Towards a Methodology for Developing Application-Oriented Report Generation*

Helmut Horacek[1] and Stephan Busemann[2]

[1] Universität des Saarlandes, FB 14, Informatik
Postfach 1150, D-66041 Saarbrücken
horacek@cs.uni-sb.de

[2] DFKI GmbH
Stuhlsatzenhausweg 3, D-66123 Saarbrücken
busemann@dfki.de

Abstract. Although research in natural language generation has led to the development of numerous methods and reusable software tools, we feel that building comparably simple application systems still involves more hand-crafted skills than systematic methodology. In our view, this is due to the fact that most available tools are oriented towards contributing to a general purpose generation system rather than supporting the economic development of dedicated applications. In order to improve this situation, we present a methodology for developing application-oriented report generation with limited effort, emphasizing domain- and user-specific preferences over general-purpose communicative principles. Key parts in our approach comprise building an ontologically minimal initial representation on the basis of user parameters and associated domain data, the successive refinement of this initial representation by making implicit information explicit enough for fleshing out selected text and sentence patterns, and the opportunistic combination of linguistically motivated methods with template-based generation. This methodology should enable system developers to build application-oriented report generators more systematically and with reduced effort.

1 Introduction

Although research in natural language generation (NLG) has led to the development of numerous methods and reusable software tools, we feel that building comparably simple application systems today still involves more hand-crafted skills than systematic methodology. In our view, this is due to the fact that most available tools (e.g. SURGE [Elhadad and Robin, 1996], KPML [Bateman, 1997]) are based on *in-depth* approaches to NLG contributing to a general purpose generation system rather than supporting the economic development of dedicated applications. They require, for instance, detailed input specifications,

* This work has been supported by a grant from the European Union to the project TEMSIS (Telematics Applications Programme, Sector C9, contract no. 2945).

some of them being irrelevant for the application in hand. Even more importantly, today's tools rarely help in organizing application conventions systematically and efficiently, and in relating non-linguistic to linguistic specifications at whatever level of description, ranging from deep semantic representations to surface forms.

While in-depth approaches to NLG are indispensable for scientific progress in the field, we suggest that certain types of applications can, at present, be built more successfully by systematically adopting more *shallow* techniques, which emphasize domain- and user-specific preferences over general-purpose communicative principles (cf. [Busemann and Horacek, 1998] for a more extensive argumentation). Motivated by the application-related deficits of general tools, we have developed a methodology for shallow multi-lingual report generation. Key parts in our approach comprise building an ontologically minimal initial representation on the basis of user parameters and associated domain data, the successive refinement of this initial representation by making implicit information explicit enough for fleshing out selected text and sentence patterns, and the opportunistic combination of linguistically motivated methods with template-based generation. Our shallow methodology was put to the test within the multi-lingual report generator developed for the TEMSIS project.

The paper is organized as follows. The TEMSIS application environment is outlined in Section 2. Section 3 describes and motivates the methods for building a report generator in a systematic and efficient manner. Based on the experience gained, we discuss the process of developing systems of a similar kind in Section 4. Finally, Section 5 characterizes the applicability conditions and the limitations of our approach.

2 The TEMSIS NL Generation Application Environment

In this section, we illustrate the goals pursued in TEMSIS from an application-oriented perspective. With TEMSIS, a **T**ransnational **E**nvironmental **M**anagement **S**upport and **I**nformation **S**ystem was created as part of a transnational cooperation between the communities in the French-German urban agglomeration, Moselle Est and Stadtverband Saarbrücken. Networked information kiosks are being installed in a number of communities to provide public and expert environmental information.

The timely availability of relevant environmental information will improve the planning and reactive capabilities of the administration considerably. Current measurement data are made available on the TEMSIS web server. The data include the measurement values of several pollutants, the location and the time the measurements were taken, and a variety of thresholds. Besides such data, the server provides metadata that allow for descriptions of the measuring locations, of the pollutants measured and of regulations or laws according to which a comparison between measurements and thresholds can be performed. This information can be accessed via the internet through a hyperlink navigation interface.[3]

The verbalization of NL air quality information in French and German is an additional service to the environmental administrations reducing the need to look up multiple heterogeneous data. Reports contain statements about the average and maximum values of the concentration of air pollutants over certain periods of time, about relations to threshold values, about comparisons with values from preceeding time periods, canned text descriptions of measuring locations and pollutants, and diagrams showing long-term developments. The generated texts can be edited and further processed by the administrations to fit additional needs.

