and regulatory information via the internet. In particular, an application (MILE; Maritime Information and Legal Explanation) has been built for accessing maritime regulatory information.

In this paper, we describe the place of Natural Language Generation (NLG) Technology in the architecture of the MILE system. We aim to demonstrate how the NLG technology has been put to use in order to satisfy the user requirements of the application. In particular, our goal is to explain how the user requirements led to a dialogue database-oriented approach which makes maximal use of the NLG technology.

The remainder of this paper consists of three sections. In Section 2, we discuss the user requirements and describe how they have been addressed in the MILE system. Section 3 consists of the current state of the application prototype which has been developed in the first two years of the CLIME project. We also discuss the planned efforts for the last year of the project which are aimed at further

[1] CLIME is funded by the EC Esprit Programme under project number EP 25.414. The partners of the CLIME project are British Maritime Technology Ltd., Bureau Veritas, TXT Ingegneria, the University of Amsterdam and the University of Brighton.

improvements of the system on the basis of feedback from the end-users. Finally, Section 4 contains our conclusions.

2 Satisfying the User Requirements

Before addressing the individual requirements for the application, let us characterize its context of use. The intended end-users of the application are surveyors who inspect sea-going vessels on their sea-worthiness. For this purpose, they use a large body of maritime regulations, which are currently available to them on paper and CD-ROM (with simple text retrieval). The MILE system is intended to change this situation. It allows a surveyor to specify the situation on a ship by means of a natural language text. The system is then able to retrieve the rules which are pertinent to the specified situation[2] and present this information by means of a natural language text which includes relevant images (of ships, ship parts, etc.) and HTML-links. Amongst other things, the links provide access to explanatory texts on how the system arrived at its answers.

For instance, a surveyor might want to know which rules apply to the situation described by the following text:

> "An oiltanker is fitted with three bilgepumps. One of them is out of order and another of them is used for firefighting. What are the rules which apply to this situation?"

The idea is that the user can enter this query and that subsequently the system can retrieve the rules which apply to the situation and present them to the user.

The task of a surveyor and the context in which s/he works give rise to a number of more specific requirements on an application of the sort we just described. Within the CLIME project, we formulated a set of such requirements in cooperation with representatives of the industrial partners of the CLIME project. These partners are two organizations which employ surveyors and other professionals who use maritime regulations: Bureau Veritas (one of the largest classification societies with over 100000 clients distributed over 150 countries) and British Maritime Technology, Ltd. (one of the world's leading maritime and engineering consultancies). We will now list the requirements which were thus gathered and describe how these requirements are addressed by the MILE system.

(1) ***The user should be able to formulate (semantically) relatively complex queries pertaining to the situation of a ship.***

For instance, in the text given above we encounter phenomena such as plurality ("three") and anaphora ("one of them"). Unfortunately, for the purpose of practical applications, natural language understanding is not yet sufficiently

[2] For technical details on the retrieval/legal reasoning functionality of the system see Winkels et al. (1998).

reliable to allow a user to enter such texts freely by means of the keyboard or speech (see, e.g., Dix et al., 1998). Therefore, an alternative approach has been explored which allows the user to construct such queries by directly performing editing operations on the semantic representation underlying the query. The approach is called WYSIWYM, for *What You See Is What You Meant* (Power et al., 1998).

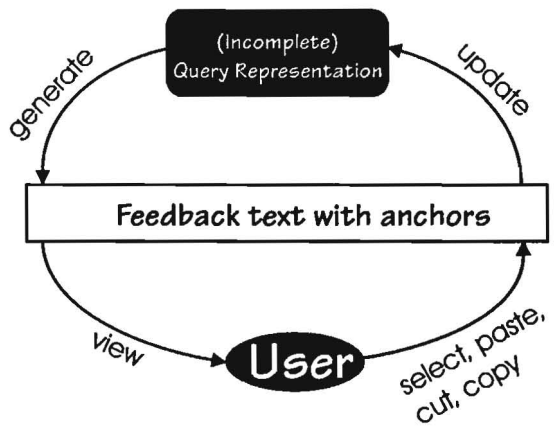

Fig. 1. The editing cycle

The idea, see Figure 1, is simple: a natural language text is generated from a yet to be completed semantic representation of a query. The text contains clickable anchors with pop-up menus. A menu presents the possible extensions of a query representation. On the basis of the extension that the user selects, the representation is updated and a new text is generated on the basis of the updated representation. Additionally, spans of text corresponding to underlying semantic objects can also be selected by means of the mouse. Cut and copy operations are available which allow the user to cut or copy the underlying semantic object into a buffer. Subsequently, such an object can be pasted into a location where the representation is still incomplete.

Consider, for instance a situation in which the following text represents the status of the query: "An oiltanker is fitted with three bilgepumps. **Some equipment** is out of order. **Some states**." Here, bold face indicates where the query is still incomplete. The user can select the span "three bilgepumps", and copy the underlying object (or a subset of it) into the anchor **Some equipment**. Subsequently, the text "An oiltanker is fitted with three bilgepumps. They are out of order. **Some states**." is generated. The underlying representation on which the copy and paste operations take place are object-oriented semantic networks (e.g., Sowa, 1984) which are closely related to Discourse Representation Structures (DRSs; Kamp & Reyle, 1993).

The MILE system uses simple representations which are equivalent to DRSs without logical connectives (such as implication). It does allow for the representation of coreference (Van Deemter & Power, 1998), Plurality (Piwek, to appear) and Speech Act Type information (Piwek et al., 1999).

In summary, the NLG-based WYSIWYM technology has been put to use for the formulation of queries. In this respect, this is a new application of the technology which was originally developed for multilingual document authoring and applied to several domains such as the authoring of software manuals in the DRAFTER II system (e.g., Scott et al., 1998) and more recently the authoring of Patient Information Leaflets in the ICONOCLAST project.[3] ICONOCLAST enables users to formulate logically complex texts. In this respect, there is a difference with MILE. This difference is motivated by the consideration that although the MILE end-users will make frequent use of the technology, the formulation of queries is from their perspective a subsidiary task. On the other hand, for a user of ICONOCLAST the editing of knowledge is the primary task. For such a type of user, the effort of learning how to construct logically complex information is therefore justified. For the average MILE user, this is less evident. We mention this point to draw attention to the tension between the theoretical possibilities of a technology and application specific considerations which can influence which aspects of a technology are made available to end-users.

(2) *The system should be accessible from anywhere in the world.*

This requirement arises out of the working environment of surveyors. Typically, they perform their task by visiting ships, whether it be at a ship yard, in a harbour or at sea. This requirement has given rise to a *web-based multi-agent distributed architecture*, where the interface can be downloaded on the user's computer as a JAVA APPLET which runs in a conventional web browser, whereas the natural language engines (which are written in PROLOG), the Dialogue Manager and the Legal Information Server (written partly in JAVA)[4] can run on high performance (windows NT) machines elsewhere. The latter modules can handle multiple users. In other words, they can handle communications with more than one user interface module.[5] An overview of the different modules and their organization is given in Figure 2.

This figure contains a screen dump of a special module (intended primarily for demonstration purposes), the "behind module", which provides the user with feedback on the system activity. It highlights the modules which are currently activated (in this case the query and response interface –that is, the user interface module– on the left at the bottom).

---

[3] See http://www.itri.brighton.ac.uk/research.html#ICONOCLAST

[4] The legal information server matches the situation which the user specified against the body of regulations

[5] Because the main functionality of the system is run by modules which can communicate with multiple user interfaces and the former modules can share user databases, it becomes also possible for users to share (e.g., look at) each other's query databases.

