Journal of Universal Computer Science, vol. 3, no. 8 (1997), 955-987
submitted: 1/6/97, accepted: 10/8/97, appeared: 28/8/97 © Springer Pub. Co.

From Natural Language Documents to Sharable Product Knowledge: A Knowledge Engineering Approach

Dietmar Rösner, Brigitte Grote, Knut Hartmann, Björn Höfling

Otto-von-Guericke-Universität Magdeburg
Institut für Informations- und Kommunikationssysteme
P.O.Box 41 20, D-39016 Magdeburg, Germany
Email: {roesner,grote,hartmann,hoefling}@iik.cs.uni-magdeburg.de

Abstract: A great part of the product knowledge in manufacturing enterprises is only available in the form of natural language documents. The know-how recorded in these documents is an essential resource for successful competition in the market. From the viewpoint of knowledge management, however, documents have a severe limitation: They do not capture the wealth of knowledge contained in these documents, since the entire knowledge is not spelled out on the linguistic surface. In order to overcome this limitation, the notion of a document as a particular kind of realization of (or view on) the underlying knowledge is introduced. The paper discusses the major steps in realizing this approach to documents: Knowledge acquisition, knowledge representation, and techniques to automatically generate multilingual documents from knowledge bases. Further, the paper describes how the required product knowledge can be represented in a sharable and reusable way.

Key Words: Artificial Intelligence, Knowledge Acquisition, Knowledge Representation Formalisms and Methods, Natural Language Processing, Language Generation, Natural Language Interfaces

Category: I.2.4 [Knowledge Representation Formalisms and Methods], I.2.7 [Natural Language Processing], H.4. [Information Systems Application], H.5 [Information Interfaces and Presentation]

1 Introduction

Currently, a great part of the product knowledge in manufacturing enterprises is only available in the form of natural language documents. Documents play a major role in the entire product life cycle, that is in design, manufacturing, quality assurance, marketing, maintenance and repair. The know-how that manifests itself in the form of such documents, for instance, product specifications, instruction manuals and trouble-shooting guides, is an essential resource for successful competition in the marketplace and should therefore be preserved and utilised as efficiently as possible.

Throughout their entire life cycle, documents are processed with computers: They are created, modified and updated with word processors, are stored in and retrieved from document data bases, are distributed over networks, and may be subject to transformations for printing or display. From the viewpoint of knowledge management, such document processing has a major limitation: It hardly captures the wealth of knowledge contained in these documents. In particular, it does not capture the knowledge implicitly referred to in the document. And even the knowledge that is explicitly expressed is only of limited use since it cannot be processed automatically and thus be exploited by different applications such as, for instance, automatic document generators.

In order to overcome this limitation, we suggest a revised understanding of what a document is: We do not regard a document as a static entity, but take it as a particular kind of realization of (or view on) underlying knowledge. The view realized by a particular document will typically be partial, as only part of the knowledge has to be spelled out on the linguistic surface of the text. A lot can be left implicit because readers will be able to reconstruct those parts of the content that are not verbalized by drawing on their background knowledge of the subject matter and on general 'world knowledge'.

Automating the creation of documents from underlying knowledge sources has many advantages for both knowledge management and document processing. Documents may be generated on demand. They may be tailored to different purposes and users, thus varying in detail, style and language. They may flexibly combine not only text but other modalities like, for instance, pictures and graphics. Finally, documents generated from knowledge structures may be used as an 'active' window on the underlying knowledge.

We have developed and exemplified this perspective on documents in our work with a particular kind of natural language document—multilingual technical manuals. In the TechDoc project [Rösner, 1994] [Rösner and Stede, 1994], we have analysed multilingual instructions with respect to the knowledge necessary to reconstruct these texts and have implemented a prototypical software system which demonstrates the feasibility of automatic generation of multilingual technical documents (currently in English, German and French) from such a knowledge base.

In this paper, we describe our approach from a knowledge engineering perspective. For knowledge acquisition, our main source of information has been an initial manual analysis of natural language documents from our domain. Section 2 reviews the techniques used and the results of this analysis. Our approach to knowledge representation and formalization is presented in section 3. Section 4 introduces the TechDoc system, which has been designed and implemented as a 'proof of concept' for the feasibility of automatic generation of multilingual technical documentations from a formal representation of content and thus for our perspective on documents. The presented approach has consequences for the management of corporate knowledge; these and the advantages of integrating documentations into the product life cycle are discussed in section 5. The paper concludes with a discussion of our approach in the context of related work in section 6, and an outlook on ongoing and future work in section 7.

2 Knowledge Acquisition from Natural Language Documents

In our understanding, documents are one means among others to express knowledge, and a particular document realizes a specific view on the underlying knowledge. However, one still faces two problems: First, one cannot access the entire knowledge contained in documents, because that knowledge is not completely spelled out on the linguistic surface. Instead, part of it has to be retrieved from the text by interpreting it within a broader understanding of the world ('world knowledge'). Second, neither knowledge that is expressed in the linguistic realization nor knowledge that has to be derived in the process of text understanding is available in a format ready for automatic processing. Instead, in order to spell out the entire knowledge contained in documents, and thus make it available for formal representation, we first have to identify the various kinds of knowledge that together give rise to a particular document. The task of *acquiring knowledge* from documents will be discussed in the remainder of this section. The knowledge acquisition phase results in specific demands on the knowledge bases as to what knowledge to represent and how to formalize it.

2.1 Approaches to Knowledge Acquisition

A number of knowledge acquisition techniques (KA) have been developed within the expert system community [Scott *et al.*, 1991], whereas knowledge acquisition has only recently become a research topic in work on natural language generation (NLG), see for instance [Reiter *et al.*, 1997]. Reiter presents four knowledge acquisition techniques, adapted from "standard" techniques in the expert system community: direct acquisition of knowledge by asking experts, creating and analysing a corpus, structured group discussions, and think-aloud sessions. The latter two mainly serve to explicate the intentions underlying certain expert decisions. At present, we are not concerned with documents that are of a persuasive nature, and therefore with the intentions motivating the production of documents. Instead, we want to obtain data on the domain knowledge encoded in documents and its realization through language. Thus, we restrict ourselves—for the time being—to the former two techniques.[1] Furthermore, we supplement the knowledge acquired this way with knowledge from additional sources.

In a nutshell, we take a twofold approach to knowledge acquisition:

– **Analysing:** We collect a range of multilingual documents from the present application domain—maintenance instructions taken from different domains such as car manuals, aircraft manuals and household appliances—and perform a thorough analysis of the documents in the corpus in terms of *content* and *structure*. A sample document from our corpus is given in figure 1.
– **Supplementing:** We supplement the resulting knowledge by knowledge acquired from experts such as mechanical engineers and technical authors. Further, we study relevant literature such as textbooks on technical documentation and mechanical engineering, guidelines and technical norms (e.g. ISO and DIN norms). Supplementing comprises two aspects:
  - enhancing the domain knowledge by simply adding more information,
  - structuring the knowledge by introducing more abstract levels of representation that combine the various fragments into a coherent whole.

Two additional points have to be mentioned with respect to the corpus analysis phase before describing its results: First, we performed a *contrastive* analysis—i.e. we studied a multilingual corpus containing German, English and French texts, and second, the analysis was conducted *manually*. The reasons for these decisions are as follows:

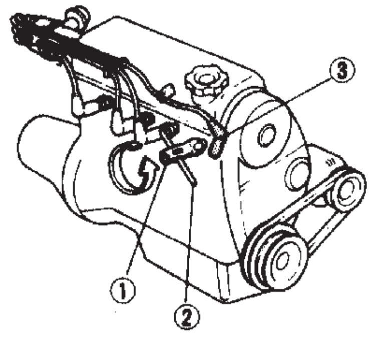
[1] We are aware, however, that one would have to employ the latter two techniques, too, if one wants to arrive at a complete picture of the knowledge contained in a document.