The structures of the texts originated from discussions with the TEMSIS user group. The reports were classified into seventeen types. The user group also provided a set of sample texts, which we categorized into nine assertion types. This categorization helped us to accomodate the wordings in the reports to the demands of the domain.

In order to obtain a report, a user specifies his demands by choosing from a hierarchy of options presented to him within the hyperlink navigation interface. The user has to specify a pollutant, a measuring location, a time interval, and in some cases comparison parameters, which leads to the instantiation of one of the seventeen report types. In addition, descriptions of pollutants and measurement stations can be requested. These are stored as canned texts in the TEMSIS database. In (1) a sample user request is shown followed by an excerpt of the resulting text translated into English. Altogether, the system can generate 384 report structures that differ in at least one linguistic aspect.

```
(1) :LANGUAGE GERMAN
    :POLLUTANT SO2
    :MEASURING STATION "Carling"
    :REPORT-TYPE AVERAGE
    :TIME January/98
```

The concentration of sulfur dioxyde in the air at the measurement station "Carling" in January 1998 was 20 $\mu g/m^3$, so that the threshold value I1 of 140 $\mu g/m^3$ for a 30 minute interval according to TA-Luft has not been exceeded (an annual threshold value is related to a daily measurement value here). In December 1997, the value was 25 $\mu g/m^3$, so that the threshold value has not been exceeded either.

The development of the TEMSIS report generator had to be carried out under rather tight resource constraints. With an investigation of eight person months, the system had to be built from scratch in large parts. We decided to reuse the TG/2 production system for shallow linguistic verbalization [Busemann, 1996] as well as some of its subgrammars including e.g. rules for temporal expressions. We looked for simple and application-oriented solutions, at the same time envisioning systematicity and a methodological development process.

[3] The URL is `http://www-temsis.dfki.uni-sb.de/`.

We refrained from reusing large in-depth NLG resources such as SURGE [Elhadad and Robin, 1996] or KPML [Bateman, 1997], because their adaptation to the needs of TEMSIS would have required considerable effort for developing interfaces to the feeding components. Moreover, the use of such resources within relatively small applications like the present one means a certain overshot.

Altogether, we believe that both the kind of functional requirements and the need of keeping the develoment effort very small are by far not unique for the TEMSIS report generation scenario. Our aim is thus to elaborate a methodology for developing systems of this kind efficiently.

3 Report Generation in TEMSIS

In this section, we illustrate the techniques used within the TEMSIS report generator. We first introduce some terms that are central for our methods:

- *User specifications* or *parameters* are the input data which trigger the composition of a report.
- These specifications play the role of *influence factors* in individual report parts, and they may be interpreted differently according to the specific assertion types. For instance, while the expression "sulfur dioxyde" used for confirming the user's choice of pollutant is interpreted as the type of pollutant, that same expression refers to the concentration of the pollutant in the air if it occurs within an assertion about threshold passing.
- *Variations* in the reports and their statements break down into two sorts. Linguistically motivated variations mainly originate from aggregation operations and depend on the target language chosen, with some of the aggregations made possible due to peculiar data constellations (e.g., identical value comparison outcomes). Variations in content have their source purely in data-specific constellations. For instance, if a measured value does not exceed some threshold, an otherwise conveyed comparison with a higher threshold is skipped.

We have partitioned the generation process into three phases (disregarding the access to the database according to the user parameters, which yields actual measurement data):

1. Organizing the material to convey in terms of single statements on the basis of user parameters.
2. Recasting the domain-oriented specifications in language-adequate, but language-neutral, terms by successively refining the specifications obtained so far.
3. Transducing the resulting intermediate structure into French or German texts, by an opportunistic combination of linguistically motivated methods with template generation.

With major simplifications, these tasks correspond to the classical phases of text planning, lexicalization, and realization, organized in a traditional pipe-line architecture. However, some of our techniques and the interfaces involved deviate from usual standards. Due to the conventionalized nature of our reports, using linguistically motivated methods, such as RST-based text planners, would be an overshot. Moreover, in-depth lexicalization methods are only an option in our approach, but we are using more surface oriented processes due to the limited functional requirements and available resources. While content realization does not encode meaning by itself, the template-based verbalization in TG/2 uses rules depending on the domain and the task at hand. Such dependencies render a simple pipe-lined architecture feasible.

In the following, we elaborate the main concepts in each phase and illustrate them with sample representations in English for the reader's convenience.