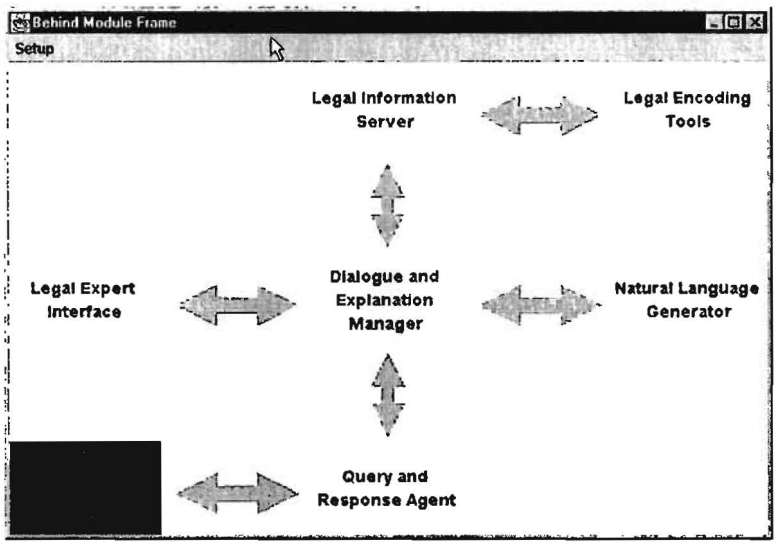

Fig. 2. The system status and architecture as it is conveyed to the user by the behind module

The relation of the Legal Expert Interface (which allows the user to direct a query to a human expert in case the system fails to provide an answer) and the Query and Response Interface (which allows the user to browse a database with queries and answers and construct, submit and manipulate queries) with the Dialogue and Explanation Manager (which manages the database of queries and answers and can generate explanatory information) and the Query Response Agent (which implements the WYSIWYM technology) is a client-server one. Communications on the server side of the architecture (the Dialogue and Explanation Manager, the Query and Response Agent, the Legal Information Server and the Natural Language Generator) are based on the CORBA (Common Object Request Broker Architecture) standard.

The Legal Encoding Tools module has a somewhat different role in the architecture. It is not part of the end-user system. There is a second version of the system for developers which includes this module. The module has been developed at the University of Amsterdam and provides an environment for encoding domain knowledge. Furthermore, in the future some tools which have been developed at the University of Brighton for semi-automatically generating linguistic resources from the domain knowledge will be integrated into this module

(3) *When the system is computing an answer to the user's query, the user should be able to direct his or her attention to other tasks (including the formulation of further queries) and be able to modify and resubmit queries which were posed earlier.*

These considerations have led to a *database-oriented dialogue model* analogous to conventional email systems. Such an architecture allows for asynchronous communication between the user and the system, e.g., the user can formulate and submit new queries before s/he has received the answers to previous queries. A simplified representation of the system architecture is depicted in Figure 3., where the arrows 1. and 6. involve the NLG technology.

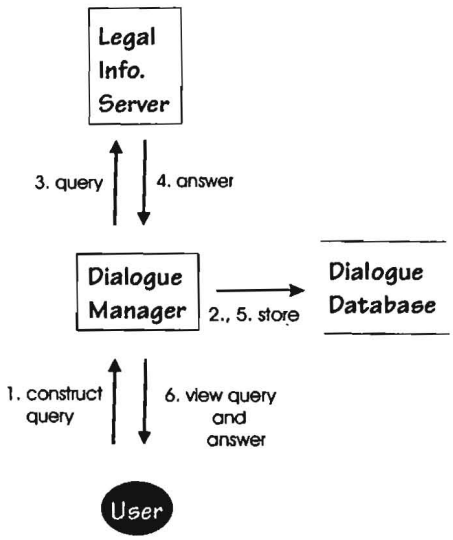

Fig. 3. The system architecture

The idea is that 1. the user constructs a query using WYSIWYM. 2. This query is stored in the Dialogue Database. More specifically, both the natural language text (in fact, several texts: one for each of languages which the system supports) and the formal representation of the query are stored in different fields of one and the same query record. This record carries a unique identifier. 3. The query (representation) is submitted to the Legal Information Server. 4. The Legal Information Server returns an answer in the form of a set of rules and a set of concepts which are pertinent to the users query and a set of properties of and relations between rules and concepts. 5. This information is stored in the Dialogue Database together with natural language texts for the answer which are produced by the NLG on the basis of the answer representation of the Legal Information Server. 6. The user is notified that the Dialogue Database has been updated and can now view the text of the question and its answer (in the language which is appropriate for her or him).

Let us now discuss the processing of a query from the user's perspective. After the user has logged in, a window with two frames appears, containing the browser interface to the MILE system, see Figure 4. The frame on the lefthand

contains the applet which controls the interface. It includes a choice panel which displays a list of queries (and, if available, their answers) which the user has constructed on previous occassions. The user can (re)name these queries, and if required organize them in folders. On the righthand side, there is a view panel which displays the text of the query/answer which the user has selected in the choice panel.

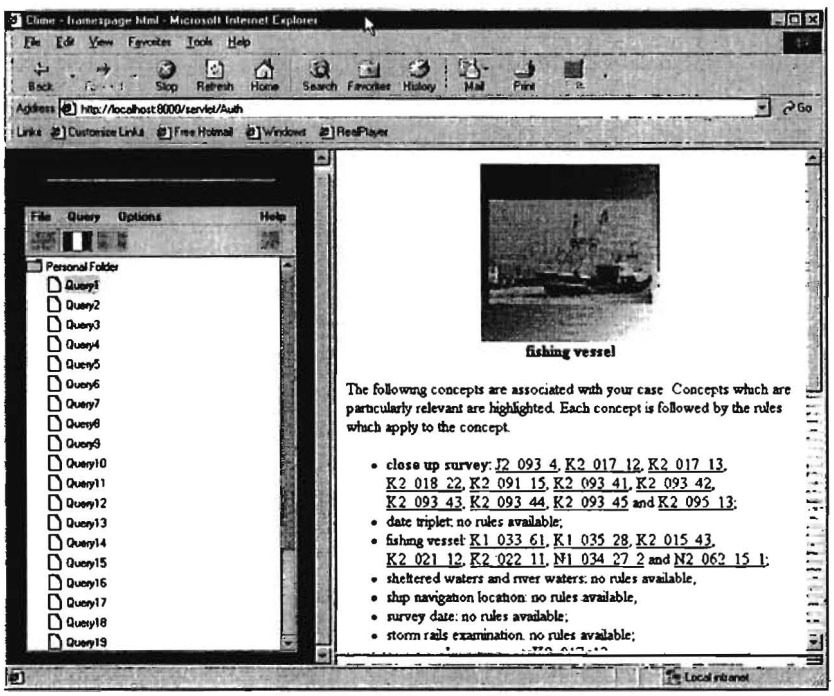

Fig. 4. The Main User Interface

In order to construct a new query, the user selects "Query" and then the option "new". This causes a query-editing window to pop up. In this window, the user can then formulate his or her query using the WYSIWYM technology. See Figure 5. for a query window with a WYSIWYM-constructed query.[6] Alternatively, the user can also access old queries which are stored in the dialogue database, alter them, and then resubmit them.

[6] For a walk through of the WYSIWYM construction process see, for instance, Scott et al. (1998) and Piwek et al. (1999). The former concerns the construction of software manuals, whereas the latter describes the process of query construction in the domain of MILE.