6. MAINTENANCE SERVICE
SPARK PLUGS
Recommended spark plugs:
European and Australian types: BPR6EY-11 (NGK), W20EXR-U11 (ND)
Other types: BP6EY-11 (NGK), W20EX-U11 (ND)
CAUTION: Never use spark plugs with an improper heat range; they will advertisely affect engine performance and durability.
Replace plugs one at a time, so you don't get the wires mixed up.

1. Clean any dirt from around the spark plug base.
2. Disconnect the spark plug wire, then remove and discard the old plug.
3. Thread the new spark plug in by hand to prevent crossthreading.
4. After the plug seats against the cylinder head, tighten 1/2 turn with a spark plug wrench to compress the washer.

CAUTION: The spark plugs must be securely tightened, but not over-tightened. A plug that's too loose can get very hot and possibly damage the engine; one that's too tight could damage the threads in the cylinder head.

① Plug Wrench ② Plug Wrench Handle ③ Plug Cap ④ Tighten



6. PFLEGE UND WARTUNG
ZÜNDKERZEN
Empfohlene Zündkerzen:
Europäische und australische Ausführung: BPR6EY-11 (NGK), W20EXR-U11 (ND)
Andere Ausführung: BP6EY-11 (NGK), W20EX-U11 (ND)
VORSICHT: Niemals Zündkerzen mit einem falschen Wärmewert verwenden; sie beeinträchtigen Motorleistung und Haltbarkeit.
Die Zündkerzen eine nach der anderen auswechseln, damit die Zündkabel nicht durcheinandergebracht werden.

1. Die Zündkerzenbasis von jeglichem Schmutz befreien.
2. Das Zündkabel abziehen, dann die alte Zündkerze herausschrauben und wegwerfen.
3. Die neue Zündkerze von Hand einschrauben, um Gewindeüberschneiden zu vermeiden.
4. Nachdem die Zündkerze am Zylinderkopf aufsitzt, mit einem Zündkerzenschlüssel um 1/2 Umdrehung anziehen, um den Dichtungsring zusammenzupressen.

VORSICHT: Die Zündkerzen müssen fest angezogen, jedoch nicht zu fest eingeschraubt werden.
Eine lose Zündkerze kann sehr heiß werden und möglicherweise den Motor beschädigen. Eine zu fest angezogene Zündkerze kann das Zylinderkopfgewinde beschädigen.

① Zündkerzenschlüssel ② Zündkerzenschlüsselgriff ③ Zündkerzenstecker ④ Anziehen

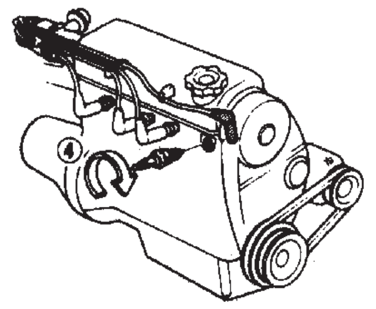


Figure 1: *A multilingual text from our corpus (From: Honda Civic Car Manual)*

- **Contrastive**: The central demand on the knowledge acquired by means of corpus analysis is that—even though the knowledge is obtained from natural language documents—it does not pertain to any specific language. In other words, it is language-independent. One way to secure this is by analysing data from various languages in parallel, because this makes it possible to abstract from the idiosyncrasies of a particular language. With this end in view, we conducted a contrastive analysis of identical sections from multilingual documents. Obviously, only non-linguistic knowledge and the more abstract linguistic levels such as text type and discourse structure are language-independent; syntactic and lexical knowledge naturally has to be language-specific at a certain level of detail.
- **Manual:** It is necessary to extract knowledge from documents that goes well beyond lexical and syntactic resources, such as content and discourse structure. However, this kind of knowledge cannot be derived from analysing text alone, but only from analysing text in the context of a broader understanding of the world. Consider, for instance, the resolution of lexical and syntactic ambiguities: These can in many cases only be resolved by employing nonlinguistic knowledge.
  At present, there exist no automated tools that could perform either of the analysis tasks in a satisfactory manner, thus we have to conduct a manual analysis, however tedious this may be at times.[2] Note, however, that this analysis only has to be performed once as part of the initial knowledge acquisition effort. The knowledge will then be readily available for use by future applications situated in a similar domain.

As noted above, the corpus analysis was performed in terms of *content* and *structure* of the documents. More specifically, this implies investigations into

- content of documents: the domain knowledge underlying the document;
- macrostructure: overall structure of the document type;
- discourse structure: relations holding between parts of the text;
- linguistic realization: lexical choice and syntactic structure.

We now describe the document analysis for each of these levels in turn, and then present the results and their augmentation by means of other sources of information in case they provide substantial supplements.

2.2 Content of Documents

In the initial step of the corpus analysis we examined the documents with respect to what kind of domain knowledge they encode, and what kind of technical knowledge is needed to interpret the domain knowledge correctly. The content analysis aims at an informal characterization of the domain in terms of objects, properties, actions, and relations holding between them. We noted no differences in content between the multilingual variants of a document, as we had expected

[2] Knowledge acquisition techniques for technical texts are a research topic in the knowledge based systems (KBS) community, so powerful tools for, at least, terminology extraction and semantic clustering should be available in near future. Initial results are described in, for instance, [Dagan and Church, 1994] [Justeson and Katz, 1995].

given the present domain, technical documentations, and the target languages involved.[3]

Objects, actions, and properties that are explicitly mentioned in the text provide the kernel of the domain knowledge. The sample text in figure 1 introduces a number of basic domain entities, for instance, *engine, spark plug, dipstick, tighten, remove*, etc. Objects tend to be domain specific, whereas actions are of a more general nature in that their use is not restricted to the domain of car maintenance. Conducting a contrastive analysis of a multilingual corpus helps to define the more abstract level of representation, as described above. We will return to this issue in the discussion of lexicalization, and in section 3.

The initial set of entities collected from the corpus has only a restricted coverage of the domain. It contains only those concepts that actually occur in the documents under consideration. Hence, there are obvious 'gaps' where related concepts are missing, where an enumeration is not complete, where properties are not specified, etc. In order to arrive at a more complete picture of the domain, we took additional knowledge sources into account: Experts, product models from industry, research literature, textbooks on mechanical engineering, and the like.

These steps resulted in a collection of 'clusters' of domain entities, for instance, a cluster of technical objects or one of connection types, with the clusters not being related to each other. In a further step, these have to be coordinated to form a coherent whole. Relations and concepts that act as links between fragments can be defined with the help of additional knowledge sources. We will return to this matter in section 3.

2.3 Macrostructure

The *macrostructure* characterizes the global semantic and pragmatic structure of a particular text type. In other words, the macrostructure describes the characteristic content of a text type in terms of typical parts (*structural elements*), and the content that is usually conveyed by these parts.

We identified the macrostructure for the given text type, *maintenance instructions*, by comparing several multilingual instances of this text type with respect to recurring structures and content, and with respect to the function of the recurring parts. There appeared to be no substantial differences between the language-specific variants. The analysis provided an initial specification of the macrostructure, which we compared in a subsequent step to findings from literature on content and structure of technical documentation and to constraints imposed by DIN norms. The resulting macrostructure for maintenance instructions can be captured by the following schema:

- *name* of maintenance instruction
- *location* of objects to be repaired/maintained
- possible *replacement* parts/substances
- *preconditions* that have to be satisfied before activity proceeds
- step-by-step description of *activity*
- further advisory information or *warnings*

[3] However, once we turn to other domains and languages (and thus cultures), which are less closely related, we might encounter differences in content due to cultural differences, too.