3.1 Phase 1: Text organization

Organizing the material that goes into a report is supported by two sorts of data structures:

- *Report skeletons* describe the application conditions of report types and provide a basic text structure to be filled with single statements.
- *Assertion structures* capture data-driven options and variations in single statements and bind relevant user parameters and domain data – the influence factors – to these statements.

Report skeletons consist of lists of assertion structures. In a report skeleton, the application conditions serve to match user specifications for selecting a unique report type, to access domain data according to these specifications, and to activate further specifications implicitly associated with a report type, such as suitable threshold specifications. Selecting a report skeleton causes the associated assertion structures to be instantiated.

```
(2)  (AVERAGE POLLUTANT
              MEASUREMENT-STATION
              TIME
              (#'DESCRIBE-VALUE VALUE)
              (#'DESCRIBE-THRESHOLD THRESHOLD)
              (#'DESCRIBE-COMPARISON COMPARISON))
```

Assertion structures consist of a small number of condition-action pairs evaluated like a case statement. Example (2) is an assertion structure for a statement about an average value including a threshold comparison, such as the sample text following (1). It is referred to from report skeletons representing average value reports, and is picked up in case a suitable threshold for the selected pollutant is available in view of the user's time specification. The elements of this assertion structure are interpreted in three different ways (see the instantiated structure in (3)):

- The first predicate, in (2) `AVERAGE`, becomes the value of the `PRED` feature of the assertion to be built.

- Atomic elements, in (2) `POLLUTANT`, `MEASUREMENT-STATION`, and `TIME`, indicate that the referred elements of the user specification are to be copied into the assertion, where they appear as attribute-value pairs.
- List elements are evaluated as procedure calls. Their parameters denote database access results for the selected report. These procedures yield descriptions of the relevant data, either as a simple attribute-value pair for atomic values `(VALUE 20)`, or as an attribute with a list of attribute-value pairs as its value (the values of the attributes `THRESHOLD` and `EXCEEDS`).

Example (3) shows the instantiated assertion structure, originating from (2), for the central statement of the sample text in (1).

```
(3) ((PRED AVERAGE)
     (POLLUTANT SO2)
     (MEASUREMENT-STATION "Carling")
     (TIME ((YEAR 1998) (MONTH 1)))
     (VALUE 20)
     (THRESHOLD ((AMOUNT 140)
                 (THRESHOLD-TIME YEARS)
                 (LAW-NAME TA-LUFT)
                 (THRESHOLD-NAME I1)
                 (MEASUREMENT-DURATION 30-MINUTES)))
     (EXCEEDS ((TIMES 0))))
```

Altogether, organizing the material to convey results in filling these data structures by interpreting user parameters accordingly. The interpretation of user parameters yields at least a partial order of statements out of an unordered set of specifications. Ontologically, the semantically condensed predicates in the user specifications are split into operationally interpretable predicates, with context-sensitive and conventionally motivated details added according to statement types. Restrictions in this interpretation process include the absence of recursion in statement expansions, and a lack of convenience in expressing data dependencies across statements – identical expressions need to be included in several assertion structures referred to by some report skeleton.

The internal organization of statement specifications is encoded in a condensed form: a top level predicate represents the type of each assertion and encapsulates the entire meaning of the associated assertion with the exception of the attached influence factors. Each of the influence factors is represented as an attribute value pair, linked to the top level predicate by an unambiguous attribute. The value itself may be one of the following:

- An atomic value, such as an option, or a name; an example is the pollutant identifier (see (3)).
- A list of values, such as a data vector; an example is a list of measurement values.
- An attribute value structure, for structured influence factors with internal dependencies; an example is a time specification (see (3)).

Altogether, this categorization imposes some sort of minimal ontology on the initial proposition representation to encapsulate 'domain communication knowledge' [Kittredge *et al.*, 1991], and it makes local parameter and data dependencies explicit.

3.2 Phase 2: Recasting information

The second processing phase, recasting the information, manipulates initial representations of assertion structures like (3). It takes care of augmenting, restructuring, and aggregating them, to meet the properties of the TG/2 verbalizer. In particular, no linguistic facts are made explicit since this is left to TG/2.

Augmenting statement specifications means making information implicitly entailed or available elsewhere explicit at the place it is needed. This concerns reestablishing report-wide information, as well as making locally entailed information accessible. An example for the latter is the unit in which the value of a measurement is expressed (see (4)). Examples for the former are the choice of the target language, and more interestingly, the number of diagrams depicted, copied into the introductory statement to these diagrams. This treatment is much simpler than using a reference generation algorithm, but it relies on a fixed number of diagrams for a given report skeleton.