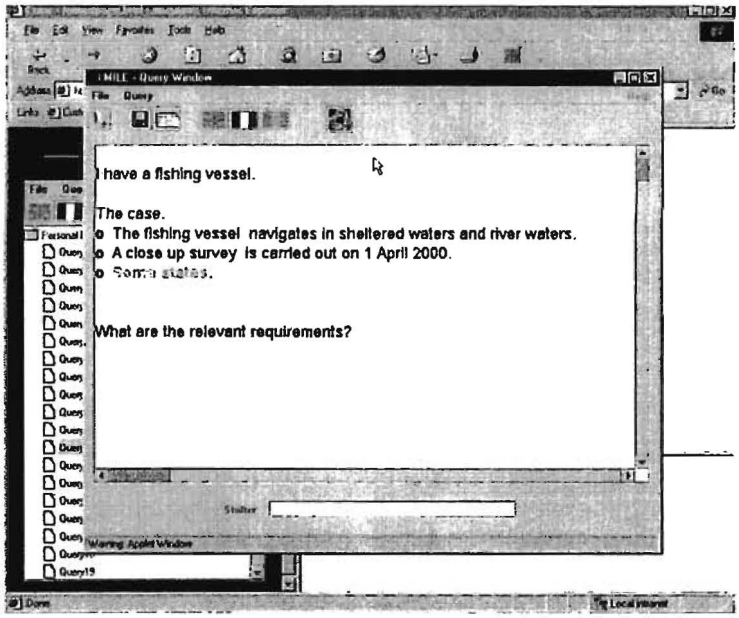

Fig. 5. Query Construction with English Feedback

(4) *The system is used in a world-wide operating company, which means that it should be adaptable to the language of the local users.*

Currently, MILE supports English and French (the lexica cover 300 domain specific concepts).[7] See Figure 6. for the French text of the query which is also depicted in Figure 5. The system uses separate generators (using a pipe-line architecture; cf. Reiter & Dale, 1997) for query formulation and answer generation, although these generators do share the lexical resources. The query formulation generator is based on a unification grammar which allows for the mixing of proper grammar rules and rules for fixed phrases. The input for the generator is the semantic network which the user constructs using the WYSIWYM technology. For the (also multilingual) answer generation, a less complex generator is used.[8] This generator is tailored to quick generation of HTML documents on the basis of the output of the Legal Information Server. A data format has been developed which is particularly suited for generation in legal domains, where the answer consists of a set of rules marked up with explanatory and background information. Bacically, this format is specified as a set of sets: a set of rules, a set of concepts and a set of properties of/relations between rules and concepts.

[7] An Italian Grammar is under construction.

[8] The choice for this generator was driven by practical considerations. Specifically, the general purpose generator which we use is less suited for incorporation of open ended concept classes, such as the set of rule names.

Rules show up in the answer text as HTML links. Clicking on these rules evokes a helper servlet which retrieves the text of the rule and depicts it in a separate window.

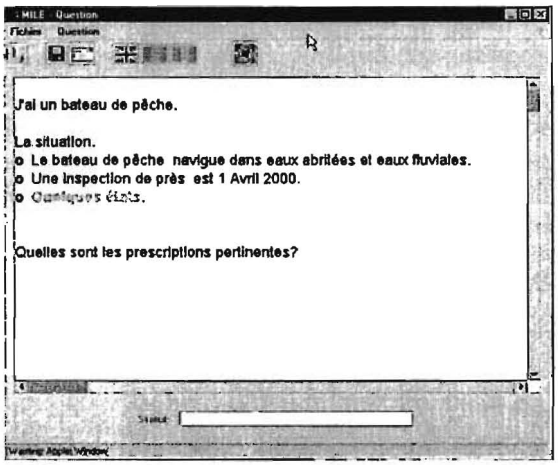

Fig. 6. Query Construction with French Feedback

3 Current State of the Project and Future Developments

Currently (April 2000) the project has just entered its third and last year. Version 3 of the system was demonstrated in November 1999 at the EC Information Society Technologies (IST) Exhibition in Helsinki. Subsequently, the system was distributed amongst the CLIME partners. Feedback from the end-user partners has led to a number of changes to the system. For instance, the layout of the answer texts has been changed. At the same time, we developed a more general representation of the data format which is passed from the Legal Information Server to the NLG. This format has been designed to make generation of different layout as easy as possible. This was achieved by keeping the data structures as "flat" as possible. The idea behind this is that extracting a piece of information from complex recursive structures requires more programming effort and can lead to less robust software.

Furthermore, the development of a demonstrator for a new domain (environmental regulations) has been initiated and should provide us with more information on the reusability of the software.

4 Conclusions

To conclude, let us summarize how we tried to address the two central questions of the workshop in this paper. Firstly, *I have invented a new technique for* NLG*!*

– *What is its impact on applications?* We have indicated how a new technology based on NLG, i.e., WYSIWYM, makes it possible for users to formulate queries of a complexity (e.g., anaphora and plurals) which is not achievable with current technology for free text interpretation. As for the question *I have built a new* NLG *application! – What is its impact on the technology?*, we have tried to show that the demands of the application suggests a new context for the use of NLG, i.e., as part of an application for asynchronous multilingual human-system interaction. Finally, our experiences of building an NLG module which has to interface with other modules (which were developed in parallel) has led us to the adaptation of flat datastructures which allow for less complicated information extraction for NLG.

Acknowledgements Thanks are due to Alessandra Bagnato for kindly letting us use a screen dump of the behind module (Fig. 2) which was implemented by her at TXT Ingegneria. The research reported in this paper was carried out as part of the CLIME project. CLIME is funded by the EC Esprit Programme under project number EP 25.414. The partners of the CLIME project are British Maritime Technology Ltd., Bureau Veritas, TXT Ingegneria, the University of Amsterdam and the University of Brighton.

References

Dix, A., J. Finlay, G. Abowd, R. Beale (1998), *Human-Computer Interaction*, Prentice-Hall, Hertfordshire.

Kamp, H. & U. Reyle (1993), 'From Discourse to Logic', Kluwer Academic Publishers, Dordrecht.

Piwek, P., R. Evans and R. Power (1999), 'Editing Speech Acts: A Practical Approach to Human-Machine Dialogues', *Proceedings of AMSTELOGUE '99: Workshop on the Semantics and Pragmatics of Dialogue*. University of Amsterdam.

Piwek, P. (to appear), 'A Formal Semantics for Generating and Editing Plurals', *Proceedings of COLING 2000*, Saarbruecken.

Power, R., D. Scott and R. Evans (1998), 'What You See Is What You Meant: direct knowledge editing with natural language feedback', *Proceedings of ECAI-98*, Brighton, UK, 1998, 180–197.

Reiter, E. & R. Dale (1997), 'Building Applied Natural Language Generation Systems', Natural Language Engineering 3 (1): 57-87.

Scott, D., R. Power and R. Evans (1998), 'Generation as a Solution to Its Own Problem', Proceedings of the 9th International Workshop on Natural Language Generation, INLG'98, Niagara-on-the-Lake, Canada, August 1998.

Sowa, J. (1984) Conceptual Structures, Addison Wesly, Reading, Massachusetts.

Van Deemter, K. and R. Power (1998), 'Coreference in knowledge editing', *Proceedings of the COLING-ACL workshop on Computational Treatment of Nominals*, Montreal, Canada, 1998, 56–60.

Winkels, R., A. Boer, J. Breuker & D. Bosscher (1998), 'Assessment Based Legal Information Serving and Co-operative Dialogue in CLIME'*Proceedings of JURIX-98*, GNI, Nijmegen, Netherlands 131–146.

Understanding Shortcuts in NLG Systems

Chris Mellish

Division of Informatics, University of Edinburgh, Scotland.
c.mellish@ed.ac.uk,
WWW home page: http://www.dai.ed.ac.uk/daidb/people/homes/chrism/

Abstract. Research on NLG, whether driven primarily by the desire to produce better theories, technologies or applications, is subject to pressures which force compromises to be made. I argue in particular that there are a number of practical reasons why the products of this research make use of "shortcuts", where necessary levels of representation are effectively bypassed. Such shortcuts may be well-motivated but are not always acknowledged, which leads to a misleading impression being given about the state of the art. I give examples of places where it is easy to take shortcuts in NLG research and suggest possible ways in which we could begin to understand and reason about them.