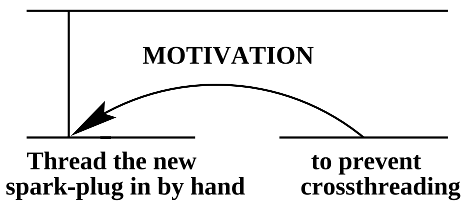

MOTIVATION
Thread the new spark-plug in by hand
to prevent crossthreading

Figure 2: *The* M*OTIVATION relation: Analysis of sample sentence.*

The analysis further reveals constraints as to what kind of content may occur within a particular element. For instance, the activity element can contain either of the three major categories of checking a part/substance, adding a part/substance or replacing a part/substance [Rösner and Stede, 1992]. Either of these presents a typical content specification of the activity element. Note that the macrostructure as presented above merely describes a *potential*, since it contains optional and obligatory elements and is underspecified with respect to the ordering of elements. The obligatory elements and their order of occurrence define the document type. In our example, obligatory elements are name, replacement and activity. Thus, any document instance that comprises the obligatory parts in the specified order and any number of optional parts will be classified as belonging to the type maintenance instruction.

2.4 Discourse Structure

While the macrostructure defines the global content structure of a document, the discourse structure operates on a more local level (the *microstructure*): It specifies the structural and functional representation of a text and of parts of a text down to sentence level. As such, it describes how *text spans* (parts of the text) are related to each other semantically and rhetorically by means of *discourse relations* to form a coherent text.

We use *Rhetorical Structure Theory* (RST) [Mann and Thompson, 1987] and its set of rhetorical relations to describe how the contents of basic semantic elements are combined to form a coherent discourse and what the specific function is that an element takes within the relation. In a nutshell, RST posits that discourse structure can be represented as a connected tree, where adjacent text spans are linked by one of approximately 20 rhetorical relations. Most relations link a *nucleus* to a *satellite*: The former is the central element, indispensable for developing the argumentation in the text, whereas the satellite has a more supportive role, presenting additional information. Consider, for instance, the MOTIVATION relation, which holds between a text span presenting some kind of action the writer intends the reader to perform (the nucleus), and a text span presenting information that increases the reader's desire to perform the action (the satellite). The satellite thus acts as a support for the information depicted in the nucleus. An instance of the MOTIVATION relation is *Thread in the new spark plug by hand to prevent crossthreading*, where *to prevent crossthreading*

ALTERNATIVE	MOTIVATION
AND	PRECONDITION
BACKGROUND	SEQUENCE
CONTRAST	STEP–SEQUENCE
CONDITION	UNTIL
ELABORATION	VOLITIONAL–RESULT

Figure 3: *RST relations in manuals*

acts as support for instructing the reader to perform the action described in the first clause in the specified manner (*thread in ... by hand*). The corresponding analysis is given in figure 2. Note that some relations, such as SEQUENCE and CONTRAST, do not have any satellite, but two or more nuclei.

An analysis of texts from our corpus revealed that only a subset of RST-relations is employed in maintenance instructions [Rösner and Stede, 1992]. These are mainly of the *subject-matter* type, that is discourse relations that reflect semantic relations holding 'in the world' such as causality and temporality. These relations are used to make the reader recognize that a particular semantic relation holds between related text spans. However, the set of relations presented in the initial work on RST turned out to be insufficient: In order to cover all the phenomena found in technical manuals, a number of new discourse relations have been introduced by [Rösner and Stede, 1992]. These are ALTERNATIVE, AND, PRECONDITION, STEP-SEQUENCE, and UNTIL. Figure 3 presents the set of rhetorical relations most frequently encountered in the corpus. Note that only BACKGROUND and MOTIVATION are *not* of the subject-matter type.

RST turned out to be a useful formalism in a *multilingual* setting: For almost every section we investigated, the RST-trees for different language versions were almost identical; we observed no substantial difference between RST-trees for English, German and French texts. In brief, the contrastive analysis of corresponding sections reveals discourse structure as a level of representation that captures the commonalities of English, German and French text.

So far we have discussed *what* relations occur in the documents of our corpus. In a further step of analysis, we investigated *why* a particular relation holds at a particular point in the document. The following method was applied: We produced paraphrases, i.e. alternative linguistic realizations, for a given text span. Then, we compared the different alternatives with respect to the effect they have on the reader. Knowing about the reasons underlying the choice of a particular discourse relation is crucial when it comes to producing documents automatically from a knowledge base (see section 4).

2.5 Linguistic Realization

The final step consisted of analysing the *linguistic realization* of content structures and functions in the texts of our corpus. Linguistic differences between English, German and French versions of the texts can be classified into lexical and syntactic ones (see [Grote *et al.*, 1993] for an in-depth contrastive study).

2.5.1 Lexical Choice

Lexical differences hold for the realization of noun phrases: For instance, German features a proliferation of compound nouns that are expressed by entire phrases in English (*Kühlervorderseite* vs. *front of the radiator*). More interesting lexical differences relate to the use of verbs in both languages. Quite often in the corpus, a fairly general English verb coincides with more specific German or French counterparts. To illustrate this point, consider the English and German realization of the action of `dissolving a connection`. In English, it is in most cases realized by *remove* regardless of the object involved, whereas the German version displays several different realizations [Rösner and Stede, 1992]:

Example 1.

(a)	*remove the dipstick*	*den Tauchmeßstab herausziehen*
(b)	*remove the spark plug*	*die Zündkerze herausschrauben*
(c)	*remove the oil filler cap*	*den Öleinfülldeckel abnehmen*

The German verbs convey additional information about the spatial characteristics of the participating objects and the nature of the connection relation. Another striking difference between English and German is that verbs may or may not incorporate a portion of meaning that may otherwise be expressed by an adverb, for instance, the English *re-install*, which incorporates the restitutive character of the action as opposed to the German variant *wieder anbringen* (see also [Grote *et al.*, 1993]).

The results of the contrastive lexical analysis regarding *granularity* and *incorporation* have significant consequences for the design of the domain knowledge base. This will be further discussed in section 3.2.

2.5.2 Syntactic Structure

The contrastive analysis of syntactic structures was performed in the following way: When the syntactic structures between corresponding text segments were incongruent, we first took the structure of the expression found in one language and tried to gloss it in the other one, and vice versa. If both resulted in an acceptable rephrasing and we could detect no obvious reasons for the choice of one construction over the other, we took them to be mere variations [Grote *et al.*, 1993]. All cases where glossing did not lead to an acceptable sentence underwent further analysis in order to capture the reasons for choosing a particular construction over another.

The most striking difference relates to the use of complex participles or prepositional phrases in German, where in English the same content has to be expressed by a subordinate clause. Example (2) illustrates the use of a prepositional phrase in German as opposed to a subordinate clause in English:

Example 2.
(a) *Den Kühlmittelstand im Reservetank bei normaler Betriebstemperatur des Motors kontrollieren.*
(b) *Check the coolant level in the reserve tank when the engine is at normal operating temperature.*

Finally, we find certain phenomena pertaining especially to our text type. In maintenance instructions, the reader is addressed with the imperative mood in English, whereas in the German texts we find a—more polite—infinitive construction:

Example 3.
(a) *Read the dipstick level.*
(b) *Den Ölstand ablesen.*

The analysis results in a specification of the range of phenomena to be covered by the linguistic knowledge bases. Obviously, the phenomena are not limited to the ones described in this section, these are only the most important ones.

Linguistic resources, such as lexicon and grammar, that are to be employed in the automatic production of multilingual technical documents thus need to have a broad coverage, be multilingual, and support flexible lexicalization.

3 Knowledge representation

Once the knowledge contained in documents of a specific domain has been accumulated, the question of how to represent and formalize that knowledge arises. To manage this task, we can employ insights gained in the acquisition phase. In particular, the knowledge acquisition phase supplies:

– a collection of objects and actions of the domain together with relations holding between objects;
– a description of linguistic and syntactic phenomena to be covered;
– an informal description of the text type under consideration.