Example (4) shows the result obtained for (3) except to aggregation, with changes in bold face.

(4) ((PRED AVERAGE)
(POLLUTANT SO2)
(LANGUAGE GERMAN)
(MEASUREMENT-STATION "Carling")
(TIME ((YEAR 1998) (MONTH 1)))
(VALUE **((AMOUNT 20) (UNIT mg/m3)**))
(THRESHOLD ((AMOUNT 140) **(UNIT mg/m3)**))
(SOURCE ((LAW-NAME TA-LUFT)
(THRESHOLD-NAME I1)
(PERIOD ((MEASUREMENT-TIME DAYS)
(THRESHOLD-TIME YEARS)))))
(MEASUREMENT-DURATION 30-MINUTES)
(EXCEEDS ((TIMES 0))))

TG/2-oriented recasting ranges from renaming an attribute over reifying it as a structured value, such as `(AMOUNT 20)` in (4), raising an embedded partial description by one level, such as `(MEASUREMENT-DURATION 30-MINUTES)` in (4), to complex operations such as building lists of attribute value pairs from data vectors. Most of these operations are data-driven utilizing schemata as they were elaborated for linguistically motivated lexicalization [Horacek, 1996]. In contrast to that work, some of our schemata are purely application-oriented, and they are 'heavier' in terms of the number of schema parameters and the size of the structures covered. In TEMSIS, we use twelve schema classes and about hundred instantiations.

In other approaches, the lexicalization process and the structural mappings involved are determined by the lexemes and the grammatical structures in the target language. In contrast, our approach breaks down the composition through mapping schemata only as far as distinctions between phrasings are required. In particular, the treatment of language specificity is handed over to TG/2. This design, which results in a much smaller number of mapping schemata, has two important consequences:

- The search control is eased significantly, because a single mapping schema is available only in most cases.
- The coarse-grained partitioning makes writing the schemata much easier, so that problems of expressibility are unlikely to occur.

As a last part of information recasting, some sorts of aggregation are performed. Operations that remove partial descriptions or add simple structures are invoked, driven by a small set of declaratively represented rules, operating on a discourse memory. Most of the rules aim at avoiding repetitions of optional constituents (e.g., temporal and locative information) over adjacent statements. The rules are formulated to meet application particularities, such as impacts of certain combinations of a value and a threshold comparison outcome, rather than to capture linguistic principles. Before aggregation, the internal representations of the two sentences in the text following (1) are identical except to the value and the month. After aggregation, the representation of the second sentence is reduced to (5), the newly added `REPEAT` slot giving raise to the use of *either* in TG/2. Several parts have been elided, as a comparison with (4) reveals.

```
(5) ((PRED AVERAGE) (LANGUAGE GERMAN)
     (TIME ((YEAR 1997) (MONTH 12)))
     (VALUE ((AMOUNT 25) (UNIT mg/m3)))
     (EXCEEDS ((TIMES 0))) (REPEAT YES))
```

3.3 Phase 3: Linguistic verbalization

Finally, our dedicated verbalization component, TG/2, selects appropriate production rules for the sorts of statements and some key parameters. The productions consist of conditioned rewrite rules with a context-free backbone. Their right-hand side consists of canned text elements, lexemes, or non-terminal symbols corresponding to recursive rule activation. This way, canned texts, template techniques and context-free grammars are integrated into a single formalism [Busemann, 1996].

The string of template elements in (6)[4] corresponds to the right-hand side of a rule verbalizing average values. The applicability conditions for this rule include the requirement that the TG/2 input contain the specification `(PRED AVERAGE)`.

[4] Capitalized words within brackets are place holders; a question mark following an opening bracket indicates an optional constituent.

```
(6)  "the concentration of " <POLLUTANT> "in the air "
     <? "at" MEASUREMENT-STATION> <? "on" DATE> "was "
     <VALUE> "(" <T1> "is related to " <T2> "here)."
```

TG/2 improves over existing tools that provide comparably poor support for both complex phrasings and for sychronising their functionality across several languages. Phrasal composition comprises fine-grained parts, such as time and value descriptions, as well as rather coarse-grained ones, such as the example above, where the expression in parentheses is triggered by the difference of the attribute values of (MEASUREMENT-TIME <T1>) and (THRESHOLD-TIME <T2>) (e.g. *an annual threshold is related to monthly values here*).

This organisation works in a comparably simple way without the need to model the associated linguistic features and their cross-language differences in full detail. More precisely, TG/2 allows us to model exactly at the level of granularity dictated by the domain, meeting our resource constraints well.