1 Introduction

Research on NLG is driven by a variety of goals, but life is short and the overall problem is so immense that everyone is forced to simplify things in some way. In *theoretical* work, one is faced with the pressure to publish insightful glimpses of a very complex phenomenon, which necessarily forces one to limit the range of issues considered and assume that other problems can somehow be solved. Work on generic *technologies*, on the other hand, is subject to the additional goals of proving correctness or efficiency, perhaps backed up by convincing empirical results. Reusable technologies will be modular, and this forces one to draw boundaries around processes which theoretically are ill-understood. Finally, in work on *applications*, there is pressure for simplicity in order that a system will be maintainable, that its results will be understood and trusted and that its input can be reliably obtained [16]. Existing applications of NLG are niche applications where the restricted domain cuts down the problems that need to be considered. Applications can help the development of techologies by refusing to allow people to ignore reality, but applications that are tightly focussed can allow simplifications that don't transport.

Reusability and the effective transfer of results from theory through technology to applications, requires us to have a rigorous understanding of what we can achieve. And yet, particularly in technology and application development, people use shortcuts of various kinds in order to achieve their goals in a timely way. The danger is not that the shortcuts are there but that their nature is not acknowledged and therefore our understanding of what we can achieve is impaired.

2 Shortcuts

Natural Language Generation is a complex task that involves reasoning at many different levels, e.g. intentions, rhetoric, semantics and syntax. Whereas the inverse problem, Natural Language Understanding, suffers from the problem of *ambiguity*, NLG suffers from the problem of *choice*: the mappings between these levels are not one-to-one and one is forced to make decisions about how to carry them out (decisions which affect the quality of the result in complex and non-local ways). Arguably the long-term solution is to have focussed research analyse the factors behind making particular linguistic choices (e.g. when is a cleft sentence preferred to a non-cleft[4]? when is *don't* used rather than *never* in a preventative expression [20]?). In the short term, however, the pressure to build working systems creates a problem: we know that these different levels are all needed for NLG (and we have some good ideas about parts of the mappings between them), but we can't produce a complete story for the mapping between level B and level C. So we hard-wire some of the connections between the two levels. Once we have done this, it doesn't really matter which of levels B and C we manipulate in order to achieve a desired effect. The danger then is that we manipulate level B implicitly motivated by the desire to achieve affects at level C. The result is that our systems shed no light at all on level B, even though on the surface they appear to. Mapping from an earlier level A seems to produce items at level B but implicitly produces items at level C. This is what I mean by a *shortcut* – a pragmatic decision to skip a level of representation (here, level B) and proceed directly to a "subsequent" level (level C). Figure 1 illustrates the general pattern. The representations at levels B and C are isomorphic and

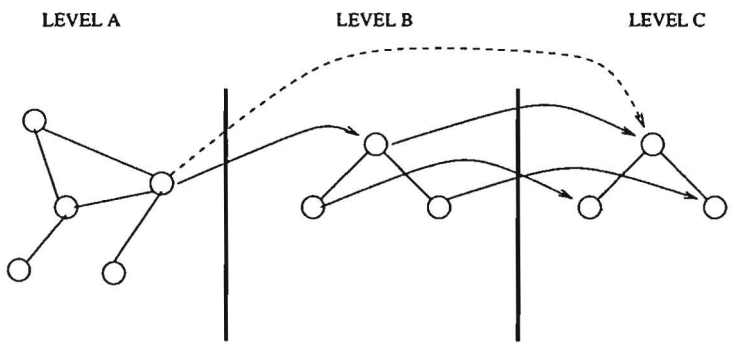

Fig. 1. Shortcutting Level B

so the mapping from level A to level B is "really" a mapping straight to level C (indicated with a dashed line).

Taking a shortcut, one can lose the ability to carry out operations reliably that really require a proper representation at the skipped level (e.g. the abil-

ity to carry out aggregation and syntactically correct concatenation on canned text). Shortcuts also negatively influence reusability, the availability of realistic input, language independence and extensibility (of lexicon, grammar, semantic representation, etc.). Such sacrifices may well be appropriate in many situations, as long as they are understood.

"Canning" material at level C is a way of explicitly acknowledging taking a shortcut, avoiding the previous level B. In a system using canning, for some parts of the task no pretence is made to ever have a representation at level B. Instead, by some means, representations at level C are created directly. Using canned text and templates in a controlled way has become increasingly accepted as part of NLG technology [15, 1]. But canning is applicable at other levels too. For instance, the Text Source Language input for HealthDoc [21] is close to being a form of canned SPL, with the various processes needed to construct this being shortcut. Hyper Template Planning Language [14] contains a proposal to allow canned syntactic structures of various kinds. Some of the referring expressions generated in the Caption Generation System [11] make use of essentially canned semantic representations (which can be contextually optimised to some extent).

A shortcut that is implemented by explicit canning is visible and cannot be misinterpreted. However, often we researchers in NLG do not acknowledge or realise the shortcuts we are making.

3 Unreported Shortcuts

In this section, I outline some of the places where I have noticed shortcuts being taken (implicitly) in my own and in others' NLG systems. The intention is not to be particularly rigorous, but to indicate how easily it can happen.

3.1 Intentional vs Rhetorical

NLG might be said to start with intentions, but it is extremely challenging to motivate the structure of a whole text in terms of intended effects and their prerequisites. For instance, how many RST EVIDENCE relations will be needed to convince a reader of some fact? If I am trying to impress someone about my new bicycle, should I include information about its colour (and, if so, where in my text)?

Whereas it may be hard to define exactly why and how I would want to mention my bicycle's colour in terms of my intentions, when it comes to writing down the rhetorical structure of my text, the ground seems firmer. Theories like RST make quite precise claims about the nature of rhetorical structure and how one might infer/choose it. Thus many NLG systems that generate multisentential text use a rhetorical level of representation at some point. Unfortunately, if the nature of possible intentions is not independently constrained, there can be an implicit shortcut to the more concrete rhetorical level. Thus if one plans entirely in terms of intentions such as "to use the RST EVIDENCE relation" (or something that makes impressive use of mutual belief but amounts to the same) then the

resulting structure of goals and subgoals will look just like an RST tree. And then one will have shed no light on the intentional level.

Moore and Pollack (e.g. [12]) argue convincingly that there can be no one-one mapping from intentions to rhetorical structure. But in some domains, rhetorical structures are relatively stereotyped (or can be inferred easily for desired examples) and so it is clear what this level should look like. In that case there can be a lot of pressure to shortcut a true intentional level.

3.2 Conceptual vs Semantic

For this section, I assume that "conceptual" representations are to do with some input that an NLG system has no control (and limited knowledge) of, whereas semantic representations are specifically designed for representing the meanings of natural language utterances. So the question at stake here is where language orientation first begins in an NLG system.

Shortcutting of the conceptual level is likely to be especially common in the development and testing of NLG technologies. Here one may have the illusion of freedom to design the input to be accepted by the system, whereas in actual applications work one is much more likely to be presented with an input format that is already constrained by external factors. In a real application (for instance, generating from an independently developed reasoning system, from numerical tables or graphical material) there may be a real gap to cross between the conceptual and the semantic ([19] and [8] pointed this out for early systems). But (necessarily?) we lack a well-articulated general way of handling this. Only in some fortunate cases is it possible to influence the conceptual level so that it is in tune with what the NLG system will need [6].

Shortcutting the conceptual level introduces the danger of no longer making realistic assumptions about the input. For instance, by assuming an input that makes available directly the semantic analogue of gradeable adjectives like *fast* and *significant*, one fails to address the question of how the effective use of these depends on context and domain. As a result of making such a simplification, one might end up with a technology that can't actually be applied to *any* application. Semantic representations are also often language-dependent, so it is dangerous to *start* with semantics.

3.3 Semantic vs Syntactic/Lexical

Although almost every NLG system uses a level of semantic representation, usually the nature of this is constrained only indirectly by the nature of the input and the nature of the output of the system. Unfortunately, that allows us to use or abuse this level without the difference being immediately noticed. In particular this level can be effectively shortcut and be simply echoing syntactic or lexical decisions.