The knowledge acquisition phase contributes in another significant way to managing the representation task: Experiences from that phase help to judge the relevance of different aspects of the domain under consideration, and thus facilitate decisions on which entities to represent and which ones to omit. In combination with the requirements imposed by the application in mind, this provides a good starting point for choosing an adequate knowledge representation paradigm and an implementation system. In section 3.1, we discuss this decision process, while section 3.2 sketches the formalization of the knowledge within the chosen paradigm.

3.1 Requirements Analysis and Representation Decisions

In addition to the requirements imposed by the domain entities to be represented, we can postulate two further demands on the domain knowledge. These are motivated by the intended usage of the knowledge sources and by potential applications:

– The knowledge sources should be usable for the generation of multilingual technical documents.
– The represented knowledge should be in a state that permits sharing among different applications, and reuse for purposes other than multilingual generation.

The first requirement accounts for the fact that the use of generation techniques in the production of multi-lingual documents has substantial advantages over the traditional approach of creating monolingual documentations and then translating them into other languages.[4] On the other hand, generation techniques require a high initial effort in knowledge acquisition and knowledge representation. Even though this is partially justified by the benefits of that approach, the initial effort constitutes the main obstacle on the way to real applications.

These considerations suggest the need for a better cost-benefits ratio, which in turn motivates the second requirement: A significant improvement of knowledge capitalization can be achieved by sharing and reusing knowledge sources developed within a particular application. Here, *knowledge capitalization* names the surplus between the costs of producing and maintaining knowledge bases on the one hand, and the benefits from having a knowledge base at one's disposal on the other hand. Other definitions focus more on methods to maximize the benefits. For example, [Simon, 1996] defines knowledge capitalisation as "the process which allows to reuse, in a relevant way, the knowledge of a given domain, previously stored and modelled, in order to perform new tasks".

One way to realize knowledge sharing and reuse is by means of ontologies. According to [Uschold and Gruninger, 1996], an *ontology* is "an explicit account or representation of (some part of) a conceptualization."; and a *conceptualization* is regarded as a "world view with respect to a given domain, which is often conceived as a set of concepts (e.g. entities, attributes, processes), their definitions and their inter-relationships".

The two demands on the domain representation discussed so far have substantial consequences for the design of the knowledge bases as such. In particular, these are:

- Reusable domain knowledge and knowledge concerning an instance of a product have to be clearly separated. This is commonly achieved by introducing abstraction barriers. We distinguish two levels of representation: an *abstract technical model*, which can be exploited for other purposes than the intended, and a *product specific model*.
- Concepts and the relationships holding between concepts of the abstract technical model have to be modelled within an ontology.
- Domain knowledge has to be language-neutral, in other words, it has to be modelled in such a way that it is not biased towards any particular natural language. This is an imperative when generating text in different languages from the same formal content representation.
- The representation has to permit an easy integration of linguistic and domain knowledge.

We identified *description logic* as a knowledge representation paradigm that makes it possible to build knowledge bases in line with the design decisions we have taken [Liebig and Rösner, 1996a]. This paradigm allows us to formalize abstract descriptions by supporting hierarchical concepts and relations including multiple inheritance. Concepts can be defined in such a way that they reflect important properties of and differences between concepts, which is a characteristic feature of ontologies. In the area of natural language generation, the

[4] Section 5 provides a detailed discussion of the advantages of multilingual document generation

main inference capability of description logic, automatic classification, can support the tasks of content determination, text planning and surface realization [Reiter and Mellish, 1992].

We have chosen Loom [LOOM, 1991], a LISP-based descendent of the KL-ONE family [Brachman and Schmolze, 1985], as the implementation system. The decision was motivated by the empirical analysis of six terminological representation systems presented in [Heinson *et al.*, 1994], which identified Loom as the most expressive and fastest system.

Furthermore, Loom is a hybrid formalism that, in addition to description logics, supports other paradigms like forward and backward chaining rules and object orientation. Those are of great help when formalizing the more procedural part of technical knowledge [Liebig and Rösner, 1996b]. Finally, the decision was influenced by the layout of other modules of the system to be: The sentence generator we are using (see section 4) also employs Loom knowledge bases; staying with that representation system facilitates the cooperation between linguistic and domain knowledge.

3.2 Knowledge Formalization

In general, we are concerned with the formalization of two types of knowledge: Abstract technical and product specific knowledge on the one hand, and linguistic knowledge on the other hand. In this section, we will restrict the discussion to the former type and its formalization within the framework of description logic. Linguistic sources such as grammar and lexis are more adequately represented using representation formalisms that are explicitly designed for linguistic purposes. Their representation will be subject of section 4.2. An exception to this is the *Upper Model* [Bateman *et al.*, 1994], a linguistically motivated ontology which serves as an interface between linguistic and non-linguistic knowledge, and which is formalized in the framework of description logic (see section 3.2.3).

3.2.1 Technical and Product Model

The analysis of technical documentations provides objects, actions, properties, and relations between them. In order to arrive at a formal conceptualization of these entities, we took the following steps in the design of the knowledge base:[5]

- grouping related entities (objects and relations), determining their commonalities, and describing them by means of properties;
- determining abstract descriptions which cover a wide range of phenomena;
- ensuring that conceptualizations made in a particular language are also reflected in the formal representation;
- realizing a modular description on different levels.

To illustrate these points, we present examples taken from the current domain for each of the design decisions. In this way we describe the influence of these design decisions on the actual layout of the knowledge bases.

Entity grouping and the description of common properties: A number of entities in the technical domain share common properties. They are

[5] Design decisions for knowledge bases are discussed in a number of articles, see for instance [Brachman *et al.*, 1991].

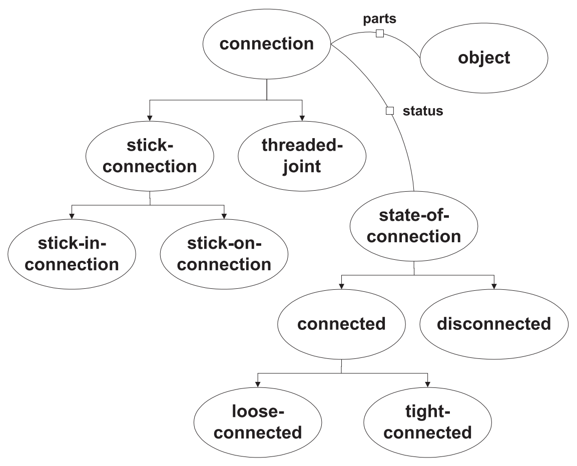


Figure 4: *The conceptualization of connections*

grouped together, and a more abstract superconcept is introduced that denotes the set union of these entities. Take, for instance, objects occurring in the document on changing spark plugs (given in the sample text in figure 1) and in similar texts that instruct the reader on how to change oil or how to check the coolant level. An analysis of the objects led to the introduction of several abstract concepts such as `technical-item` (spark plug, spark plug wire), `technical-instrument` (spark plug wrench, screw driver), `measuring-instrument` (dipstick, coolant thermometer), `container` (oil tank, coolant tank), `machine` (engine), `vehicle` (car), to name a few. The common properties of these abstract concepts in turn are covered by the concept `technical-object`.

Abstract description of phenomena: There are several starting points for a more abstract description of domain specific concepts and relations. An important abstraction is captured by *part-whole* relations. In our domain they mainly hold between technical objects. A spark plug, for instance, is part of an engine, which is in turn part of a car. The part-whole relation has been studied in previous work (e.g. [Winston *et al.*, 1987], [Artale *et al.*, 1996]), where several part-whole relations are distinguished and where problems regarding the transitivity of this relation are pointed out. In analogy to the inheritance of properties from superconcepts to their subconcepts, properties of the whole can be inherited by their parts and vice versa (vertical relationship). Furthermore, properties of parts can affect properties of other parts (horizontal relationship). Hence, our conceptualization has to reflect these different part-whole relations and the transitivity of the different part-whole relations. For a detailed discussion of the part-whole relation and an abstract conceptualization within the technical domain see [Liebig and Rösner, 1996b] and [Liebig and Rösner, 1996a].