4 Developing an Application-Oriented Report Generator

In this section, we illustrate the development process of an application-oriented report generator. We abstract away from particularities of the TEMSIS scenario in order to obtain a development methodology for applications with similar properties. The development process can be divided into three major activities:

– Organizing the material to convey by structuring it in terms of proposition sequences, and making influence factors on individual propositions explicit.
– Designing an intermediate representation structure mediating between statement specifications and a suitable verbalization component.
– Fleshing out data structures that support manipulations of the intermediate representation structure: mapping schemata, aggregation and grammar rules.

The first activity, organizing the material to convey, is based on user specifications. Report specifications and a corpus of sample texts, which are the two central building blocks that formally untrained users can and should provide to system builders, must be examined in terms of their cross-dependencies (cf. [Reiter *et al.*, 1997]). Sentence patterns must be abstracted from these examples, and they must be categorized and related to report specifications. This analysis should lead to definitions of report skeletons, assertion specifications, and influence factors, composed in a systematic way, as described in the previous section. There is a certain choice involved between having a larger number of report types, or assertion propositions with more conditionals.

The contribution of the process model underlying this activity lies in providing a specific approach to build initial representations with a widely free vocabulary that can be accomodated to elements in an associated knowledge base. Building such a draft specification is based on human pre-analysis to build schemata that bear some similarities to McKeown's [McKeown, 1985]. Our schemata can also represent conditionals, but they are flat structures without recursion, and their instantiation is geared by user-based specifications rather than by linguistically motivated concepts such as focus constraints.

The second activity, designing an intermediate representation structure, is influenced by two factors imposing different demands on this structure: (1) the distinctions necessary in the output texts must be represented explicitly and as economically as possible; (2) an available verbalization component should be made exploitable. Several divergent attitudes towards resolving the ontological differences between domain-oriented perspectives (which manifest themselves in initial representations) and language-oriented perspectives (which manifest themselves in – in-depth – intermediate representations) may result:

- For a canned text verbalization component, that difference is essentially ignored; necessary distinctions are made explicit, and resulting structures are directly converted into text templates.
- For a shallow verbalization component such as TG/2 (also some of the so-called intermediate techniques in IDAS [Reiter *et al.*, 1995] fall under this type), bridging that difference is transferred to the verbalization component's grammar rules;
- For an in-depth realization component such as SURGE, structure-changing rules should take care of bridging that ontological difference.

Depending on which approach is chosen, there is some sort of compensative effect between minimal development effort, as in the first alternative, and optimum extendability, as in the third alternative.

Finally, the third activity, fleshing out the various sorts of production rules involved, depends on choices made for the intermediate representation structure. For a shallow approach, building a small set of schemata that map complete sub-structures is frequently adequate. For more in-depth approaches, a much larger set of elementary schemata is required. In the former case, a certain amount of effort is certainly needed to enhance the set of schemata available for a new application. In the latter case, more effort is needed to analyze the differences between initial and intermediate representations in terms of their decomposition into schemata that are elementary enough for the ontological level of the intermediate representation structure. Similar trade-offs concern aggregation and grammar rules.

5 Applicability and Limitations

Our methodology is applicable to reports in which the organization principles underlying the information conveyed and the associated wordings reflect more the domain conventions and the user preferences than the linguistic concepts. These reports can exhibit only limited degrees of variations, their structure and content is widely determined by parameters, and reasoning about contents of a knowledge base should be restricted to subtasks with a local impact. All conventions and preferences are interpreted much more directly and concretely than typical rhetorical parameters for experimental systems, such as "colloquial", "verbose" etc., so that reasoning about them is not required.

Our methodology covers reports of a complexity comparable to those produced by Ana [Kukich, 1983], FOG [Boubeau *et al.*, 1990], IDAS [Reiter *et al.*, 1995], and it may in large parts be extendible to those produced by PLANDOC [McKeown *et al.*, 1994]. Ana is the first natural language report generator, and the kind of texts it produces can still be considered valuable today. However, Ana is implemented as a widely unstructured rule-based system, which does not seem to be easily extendable and portable. In comparison to that, we gain over Ana through several motivated representation levels with well-defined structures and vocabulary. Moreover, our generator is widely driven by easily readable and interpretable data structures rather than by collections of production rules for each of three processing phases in Ana. Hence, we assess our main contribution in comparison to Ana as an increase of systematicity, which rather similarly holds for FOG. In comparison to IDAS, we are lacking a component for referring expressions, but we gain in terms of multi-linguality. In comparison to PLANDOC, we also gain in terms of multi-linguality, and our approach requires significantly less development effort. However, PLANDOC has a number of more powerful components, such as the sorts of aggregation it can deal with, and it has available to it the generality of SURGE.