The mapping between semantic and syntactic levels is often relatively simple in NLG systems. There may be a near one-one mapping between semantic and syntactic roles; for example (apart from the active/passive issue), the roles of

agent and *patient* may be little more than labels for *subject* and *direct object.* Also the assumption that semantic class predicts syntactic category/behaviour may be exploited in many places. Implicitly anticipating the syntax allows sentence planning to operate on "semantic" representations but really to be making syntactic decisions. For instance, one might make decisions about how much material can be expressed within a noun phrase or sentence at the semantic level, but these are decisions which at the very least are highly constrained by syntax (and will be exposed in a multilingual system by the classic "head switching" examples from MT). Meteer's Text Structure [10] and its later versions are an excellent attempt to provide a level of "abstract syntactic" representation at which this kind of reasoning can be expressed efficiently and reliably. This is not a strictly semantic level of representation.

Another way in which semantic representations can be covert representations of syntax is if the semantic predicates correspond nearly one-to-one with particular lexical items (where by this I mean items whose syntactic category is uniquely determined). Lexical choice is a well-known complex problem, and nobody really knows when it should happen in an NLG system [18]; it is not surprising that many researchers would like to factor out the extra complexities that it introduces. But a semantic notation where predicates map simply onto words is really just a "wishful notation" of the kind discussed in McDermott's criticism of sloppy practices in AI research [7].

3.4 Rhetorical vs Lexical/Syntactic

Although surface cues may provide us with a good way to discover the rhetorical relations used in real texts, that does not mean that the relations used correspond in a one-to-one manner with such cues [5]. For instance, a text may signal a relation using no explicit cue, or the cue (and its syntactic status) may depend on issues such as the size of the text spans that are related. Unfortunately, although there are tantalising glimpses into the structure of the space of coherence relations, in many ways the best handle we have on them is through the visible cues. It is therefore not surprising that many practical NLG systems choose to manipulate the cue phrases rather than the relations (e.g. the STOP system [17]). This amounts to a shortcutting of the true rhetorical level of representation. The papers on the STOP system explicitly acknowledge this simplification, but many of us are guilty of evaluating our "RST trees" (e.g. deciding which rhetorical relations, or how much material, should be able to appear at which points in the tree) by implicitly thinking about what some particular hard-wired realisation of them will come out like. The result is that we are locked into a view of the world that ends up with effectively syntactic decisions being taken at a very early stage in the generation process.

When it comes to thinking about aspects of rhetorical structure that are manifested *within* clauses, existing theories of discourse structure begin to let us down. In such circumstances, it is especially tempting to have special-purpose mechanisms that are making syntactic and lexical decisions when rhetorical structure is being considered. For instance, in the ILEX system [13], phrases

like "also" and "as already mentioned" are effectively inserted directly into the syntactic representation (and that is only possible because assumptions about the shape of the syntactic structure are made very early in the system).

4 Understanding Them

I hope that the above sections indicate that shortcuts are an expected phenomenon given the current state of the art in NLG. But hidden shortcuts are dangerous because they obscure the progress or lack of progress that we are making. The key thing is for us to understand, expose and discuss the shortcuts we are making. How can this happen? I claim that there are three main components to this:

1. Having some kind of agreed ontology for representations manipulated by NLG systems. Beyond this, one might hope to establish for each level perhaps in sequence:
   - Guidelines for what can be stated.
   - A constrained notation and principles about what transformations of this are meaning-preserving.
   - A formal semantics for this notation.
   - A particular instantiation of the notation that uses a generally-accepted repertoire of terms, predicates, relations, etc.
2. Having a general way to talk about partial representations in NLG systems.
3. Having a general way to talk about mixed representations in NLG systems.

It is not necessary that an ontology be watertight in all respects, but it should include generally accepted criteria that distinguish the different levels of representation. How much further one can go in fleshing out the levels is really a question about how much underlying agreement there is in the NLG community notwithstanding the apparent great diversity of theoretical positions taken.

We need a way of talking about partial and mixed representations, because a principled account of shortcuts almost always introduces both. If an NLG system sometimes (but not always) shortcuts level B in the process of mapping from A to B to C, it cannot honestly claim as level B representations those parts that are acting simply as proxies for level C representations. The honest way to represent these at level B is as "placeholders" or representations whose content is unspecified. The overall level B representation will then be *partial* in that it will have "holes" where these placeholders appear. And if the result of the mapping from a structure at level A is to be communicated between system modules, that result will consist of a *mixture* of structures at level B and C, and the only way to make sense of this mixture will be via their connections to one another and to the original structure at level A.

The RAGS project (based at the Universities of Edinburgh and Brighton) is an initial attempt to address some of these issues. The work, which is still underway, seeks to define a number of levels of representation used by NLG systems in enough detail that datasets can be exchanged between systems (using XML as

the interchange format) [2]. The different levels of representation are related to a single underlying data model which by its nature allows for the expression of partial and mixed structures [3, 9]. The RAGS work is by no means the ultimate ontology for NLG systems or the last word in how complex evolving representations can be supported for NLG, but it is a useful start. Part of the RAGS work involved reimplementing the Caption Generation System [11] in a way that followed the RAGS ontology explicitly. Although (or maybe because) the CGS was a simple pipeline system, the reimplementation needed to make extensive use of mixed representations in order to capture what the system actually did. It is highly likely that reinterpretations of existing NLG systems in terms of a generally accepted ontology will involve similar use of mixed representations.

A key feature of the RAGS approach to data representation is to allow shortcuts, but to make it obvious that they are there. Shortcutting of levels and opaque modules whose internal processing is better not examined in too much detail can be accommodated, and that is just as well, given the current state of the art. Such techniques may not lead to maximum reusability, but the RAGS approach does give some basis at least for describing clearly what is actually going on.

NLG researchers need to be able to represent explicitly what is happening in their systems and admit to it. People have started admitting to using canned text and templates. There is a wider range of shortcuts that we're all responsible for. It's time that we admitted it and tried to understand what we are really doing.

5 Acknowledgements

The RAGS project ("Reference Architecture for Generation Systems") is funded by the UK EPSRC via grants GR/L77041 (Edinburgh) and GR/L77102 (Brighton).

References

1. Busemann, S. and Horacek, H., "A Flexible Shallow Approach to Text Generation", *Procs of the Ninth International Natural Language Generation Workshop*, Niagara-on-the-Lake, Canada, pp238-247, 1998.
2. Cahill, L., Doran, C., Evans, R., Mellish, C., Paiva, D., Reape, M., Scott, D. and Tipper, N., "Towards a Reference Architecture for Natural Language Generation Systems", Technical Report ITRI-99-14, Information Technology Research Institute, University of Brighton, 1999. Available at `http://www.itri.brighton.ac.uk/projects/rags`.
3. Calder, J., Evans, R., Mellish, C. and Reape, M., "'Free choice' and templates: How to get both at the same time". Proceedings of *"May I Speak Freely?" Between Templates and Free Choice in Natural Language Generation*, KI-99 Workshop, pp 19-24, Bonn, 1999.
4. Delin, J., "Cleft Constructions in Discourse", PhD thesis, University of Edinburgh, 1990.