Lexical influences on the conceptualization: The actual definition of a taxonomy for a certain area of the domain knowledge strongly interacts with results from the lexical analysis. In section 2.5.1, we introduced different lexical options for realizing one and the same concept (the `disconnect-resolvable-connection` action), which are available in English and German respectively. These different lexicalizations reflect different restrictions on the type of connection concerned and on the object involved in the action. For instance, lexicalizations may differ with respect to the level of granularity in which they reflect differences between connection types. To give an example: The analysis in section 2.5.1 reveals that the German verbalization of a `disconnect-resolvable-connection` action is sensitive to the connection type: *Herausziehen* occurs when objects are connected with a `stick-in-connection`, whereas *abziehen* is chosen if objects are connected by means of a `stick-on-connection`. English does not mirror this discrimination in the choice of lexemes. In a nutshell, a representation of objects, actions and relations, which can support the appropriate verbalization of the `disconnect-resolvable-connection` action in English *and* German, has to reflect the different connection types and their distinguishing features. Examples are thread connections (`threaded-joint`) and connections that are established by sticking some object into or onto another object (`stick-connection`).

Figure 4 shows a detail from the taxonomy for connections.[6] The taxonomy also reflects the state of a connection (`state-of-connection`), such as whether the connection holds or does not hold. As states play an important role for the conceptualization of actions, we will return to states in the next subsection.

Modular description: Since all the phenomena described so far (part-whole relation, connections) provide abstract means for formalizing many technical objects, we place them in the abstract technical model. In this view, the abstract technical model contains a collection of related ontologies. As we emphasized in section 3.1, a modular layout of the knowledge bases facilitates the reuse of parts of the knowledge base. This increases the benefits of the initial knowledge acquisition effort.

The conceptualization of the abstract technical model draws heavily on a supplementing analysis of additional knowledge sources. The abstract technical model is the very place where the additional knowledge that is employed by the reader when interpreting a given technical documentation is represented explicitly. As already mentioned in section 2, additional knowledge sources such as textbooks and encyclopedias are used to acquire the information implicitly contained in natural language texts. Textbooks, especially those on mechanical engineering, often contain taxonomies of domain entities as well as differentiation criteria (see for instance [Roth, 1982]), which can be exploited in setting up the abstract technical model.

3.2.2 Plan Structures

Instructive texts can be well formalized by means of *plans*, which represent complex actions [Fikes and Nilsson, 1971]. Complex actions can be decomposed into

[6] We use the following notational conventions: Concepts are denoted by ellipses, whereas instances are denoted by rectangles. The `is-a` relation between concepts and the `is-instance-of` relation between an instance and a concept is indicated by arrows, while other relations between concepts are indicated by a line, labelled with the relation name. If several sister concepts represent a disjoint covering of a superconcept, such as the states `connected` and `disconnected` in figure 4, they are connected to the superconcept by branching arrows.

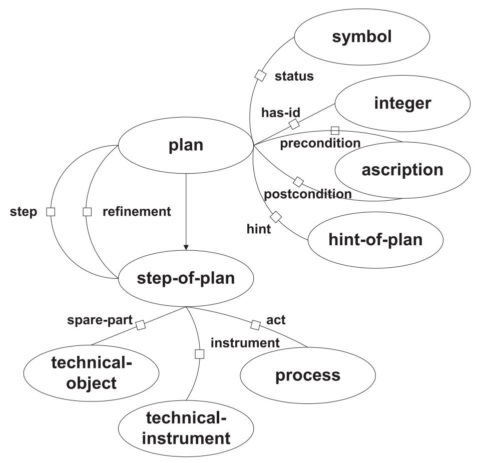


Figure 5: *The conceptualization of plans*

replace-spark-plug_plan	plan
status	command
step	clean-any-dirt_step
	disconnect-spark-plug-wire_step
	remove-and-discard-old-spark-plug_step
	thread-in-new-spark-plug_step
	tighten-new-spark-plug_step
hint	tighten-but-prevent-overtighten_hint
	caution-improper-heat-range_hint

Figure 6: *The representation of the instructive parts in the sample text (figure 1)*

a sequence of actions. Figure 5 presents an abridged version of our formalization of actions and plans. For a given plan, the relation `step`, as depicted in figure 5, contains a sequence of complex or elementary actions. An action is elementary if the `act` relation is filled with an action instance, and complex if the `refinement` relation is filled by another plan instance. Each action and plan is associated with a set of conditions necessary for the action or plan to be applied, the *preconditions*, and a set of facts that hold after the action or plan has been performed, the *postconditions*. Within a *situation* we represent an incomplete description of the state of objects in the world. When an action is performed, the state of the objects that participate in this action may be modified. Thus, actions transform a given situation into a new one.

According to their different pre- and postconditions one can differentiate between three types of `connective-actions` in our domain: `resolve-connection-action`, `modify-connective-action` and `create-connective-action`. In figure 8 the conceptualization of connective actions is presented. Each of these actions alters the state of a connection within a given situation. States of connections are represented by the subconcepts of `state-of-connection` in figure 4. A `create-connective-action`, for instance, establishes a connection between two unconnected objects and thus alters the status of the connection from `disconnected` to `loose-connected`, whereas after the application of a `resolve-connective-action`, the connection state changes in the opposite direction.

In addition to the pre- and postconditions, warnings on potential errors and further advisory information such as information on trouble-shooting and on possible dangers that may occur when performing this action may be attached to an action. These elements correspond to the structural elements *preconditions*, *activity* and *warnings* in the macrostructure of maintenance instructions presented in section 2.3. To illustrate this point, consider the sample text in figure 1: The instructive parts of this text (how to replace spark plugs) can be modelled as a plan containing four steps and two hints. The corresponding plan representation is given in figure 6.[7]

3.2.3 Upper Modelling

We employ the *Upper Model* (as described in [Bateman *et al.*, 1994]) to bridge the gap between *non-linguistic* knowledge encoded in the product specific model and the abstract technical model, and *linguistic knowledge*. The upper model is a linguistically motivated ontology, whose classifications of entities reflect semantic distinctions in at least one of the target languages. As such, it offers a way of defining objects, actions and properties by supplying a hierarchy of general concepts, which are domain and task independent.

There are several approaches to linking domain knowledge to the upper model, in other words, to relating non-linguistic to linguistic knowledge. In the present application, we realized the linking by subordinating entities of the abstract technical model to upper model concepts. This has been mainly motivated by design decisions of the current implementation, for another more flexible approach, which is motivated by work in lexical semantics, see [Stede, 1996].

To illustrate our approach, consider the taxonomy in figure 7. It exemplifies the connection between the upper model, the abstract technical model and the product specific model for particular concepts. In this example, the concept `connective-action`, as a part of the abstract technical model, is related to the upper model by subordinating it to the upper model concepts `dispositive-material-action`, `material-process`, `process` and `thing`.

As the concepts of the abstract technical model and the product specific model are subconcepts of the upper model, they inherit the relations defined for their superconcepts in the upper model. Consider the concept `disconnect-resolvable-connection` in figure 7, which is a subconcept of the concepts `pro-`

[7] We use the following notational conventions to describe instances of the knowledge base: The heading of the table contains the name of the instance on the left side and the name of its direct superconcepts on the right side. In the body of the table for every relation the relation name is given on the left side and the range of relation under consideration on the right side.