In this context, the limitations and prospects of our approach can be assessed as follows (cf. [Busemann and Horacek, 1998] for more details):

- Aggregation techniques can easily be extended, although their formulation would require rather accessing properties of domain parameters than evaluating in-depth linguistic features.
- Extending the approach by a reference mechanism is possible, but this would require major enhancements to the report structure and statement representation schema, and to the associated interpretation processes.
- Modeling the language structures explicitly in more detail is definitely an option that can and should be pursued. However, the more the flexibility is increased this way, the bigger become the problems of search control and mastering expressibility in the lexicalization phase.

6 Conclusion

In this paper, we have presented a methodology for developing application-oriented shallow report generators with comparably low development effort. We have described the TEMSIS generator as an instance for the domain of air-quality reports. The development process comprises building an ontologically minimal initial representation on the basis of user parameters and associated domain data, the successive refinement of this initial representation by making implicit information explicit enough for fleshing out selected text and sentence patterns, and the opportunistic combination of linguistically motivated methods with template-based generation. We have discussed a number of options in the development process, with divergent costs and benefits. While this methodology allows one to develop systems that can only produce reports of limited complexity, the comparably low effort associated makes our approach competitive.

References

[Bateman, 1997] John Bateman. KPML delvelopment environment: multilingual linguistic resource development and sentence generation. Report, German National Center for Information Technology (GMD), Institute for integrated publication and information systems (IPSI), Darmstadt, Germany, January 1997. Release 1.1.

[Boubeau *et al.*, 1990] L. Boubeau, D. Carcagno, E. Goldberg, R. Kittredge, and A. Polguere. Bilingual generation of weather forecasts in an operations environment. In *Proceedings of the 13 th International Conference on Computational Linguistics (COLING-90), Volume 1*, pages 90–92, Helsinki, 1990.

[Busemann and Horacek, 1998] Stephan Busemann and Helmut Horacek. A flexible shallow approach to text generation. In Eduard Hovy, editor, *Nineth International Natural Language Generation Workshop. Proceedings*, Niagara-on-the-Lake, Canada, 1998.

[Busemann, 1996] Stephan Busemann. Best-first surface realization. In Donia Scott, editor, *Eighth International Natural Language Generation Workshop. Proceedings*, pages 101–110, Herstmonceux, Univ. of Brighton, England, 1996. Also available at the Computation and Language Archive at `http://xxx.lanl.gov/abs/cmp-lg/9605010`.

[Elhadad and Robin, 1996] Michael Elhadad and Jacques Robin. An overview of SURGE: a reusable comprehensive syntactic realization component. In Donia Scott, editor, *Eighth International Natural Language Generation Workshop. Demonstrations and Posters*, pages 1–4, Herstmonceux, Univ. of Brighton, England, 1996.

[Horacek, 1996] Helmut Horacek. Lexical choice in expressing metonymic relations in multiple language. *Machine Translation*, 11:109–158, 1996.

[Kittredge *et al.*, 1991] Richard Kittredge, Tanya Korelsky, and Owen Rambow. On the need of domain communication knowledge. *Computational Intelligence*, 7:305–314, 1991.

[Kukich, 1983] Karen Kukich. Design and implementation of a knowledge-based report generator. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150, Cambridge, MA, 1983.

[McKeown *et al.*, 1994] Kathleen McKeown, Karen Kukich, and James Shaw. Practical issues in automatic documentation generation. In *Proceedings of the 4 th Conference on Applied Natural Language Processing (ANLP)*, pages 7–14, Stuttgart, 1994. Morgan Kaufmann.

[McKeown, 1985] Kathleen McKeown. Discourse strategies for generating natural language texts. *Artificial Intelligence*, 27:1–41, 1985.

[Reiter *et al.*, 1995] Ehud Reiter, Chris Mellish, and John Levine. Automatic generation of technical documentation. *Applied Artificial Intelligence*, 9, 1995.

[Reiter *et al.*, 1997] Ehud Reiter, Alison Cawsey, Liesl Osman, and Yvonne Roff. Knowledge acquisition for content selection. In *Proceedings of the 6th European Workshop on Natural Language Generation (ENLGWS-97)*, pages 117–126, Duisburg, 1997.

This article was processed using the LaTeX macro package with LLNCS style