5. Knott, A., "A Data-Driven Methodology for Motivating a Set of Coherence Relations", PhD thesis, University of Edinburgh, 1996.
6. Levine, J., Rogers, I., Bennington, T. and Pattison, C., "Class Hierarchies as a Multi-Purpose Knowledge Representation in a Requirements Capture and Design Tool", in *Research and Development in Expert Systems XIII*, eds. J. Nealon and J. Hunt, SGES Publications, 1996.
7. McDermott, D., "Artificial Intelligence meets natural stupidity", SIGART Newsletter 57, 1976.
8. Mellish, C. and Evans, R., "Natural Language Generation from Plans", *Computational Linguistics* Vol 15, No 4, 1989, pp233-249.
9. Mellish, C., Evans, R., Cahill, L., Doran, C., Paiva, D., Reape, M., Scott, D. and Tipper, N., "A Representation for Complex and Evolving Data Dependencies in Generation", *Procs of the Applied Natural Language Processing (ANLP-NAACL2000) Conference*, Seattle, May 2000.
10. Meteer, M., "The Generation Gap: The Problem of Expressivity in Text Planning", PhD thesis, University of Massachusetts, 1990.
11. Mittal, V., Moore, J., Carenini, G. and Roth, S., "Describing Complex Charts in Natural Language: A Caption Generation System". *Computational Linguistics*, 24(3), pp431-468, 1998.
12. Moore, J. and Pollack, M., "A Problem for RST: The Need for Multi-Level Discourse Analysis", *Computational Linguistics* Vol 18, No 4, pp 537-544, 1992.
13. Oberlander, J., O'Donnell, M., Knott, A. and Mellish, C., "Conversation in the museum: experiments in dynamic hypermedia with the intelligent labelling explorer." *New Review of Hypermedia and Multimedia*, 4, 11–32, 1998.
14. Pianta, E. and Tovena, L., "Mixing Representation Levels: The Hybrid Approach to Automatic Text Generation", *Proceedings of the AISB'99 Workshop on Reference Architectures and Data Standards for NLP*, Society for the Study of Artificial Intelligence and the Simulation of Behaviour, April 1999.
15. Reiter, E., Mellish, C. and Levine, J., "Automatic Generation of On-Line Documentation in the IDAs Project", *Procs of the Third Conference on Applied Language Processing*, pp64-71, 1992.
16. Reiter, E., Mellish, C. and Levine, J., "Automatic Generation of Technical Documentation", *Applied Artificial Intelligence* Vol 9, No 3, pp259-287, 1995.
17. Reiter, E., "Shallow vs Deep Techniques for Handling Linguistic Constraints and Optimisations". Proceedings of *"May I Speak Freely?" Between Templates and Free Choice in Natural Language Generation*, KI-99 Workshop, Bonn, 1999.
18. Stede, M., *Lexical Semantics and Knowledge Representation in Multilingual Text Generation*, Kluwer, 1999.
19. Swartout, W., "XPLAIN: A System for Creating and Explaining Expert Consulting Programs", *Artificial Intelligence* Vol 21, 1983.
20. Vander Linden, K. and Di Eugenio, B., "Learning Micro-Planning Rules for Preventive Expressions", *Procs of the 8th International Natural Language Generation Workshop*, Herstmonceux, 1996.
21. Wanner, L. and Hovy, E., "The HealthDoc Sentence Planner", *Procs of the 8th International Natural Language Generation Workshop*, Herstmonceux, 1996.

Invited Talk

Parsing to Text Structure: The Basis of a Reversible Natural Language Generation System

David McDonald, Brandeis University

How to get a source that is rich enough conceptually and structurally to generate from comfortably is a problem that has always vexed NLG researchers. Our answer is that you should build it yourself - a conclusion we have drawn after a decade of work on this problem from the perspective of natural language understanding and semantic representation. This is because we believe that the natural origin of the conceptual content for applications in summarization or derived reports is human-authored text. Applying our information extraction system to these texts will automatically render the content into our preferred source representation for generation.

We go a step further than simple IE and use a system that recovers not just domain-level objects but also reconstructs the text structure that would have generated the texts. This puts us in a position (1) to mine sets of text structures with different realizations of the same object types to learn the suites of realization perspectives that human authors use in a given genre; and (2) to record the collective idiosyncrasies that govern the realization of the complex, multi-term relations found in everyday business text.

This talk will describe the architecture of our system. How it uses Tree Adjoining Grammar as the representation of its linguistic resources for both parsing and surface realization. How it associates these resources with the type definitions in the domain model to automatically construct the semantic parsing grammar. How it uses the type definitions schematically to ensure the expressibility of individual objects and known collections of objects. And how it uses a semantic representation of partially saturated relations to simplify the linguistic reasoning need to produce contextually cohesive texts. Examples will be taken from the domain of corporate quarterly earnings.

Burning Issues Session

Corpora, Evaluation and Generation

Srinivas Bangalore, AT&T Research, Florham Park, NJ

The availability of a parse-annotated treebank (e.g. Penn Treebank) and an parse evaluation metric (e.g. Parseval) has led to ever increasing models for stochastic parsing. The availability of corpora has also spurred a methodology for developing large-scale grammars capable of parsing real-world texts (e.g. XTAG, HPSG).

It appears that some aspects of generation can benefit from the availability of an annotated corpus and an evaluation metric. In particular, we will focus on their contribution to sentence planning and surface realization components of a generation system. Some of the questions, we would like to raise for discussion include:

Corpus related:

- What phenomena are suitable for corpus-based analysis?
- What kinds of annotations are needed?
- Can we reuse corpora created for training parsers and word-sense disambiguation models?
- What about a corpus of paraphrases?

Evaluation Metrics related:

- What do we evaluate?
- How do we evaluate?
- Relevance of metrics to human judgements

Consequences of corpus-based techniques for Generation:

- What are corpus-based techniques good at?
- What are their limitations?
- Can we exploit large-scale grammars along with corpora?
- Issues for commercial applications?

Burning Issues Session

Burning Issue for NLG: The Opportunities and Limits of Statistics-Based Generation

Eduard Hovy, USC/ISI, Los Angeles:

Since the early 1970s, many aspects of NLP (speech recognition, IR, word segmentation, part of speech tagging, and recently parsing and MT) have been addressed, some very successfully, by statistical methods. Often, these systems overcame exactly the problems that plague NLG systems: brittleness, domain-dependency, labour-intensive rule construction, and the inability to formulate clear criteria of choice in symbolic terms.

Over the past 4 years, an entirely new type of language generation system has made its appearance: the generator based on statistical knowledge. Are we witnessing the birth of a new paradigm in NLG? Will statistical systems allow NLG to evolve from an essentially research-only area to an area with true application-level technology?

This session is devoted to understanding better the opportunities and limits of statistics-based NLG. It will focus on three major points:

1. What does 'statistical NLG' mean, exactly?

   ... three case studies, in brief

2. What is 'statistical' knowledge? Can all the knowledge required for NLG be 'statistical'? If not, why not?

   ... general characteristics of 'statistical' knowledge in NLP systems, and the nature of the four kinds of knowledge required for NLG

3. How can one expect research on statistical NLG to proceed, in general terms? What will statistics not be able to do (ever)?

   ... a hierarchy of increasing sophistication of statistical models. The kinds of things statistical models do not do

Burning Issues Session

Summarization and Generation

Daniel Marcu, USC/ISI, Los Angeles

During the last five years, dozens of "summarization" systems have been produced by University and Research Labs, News Providers, and Internet-based DotComs. The vast majority of these "summarizers" are extraction systems: they identify clauses and sentences that are important in the input texts; and they catenate them to often produce incoherent outputs that contain dangling references and abrupt topic shifts.

Traditionally, the NLG community has focused on mapping abstract representations into well-written texts. But recently established markets desperately need NLG technologies capable of producing coherent texts out of text fragments extracted from single and multiple documents, which may be written at different levels of competency in multiple languages and styles. Over the next five years, will these markets induce the NLG community to shift its research focus? Will the community end up concentrating primarily on generating well-written texts out of text fragments and/or badly-written texts? What algorithms and techniques are needed to solve this type of generation problem?

This session is devoted to discussing open problems and opportunities that lie at the boundary between text summarization and natural language generation.

Burning Issues Session

What are reusable modules for NLG?

Chris Mellish, University of Edinburgh:

Questions to be addressed will include:

- Are we mature enough to produce reusable modules (other areas are)? If not, how can we get that maturity?
- What would a reusable module be like (how would its behaviour be defined? what background assumptions would make it easier to clearly define such modules and actually use them in many situations?)
- How can we optimise reusability given the diversity of theoretical approaches in NLG?