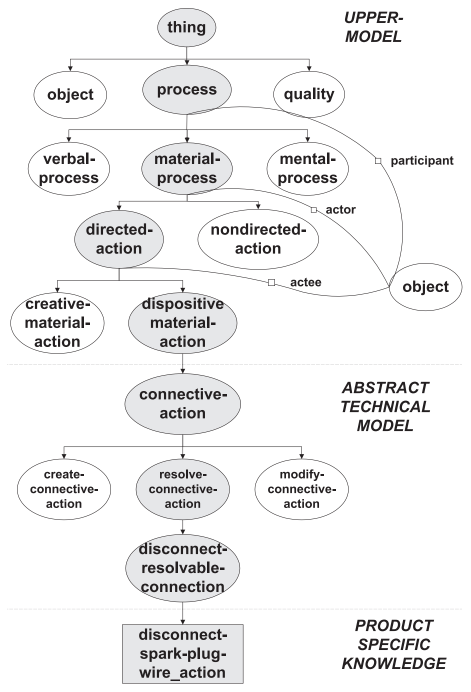


Figure 7: *The association of connective action with the upper model*

cess, `material-process` and `directed-action`. The concept `process` introduces the `participant` relation to those objects that partake in a process. This relation is further specialized by the subrelations `actor` and `actee`, which are defined by `material-process` and `directed-action`. Both relations describe the role an object takes in a given process.

Recall the instructive parts of the sample text, which describe how to change the spark plugs and their formal representation in figure 6. The second step `disconnect-spark-plug-wire_step` contains a reference (within the `act` relation) to the elementary action `disconnect-spark-plug-wire_action`. The representation of the plan step and the action is given in figure 8. Within the action instance `disconnect-spark-plug-wire_action`, fillers for the relations defined by upper model concepts are given. Figure 7 shows how this action instance is related to concepts within the upper model and the abstract technical model.

4 Expressing the Knowledge: The TechDoc Generator

We have already stressed the importance of documents within the product life cycle, and in particular noted the importance of automating access to the knowledge contained in these documents. A prerequisite for that is the formal repre-

disconnect-spark-plug-wire_step	step-of-plan
act	disconnect-spark-plug-wire_action

disconnect-spark-plug-wire_action	disconnect-resolvable-connection
actor	reader_instance
actee	spark-plug-wire_instance

Figure 8: *The representation of a single step within the plan in figure 6.*

sentation of the knowledge of a particular domain, especially that knowledge that is expressed—be it explicitly or implicitly—in documents situated in the domain under consideration. An initial formalization of technical knowledge, more specifically knowledge expressed in car manuals, has been attempted in section 3. The next step is to validate the resulting knowledge bases with respect to a particular practical application. The application we have opted for supports our view of documents as a particular view on knowledge: the automatic production of documents from a common underlying knowledge base. The quality of the documents will give valuable feedback to the representation phase.

To meet this goal, we have developed the TechDoc system, a multilingual text generation system that automatically produces a particular type of document—multilingual technical manuals in English, German and French—from a common knowledge representation. The generated documents reflect a particular view on the knowledge: Given some parameters such as level of expertise of the reader, language and text type, the TechDoc generator creates different documents from the same knowledge pool. In this way, the TechDoc system can be regarded as a proof of concept, as it demonstrates the feasibility of our understanding of the interaction between knowledge bases and natural language documents.

In the remainder of this section, we describe the TechDoc prototype and the state of implementation in detail, and then present a sample run of the generation process as it has been implemented so far.

4.1 System Architecture

The overall architecture of the TechDoc text generation system is given in figure 9. A detailed description of the TechDoc text generation system is given in [Rösner and Stede, 1994] and [Rösner, 1994]. The system components fall into three major groups:

- **knowledge sources**:
  - knowledge bases of different types and abstraction levels,
  - linguistic resources like multilingual grammars and lexica,
- **processing modules** that transform one level of representation (data structures) into another,

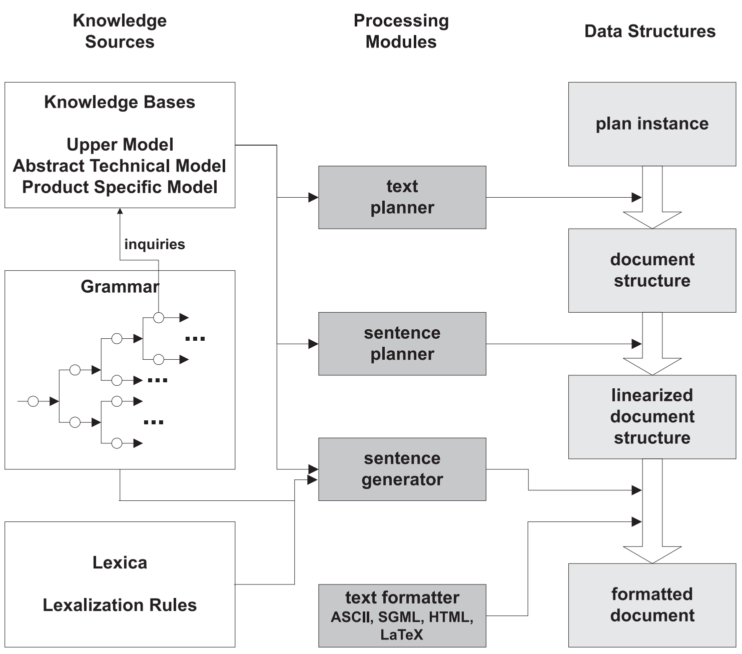


Figure 9: *The TechDoc System Architecture*

– **interfaces** that support the user-system interaction.

Not all processing modules are worked out in detail and hence are not fully integrated. Some aspects need further investigation and verification, and are subject to current research. We will go into some of these issues in section 4.3.

The TechDoc system is based on knowledge bases encoding product specific knowledge and more abstract technical knowledge, schematic text structure and task representation as described in section 3. These knowledge sources are implemented in Loom. TechDoc takes as input an instance of a plan, i.e. a formal representation of a specific task, together with other elements from the macrostructure (document structure, see section 2.3), such as *name, location, warning*, etc. Starting from this representation, a language-independent discourse structure is generated. This step includes selecting what to express, deciding on the textual representation of selected content structures and imposing a partial ordering on them. The basic elements at this level of representation are (complex) RST relations (see section 2.4). In a subsequent step, the discourse structure is broken down into clause sequences. This transformation must determine:

– clause boundaries,
– syntactic structure of single clauses,

- theme of clauses,
- appropriate referential expressions (pronouns, definite or indefinite descriptions) within clauses, and
- linguistic realizations (e.g. function word or phrases) that signal rhetorical relations.

In [Rösner and Stede, 1992] a pattern-based algorithm is proposed that incorporates linguistic and semantic constraints and preferences to transform a plan structure into a discourse representation. A more complex treatment of the text planning tasks are currently investigated within the research group.

The resulting sentence plans, specified in the sentence planning language (SPL) [Kasper, 1989], are successively handed over to the sentence generation module. As a front-end generator, we employ the PENMAN [Mann, 1983] sentence generator, which has been enhanced by German and French grammars and separate morphology modules. PENMAN takes an SPL expression as input term from which it generates sentences in either English, German or French. In a final step, the document structure is exploited for an automatic formatting of the output text. The formatting module take the output medium (screen or printed) into account, and realizes generated text in the selected format (e.g. ASCII, SGML, HTML, LaTeX).

Our approach to treat documents as specific views on the underlying knowledge bases is particularly well demonstrated by the hypertext versions of the generated text. Elements in the text are linked to the underlying knowledge base instances that gave rise to their existence. The user can use the text as a query interface to the knowledge sources, for instance, s/he can prompt for additional information about a technical object or action by simply clicking on the respective noun phrase or verbal structure. In this sense, documents are 'active', and can change the view on the knowledge base in response to user interaction.