I would start by describing the essential elements of the RAGS approach to this. That is, of course, only one view and it would be very interesting to see what views come from other directions.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

-Bibliothek, Information
und Dokumentation (BID)-
Postfach 20 80
D-67608 Kaiserslautern
Germany

Telefon (0631) 205-3506
Telefax (0631) 205-3210

e-mail
bib@dfki.uni-kl.de
WWW
http://www.dfki.uni-sb.de/dfkibib

Veröffentlichungen des DFKI

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder (so sie als per ftp erhaeltlich angemerkt sind) per anonymous ftp von ftp.dfki.uni-kl.de (131.246.241.100) im Verzeichnis pub/Publications bezogen werden. Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or (if they are marked as obtainable by ftp) by anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) in the directory pub/Publications.
The reports are distributed free of charge except where otherwise noted.

DFKI Research Reports

2000

RR-00-01
Michael Schillo
Vertrauen und Betrug in Multi-Agenten Systemen
Erweiterung des Vertrauensmodells von Castelfranchi und Falcone um eine Kommunikationskomponente
11 Seiten

1999

RR-99-04
Gero Vierke, Christian Ruß
The Matrix Auction: A Mechanism for the Market-Based Coordination of Enterprise Networks
11 pages

RR-99-03
Christian Gerber, Jörg Siekmann, Gero Vierke
Holonic Multi-Agent Systems
42 pages

RR-99-02
Michael Schillo, Jürgen Lind, Petra Funk, Christian Gerber,
Christoph Jung
SIF - The Social Interaction Framework
System Description and User's Guide to a Multi-Agent System Testbed
30 pages

RR-99-01
Jürgen Lind, Stefan Philipps
Ein System zur Definition und Ausführung von Protokollen für Multi-Agentensysteme
61 Seiten

1998

RR-98-04
Bernd Kiefer, Hans-Ulrich Krieger
A Bag of Useful Techniques for Efficient and Robust Parsing
9 pages

RR-98-03
Heiko Mantel
Developing a Matrix Characterization for $\mathcal{MELL}$
59 pages

RR-98-02
Klaus Fischer, Christian Ruß, Gero Vierke
Decision Theory and Coordination in Multiagent Systems
134 pages

RR-98-01
Christoph G. Jung, Klaus Fischer
Methodological Comparison of Agent Models
58 pages

1997

RR-97-08
Stefan Müller
Complement Extraction Lexical Rules and Argument Attraction
14 pages

RR-97-07
Stefan Müller
Yet Another Paper about Partial Verb Phrase Fronting in German
26 pages

RR-97-06
Stefan Müller
Scrambling in German – Extraction into the *Mittelfeld*
24 pages

RR-97-05
Harald Meyer auf'm Hofe
Finding Regions of Local Repair in Hierarchical Constraint Satisfaction
33 pages

RR-97-04
Serge Autexier, Dieter Hutter
Parameterized Abstractions used for Proof-Planning
13 pages

RR-97-03
Dieter Hutter
Using Rippling to Prove the Termination of Algorithms
15 pages

RR-97-02
Stephan Busemann, Thierry Declerck, Abdel Kader Diagne, Luca Dini, Judith Klein, Sven Schmeier
Natural Language Dialogue Service for Appointment Scheduling Agents
15 pages

RR-97-01
Erica Melis, Claus Sengler
Analogy in Verification of State-Based Specifications: First Results
12 pages

1996

RR-96-06
Claus Sengler
Case Studies of Non-Freely Generated Data Types
200 pages

RR-96-05
Stephan Busemann
Best-First Surface Realization
11 pages

RR-96-04
Christoph G. Jung, Klaus Fischer, Alastair Burt
Multi-Agent Planning
Using an *Abductive*
EVENT CALCULUS
114 pages

RR-96-03
Günter Neumann
Interleaving
Natural Language Parsing and Generation
Through Uniform Processing
51 pages

RR-96-02
E.André, J. Müller , T.Rist:
PPP-Persona: Ein objektorientierter Multimedia-Präsentationsagent
14 Seiten

RR-96-01
Claus Sengler
Induction on Non-Freely Generated Data Types
188 pages

1995

RR-95-20
Hans-Ulrich Krieger
Typed Feature Structures, Definite Equivalences, Greatest Model Semantics, and Nonmonotonicity
27 pages

RR-95-19
Abdel Kader Diagne, Walter Kasper, Hans-Ulrich Krieger
Distributed Parsing With HPSG Grammar
20 pages

RR-95-18
Hans-Ulrich Krieger, Ulrich Schäfer
Efficient Parameterizable Type Expansion for Typed Feature Formalisms
19 pages

RR-95-17
Hans-Ulrich Krieger
Classification and Representation of Types in TDL
17 pages

RR-95-16
Martin Müller, Tobias Van Roy
Title not set
0 pages

Note: The author(s) were unable to deliver this document for printing before the end of the year. It will be printed next year.

RR-95-15
Joachim Niehren, Tobias Van Roy
Title not set
0 pages

Note: The author(s) were unable to deliver this document for printing before the end of the year. It will be printed next year.

RR-95-14
Joachim Niehren
Functional Computation as Concurrent Computation
50 pages

RR-95-13
Werner Stephan, Susanne Biundo
Deduction-based Refinement Planning
14 pages

RR-95-12
Walter Hower, Winfried H. Graf
Research in Constraint-Based Layout, Visualization, CAD, and Related Topics: A Bibliographical Survey
33 pages

RR-95-11
Anne Kilger, Wolgang Finkler
Incremental Generation for Real-Time Applications
47 pages

RR-95-10
Gert Smolka
The Oz Programming Model
23 pages

RR-95-09
M. Buchheit, F. M. Donini, W. Nutt, A. Schaerf
A Refined Architecture for Terminological Systems: Terminology = Schema + Views
71 pages

RR-95-08
Michael Mehl, Ralf Scheidhauer, Christian Schulte
An Abstract Machine for Oz
23 pages

RR-95-07
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt
The Complexity of Concept Languages
57 pages

RR-95-06
Bernd Kiefer, Thomas Fettig
FEGRAMED
An interactive Graphics Editor for Feature Structures
37 pages

RR-95-05
Rolf Backofen, James Rogers, K. Vijay-Shanker
A First-Order Axiomatization of the Theory of Finite Trees
35 pages

RR-95-04
M. Buchheit, H.-J. Bürckert, B. Hollunder, A. Laux, W. Nutt,
M. Wójcik
Task Acquisition with a Description Logic Reasoner
17 pages

RR-95-03
Stephan Baumann, Michael Malburg, Hans-Guenther Hein, Rainer Hoch,
Thomas Kieninger, Norbert Kuhn
Document Analysis at DFKI
Part 2: Information Extraction
40 pages

RR-95-02
Majdi Ben Hadj Ali, Frank Fein, Frank Hoenes, Thorsten Jaeger,
Achim Weigel
Document Analysis at DFKI
Part 1: Image Analysis and Text Recognition
69 pages

RR-95-01
Klaus Fischer, Jörg P. Müller, Markus Pischel
Cooperative Transportation Scheduling
an application Domain for DAI
31 pages

DFKI Technical Memos

2000

TM-00-01
Jürgen Lind
Specifying Agent Interaction Protocols with UML Activity Diagrams
12 pages

1999

TM-99-04
Christoph Endres
The MultiHttpServer - A Parallel Pull Engine
18 pages

TM-99-03
Jürgen Lind
A Process Model for the Design of Multi-Agent Systems
20 pages

TM-99-02
Hans-Jürgen Bürckert, Petra Funk, Gero Vierke
An Intercompany Dispatch Support System for Intermodal Transport Chains
12 pages

TM-99-01
Matthias Fischmann
The Smes Client/Server Protokol (SMESPR/1.0)
10 pages

1998

TM-98-09
Jürgen Lind
The EMS Model
15 pages

TM-98-08
Michael Schillo, Petra Funk
Spontane Gruppenbildung in künstlichen Gesellschaften
10 Seiten

TM-98-07
Markus Perling
The RAWAM: Relfun-Adapted WAM Emulation in C
49 pages

TM-98-06
Petra Funk, Gero Vierke, Hans-Jürgen Bürckert
A Multi-Agent Perspective on Intermodal Transport Chains
8 pages

TM-98-05
Jürgen Lind, Klaus Fischer
Transportation Scheduling and Simulation in a Railroad scenario: A Multi-Agent Approach
17 pages

TM-98-04
Hans-Jürgen Bürckert, Gero Vierke
Simulated Trading Mechanismen für Speditionsübergreifende Transportplanung
12 pages

TM-98-03
Petra Funk
Fast Loading and Unloading Devices: Planning and Scheduling Requirements
7 pages

TM-98-02
Christian Gerber, Christian Ruß, Gero Vierke
An Empirical Evaluation on the Suitability of Market-Based Mechanisms for Telematics Applications
20 pages

TM-98-01
Christian Gerber
Bottleneck Analysis as a Heuristic for Self-Adaption in Multi-Agent Societies
16 pages

1997

TM-97-03
Hans-Jürgen Bürckert, Klaus Fischer, Gero Vierke
TeleTruck: A Holonic Fleet Management System
10 pages

TM-97-02
Christian Gerber
Scalability of Multi-Agent Systems - Proposal for a Dissertation
49 pages

TM-97-01
Markus Perling
GeneTS: A Relational-Functional Genetic Algorithm for the Traveling Salesman Problem
26 pages

1996

TM-96-02
Harold Boley
Knowledge Bases in the World Wide Web:
A Challenge for Logic Programming
(Second, Revised Edition)
10 pages

TM-96-01
Gerd Kamp, Holger Wache
CTL — a description Logic with expressive concrete domains
19 pages

1995

TM-95-04
Klaus Schmid
Creative Problem Solving
and
Automated Discovery
— An Analysis of Psychological and AI Research —
152 pages

TM-95-03
Andreas Abecker, Harold Boley, Knut Hinkelmann, Holger Wache,
Franz Schmalhofer
An Environment for Exploring and Validating Declarative Knowledge
11 pages

TM-95-02
Michael Sintek
FLIP: Functional-plus-Logic Programming
on an Integrated Platform
106 pages

TM-95-01
Martin Buchheit, Rüdiger Klein, Werner Nutt
Constructive Problem Solving: A Model Construction
Approach towards Configuration
34 pages

DFKI Documents

2000

1999

D-99-01
Tilman Becker, Stephan Busemann
May I Speak Freely? Between Templates and Free
Choice in Natural Language Generation. Workshop
at the 23rd German Annual Conference for Artificial
Intelligence (KI '99), Bonn 14.-15. September 1999
69 pages

1998

D-98-03
Stephan Busemann, Karin Harbusch, Stefan Wermter(Hrsg.)
Hybride konnektionistische, statistische und regelbasierte Ansätze zur Verarbeitung natürlicher Sprache
Workshop auf der 21. Deutschen Jahrestagung für Künstliche Intelligenz, Freiburg, 9.–10. September 1997
75 Seiten

D-98-02
Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann , Otto Kühn,
Michael Sintek
Techniques for Organizational Memory Information Systems
66 pages

D-98-01
Stephan Baumann, Jürgen Lichter, Michael Malburg, Heiko Maus,
Harald Meyer auf'm Hofe, Claudia Wenzel
Architektur für ein System zur Dokumentanalyse im Unternehmenskontext Integration von Datenbeständen, Aufbau- und Ablauforganisation
76 Seiten

1997

D-97-08
Christoph G. Jung, Klaus Fischer, Susanne Schacht
Distributed Cognitive Systems
Proceedings of the VKS'97 Workshop
50 pages

D-97-07
Harold Boley, Bernd Bachmann, Christian Blum, Christian Embacher,
Andreas Lorenz, Jamel Zakraoui
PIMaS:
Ein objektorientiert-regelbasiertes System zur Produkt-Prozeß-Transformation
45 Seiten

D-97-06
Tilman Becker, Stephan Busemann, Wolfgang Finkler
DFKI Workshop on Natural Language Generation
67 pages

D-97-05
Stephan Baumann, Majdi Ben Hadj Ali, Jürgen Lichter, Michael Malburg,
Harald Meyer auf'm Hofe, Claudia Wenzel
Anforderungen an ein System zur Dokumentanalyse im Unternehmenskontext
— Integration von Datenbeständen, Aufbau- und Ablauforganisation
42 Seiten

D-97-04
Claudia Wenzel, Markus Junker
Entwurf einer Patternbeschreibungssprache
für die Informationsextraktion
in der Dokumentanalyse
24 Seiten

D-97-03
Andreas Abecker, Stefan Decker, Knut Hinkelmann, Ulrich Reimer
Proceedings of the Workshop „Knowledge-Based Systems for Knowledge Management in Enterprises" 97
167 pages

D-97-02
Tilman Becker, Hans-Ulrich Krieger
Proceedings of the Fifth Meeting on Mathematics of Language (MOL5)
168 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-97-01
Thomas Malik
NetGLTool Benutzeranleitung
40 Seiten

1996

D-96-07
Technical Staff
DFKI Jahresbericht 1995
55 Seiten

Note: This document is no longer available in printed form.

D-96-06
Klaus Fischer (Ed.)
Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems
63 pages

D-96-05
Martin Schaaf
Ein Framework zur Erstellung verteilter Anwendungen
94 pages

D-96-04
Franz Baader, Hans-Jürgen Bürckert, Andreas Günter, Werner Nutt (Hrsg.)
Proceedings of the Workshop on Knowledge Representation and Configuration WRKP'96
83 pages

D-96-03
Winfried Tautges
Der DESIGN-ANALYZER - Decision Support im Designprozess
75 Seiten

D-96-01
Klaus Fischer, Darius Schier
Ein Multiagentenansatz zum Lösen von Fleet-Scheduling-Problemen
72 Seiten

1995

D-95-12
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)
Working Notes of the KI'95 Workshop:
KRDB-95 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
61 pages

D-95-11
Stephan Busemann, Iris Merget
Eine Untersuchung kommerzieller Terminverwaltungssoftware im Hinblick auf die Kopplung mit natürlichsprachlichen Systemen
32 Seiten

D-95-10
Volker Ehresmann
Integration ressourcen-orientierter Techniken in das wissensbasierte Konfigurierungssystem TOOCON
108 Seiten

D-95-09
Antonio Krüger
PROXIMA: Ein System zur Generierung graphischer Abstraktionen
120 Seiten

D-95-08
Technical Staff
DFKI Jahresbericht 1994
63 Seiten

Note: This document is no longer available in printed form.

D-95-07
Ottmar Lutzy
Morphic - Plus
Ein morphologisches Analyseprogramm für die deutsche Flexionsmorphologie und Komposita-Analyse
74 Seiten

D-95-06
Markus Steffens, Ansgar Bernardi
Integriertes Produktmodell für Behälter aus Faserverbundwerkstoffen
48 Seiten

D-95-05
Georg Schneider
Eine Werkbank zur Erzeugung von 3D-Illustrationen
157 Seiten

D-95-04
Victoria Hall
Integration von Sorten als ausgezeichnete taxonomische Prädikate in eine relational-funktionale Sprache
56 Seiten

D-95-03
Christoph Endres, Lars Klein, Markus Meyer
Implementierung und Erweiterung der Sprache $\mathcal{ALCP}$
110 Seiten

D-95-02
Andreas Butz
BETTY
Ein System zur Planung und Generierung informativer Animationssequenzen
95 Seiten

D-95-01
Susanne Biundo, Wolfgang Tank (Hrsg.)
PuK-95, Beiträge zum 9. Workshop „Planen und Konfigurieren", Februar 1995
169 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

IMPACTS in Natural Language Generation
NLG Between Technology and Applications

Workshop at Schloss Dagstuhl, Germany
July 26-28, 2000

Tilman Becker, Stephan Busemann (eds.)

D-00-01
Document