4.2 Linguistic Resources

The sentence generator requires linguistic resources—grammar, lexica—and the upper model as the interface between non-linguistic and linguistic knowledge to produce well-formed output. The linguistic resources have to meet the requirements identified in section 2.5. In the following, we give a short account of the linguistic resources as employed by the PENMAN sentence generator.

4.2.1 Syntactic Knowledge

Syntactic and some lexical knowledge is encoded in a *systemic-functional grammar* [Halliday, 1985], [Matthiessen and Bateman, 1991]. Such a grammar is organized around *systems* and *choosers*. Each system represents a minimal semantic difference, and a chooser represents the criteria for choosing between the alternatives (or *features*) a system offers. Choosers are organized as decision trees, which post inquiries to knowledge bases and other external information sources at each decision point. Thus, choices in the system network (grammar) are ultimately constrained by the context in which an utterance is produced. While traversing the grammar, a set of features is accumulated, which gives

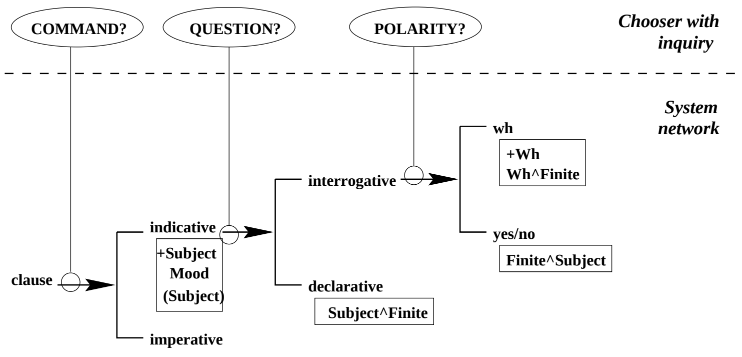


Figure 10: *The system network realizing* MOOD *[Matthiessen and Bateman, 1991].*

rise to a specific linguistic realization. PENMAN builds natural language sentences from sentence plans as a side-effect of traversing the system network of a systemic-functional grammar.

Figure 10 shows part of the system network and the corresponding choosers. The network can be interpreted as follows: It describes some of the options available in the realization of mood, i.e. whether a clause is realized as declarative, interrogative or imperative. These are options in the network (`declarative`, `interrogative`, `imperative`). A choice among these features has to be made for every clause, therefore the feature `clause` acts as entry condition to the network. Each feature has associated *realization statements*, which posit constraints on the resulting linguistic realization; for instance, by selecting the branch `indicative`, a subject is introduced to the clause (`+Subject`). Realization statements are given in boxes below the feature they are associated with. + triggers the introduction of an element, and ∧ indicates the order of elements. As already mentioned, choices among alternative features are made by choosers that query the knowledge bases. For instance, the choice between `imperative` and `indicative` is performed by a chooser that inquires whether the utterance is intended as a command or not. In case the answer is 'command', the imperative feature is chosen and the MOOD can be realized, in case the answer is 'no command', `indicative` is selected and further choices have to be made.

4.2.2 Lexical Knowledge

Lexical knowledge can be of two kinds: On the one hand, we observe a one-to-one correspondence between concepts and lexemes. In this case, we simply introduce a lexical rule stating this correspondence. In our domain, for instance, the concept `dipstick` is always mapped on the German lexeme *Tauchmeßstab* and the English *dipstick*. More frequently, we encounter cases as described in section 2.5, where German and English differ with respect to granularity, and hence, one and the same concept is realized differently in different contexts. This requires a more complex treatment. We deal with this phenomenon by

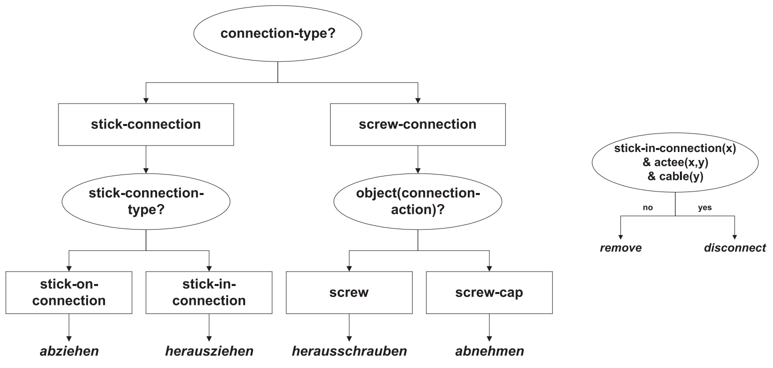


Figure 11: *Lexical rules for German and English disconnection concepts*

introducing *lexicalization rules* that invoke filler restrictions for the roles of the concept to be lexicalized.

Figure 11 presents decision trees that produce adequate German and English lexemes for the action of disconnecting a dissolvable connection. A detailed discussion of these lexicalization rules follows in the next section, where we present a sample run of the TechDoc generator.

4.3 Example

This section illustrates the automatic production of documents by means of a detailed example, the spark plug text from figure 1. For reasons of comprehensibility and space, we will restrict the discussion to a subpart of the text, the second half of the text starting with the enumeration. These paragraphs realize the structural elements *activity* and *warnings*.

The activity element comprises a step-by-step description of an activity eventually to be performed by the user, whereas the element warning realizes further advisory information or warnings. Both elements can easily be represented as partial plan structures (see section 3.2.2), in this case by the plan instance `replace-spark-plug_plan` ; the corresponding representation on the knowledge level has already been introduced in figure 6. Here, the activity element is expressed by the `step` relation, the warning element matches the content of the `hint` relation. The text generation system takes plan instances as input; see figure 9 for an outline of the generation process.

In a first step, the plan structure is transformed into a discourse structure, with the plan elements acting as leafs in the discourse tree. Figure 12 shows the discourse structure that has been built automatically from the `replace-spark-plug_plan`. In order to facilitate the coreference between plan and discourse elements, we list the plan elements again and assign numbers to them:

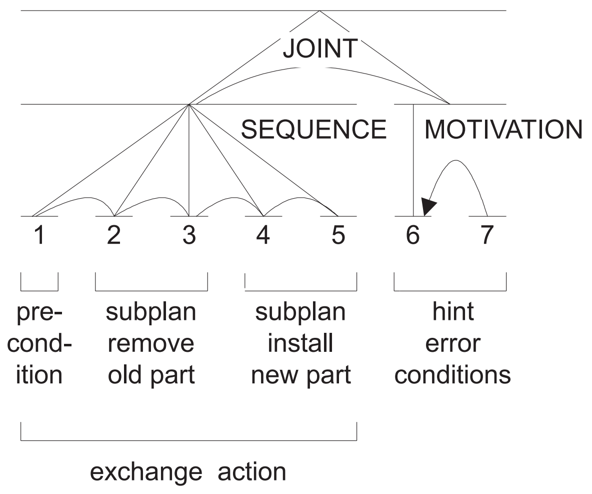


Figure 12: *The discourse structure built from the plan structure in figure 6.*

1 clean-any-dirt_step
2 disconnect-spark-plug-wire_step
3 remove-and-discard-old-spark-plug_step
4 thread-in-new-spark-plug_step
5 tighten-new-spark-plug_step
6 tighten-but-prevent-overtighten_hint
7 caution-improper-heat-range_hint

These numbers are used in the discourse structure in figure 12 to refer to a plan element. Plan steps 1 to 5 enter into a SEQUENCE relation. Hints 6 and 7 are governed by a MOTIVATION relation. Both subtrees are combined by a JOINT relation. We noted earlier that the plan structure as presented in figure 6 has not been fully expanded: The plan steps can be further refined by subplans and actions, containing information on their pre- and postconditions, giving rise to a more fine-grained discourse structure. At present, the transformation of plan structures to discourse structures is performed in a straightforward manner without accounting for all the interdependencies between the different tasks of the transformation step as listed in section 4.1.

The discourse representation, that is, plan elements from the plan structure (which are the leaf nodes in the tree) and the discourse relations holding between them, are then passed to the sentence planner, which is responsible for linearising the tree and building SPLs. A sample SPL, constructed from the `disconnect-spark-plug-wire_action` instance given in figure 8), is shown in figure 13.

An SPL term has the form `(Variable / Type Attribute*)`, where `Type` refers to a concept in the knowledge base. The sample SPL term in figure 13 invokes the concepts `disconnect-resolvable-connection`, `person`, and `spark-`

```
(drc / disconnect-resolvable-connection
  :actor (reader_instance / person)
  :actee (spark-plug-wire_instance / spark-plug-wire)
  :speechact command)
```

Figure 13: *An SPL expression built from the instance in figure 8.*

`plug-wire`. The instance `reader_instance` denotes the addressee of the utterance. Attributes describe the functional roles of specifiers, modifiers or complements of the head element. In our example, the head `disconnect-resolvable-connection`, which is a subconcept of `process` as shown in figure 7, has the attributes `actee` and `actor`. In addition to the propositional knowledge encoded by domain concepts and relations, the SPL from figure 13 contains information on the speech act (`:speechact command`). The filler of the speech act role has been derived from the filler `command` of the `status` relation within the sample plan given in figure 6. In case the status is instantiated with `description`, the sentence plan for `disconnect-spark-plug-wire_action` would contain the line `:speechact assertion`.

In the sentence planning module, SPLs are built for all leaf elements of the discourse structure. These are then successively passed to the sentence generator for syntactic and lexical realization.

The SPL in figure 13 triggers the generation of *Disconnect the spark plug wire* and *Die Zündkerze abziehen*. The corresponding assertional variant is *The reader disconnects the spark-plug-wire* and *Der Leser zieht die Zündkerze ab*. These different realizations are due to different instantiations of the `speech-act` role. The filler `command` will eventually give rise to the imperative mood, as described in the discussion of the MOOD system from figure 10. In case the speech act is instantiated with `assertion`, the features `indicative` and `declarative` from the MOOD network are selected by the relevant choosers, thus yielding a declarative sentence.

Let us now turn to the lexicalization: We have already mentioned in section 2.5 that German and English differ regarding the verbalization of processes. Hence, we introduce a set of lexicalization rules (see also figure 11). These rules help to transform the language-independent SPL expressions into well-formed verbalizations in different languages.

Consider the SPL given in figure 13, which contains the `disconnect-resolvable-connection` action. The lexical realization of this action in German depends on the connection type referred to. Using the information on connection types given in figure 4 in the previous section, the lexical realization depends on whether a `screw-connection` or a `stick-connection` should be resolved. For actions of the type `stick-connection` further decisions depend on whether the connected parts stick *in* or *on* another, and for actions of the type `screw-connection` on the object of the disconnecting action. When the object is a subconcept of the concept `screw`, one would use the term *herausschrauben (screw off)*, but when it is a subconcept of the concept `cap` (e.g. `oil-filler-cap`), one would prefer the term *abnehmen (take off)*. In English, the `disconnect-resolv-`

Spark plugs

1. Clean any dirt off the spark plug base.
2. Disconnect the spark plug wire, remove the old plug, and discard it.
3. Thread in the new spark plug by hand, in order to prevent a cross-threading.
4. After the plug seats against the cylinder head, tighten the plug securely with a spark plug wrench, in order to compress the washer.

CAUTION: The spark plugs must be tightened securely, but they shouldn't be overtightened. A plug that is too loose can be very hot, and it can possibly damage the engine, but a too tight plug could damage the threads in the cylinder head.

Zündkerzen

1. Jeglichen Schmutz von der Zündkerzenbasis entfernen.
2. Das Zündkabel abziehen, dann die alte Zündkerze herausschrauben und sie wegwerfen.
3. Die neue Zündkerze von Hand einschrauben, um Gewindeüberschneiden zu vermeiden.
4. Nachdem die Zündkerze dem Zylinderkopf aufsitzt, die Zündkerze fest mit einem Zündkerzenschlüssel anziehen, um den Dichtungsring zusammenzupressen.

ACHTUNG: Die Zündkerzen müssen fest angezogen werden, aber sie sollten nicht überdreht werden. Eine Zündkerze, die zu lose ist, kann sehr heiß sein, und sie kann den Motor möglicherweise beschädigen, aber eine zu feste Zündkerze kann die Gewinde in dem Zylinderkopf beschädigen.

Figure 14: *A snapshop of a multilingual text generated by the TechDoc system*

`able-connection` action in the sample SPL would be realized by *disconnect* due to the lexicalization rule given in figure 11, which states that in case the object is a `cable` or its subconcept, *disconnect* is the proper realization, while in all other cases the system would opt for *remove*.

Figure 14 shows the system output for the `replace-spark-plug_plan` after it has been processed by the formatting module. The text reflects the current state of implementation of the prototype. Note that the texts still contain minor syntactic errors and some awkward lexical choices, but these can be removed by extending the grammar coverage and fine-tuning the lexical rules. More importantly, the texts convey the content to be realized in a satisfactory manner. In addition, they account for syntactic and lexical peculiarities of English and German, as identified in section 2. For instance, in the German version, directives are realized using an infinitive construction, whereas English uses the imperative. These differences do not pose any problems to multilingual generation, since all processes that are involved in mapping the abstract content representation to the linguistic surface are tailored towards the languages to be realized.

5 Knowledge Management

The first three stages of the knowledge engineering process have been described in the previous sections: the acquisition of knowledge, the representation and formalization of that knowledge, and finally its application. The latter plays a crucial role in the entire knowledge engineering process as it serves as a test bed for design decisions made in earlier phases. Feedback from this phase may yield changes in the knowledge bases, thus necessitating knowledge management facilities. The capitalization of knowledge makes further demands on the knowledge management task. As noted above, acquisition and representation of the knowledge required for a particular application and in a particular domain are time-consuming and cost-intensive. This ascribes a major role to the capitalization of knowledge in order to make the acquisition and representation effort worthwhile. In general, there are two possibilities to improve capitalization: by reducing the cost of creation and maintenance, and by increasing the benefits of a knowledge base. In this section, we will discuss how the initial costs for creation can be reduced, how maintenance can be optimized, and how capitalization can be increased by changing the way in which knowledge is managed in an enterprise.

5.1 Benefits

Our approach of representing knowledge in an explicit and sharable way instead of describing it by means of natural language texts (documents) has a lot in common with the broader area of *corporate memory.* In [van Heijst *et al.*, 1996] a corporate memory is defined as "an explicit, disembodied, persistent representation of the knowledge and information in an organization". A major advantage of a corporate memory is that by storing corporate know-how explicitly and making it available to the whole company, an enterprise's competitiveness in the marketplace can be improved [Kühn and Höfling, 1994]. This is, above all, due to the reduced risk of loss of information (experts may leave), better information flow (fast and easy access) and a reduction in the time and demands placed on human experts.

We argued above that at present, natural language documents constitute one of the major resources for the knowledge accumulated during product development and the product life cycle. We then demonstrated in an exemplary way how it is possible to arrive at an explicit representation of the knowledge contained in documents. The resulting knowledge bases could potentially be treated as a corporate memory of technical know-how in the car domain—of course, they would still have to be enriched.

An important observation regarding corporate know-how is that it is not static but changes over time. This could lead to problems regarding consistency and completeness of technical documentations. A crucial benefit of taking documents as a particular view on a common underlying knowledge base, and of producing documents from that knowledge base, is that changes in the corporate memory immediately find their expression in documents. In a nutshell, changes and updates need only be performed at one place, within the common representation; they will then be propagated to all new document versions produced from that knowledge base. Former versions will still be available. Obviously, this will improve the effectiveness of the information flow inside (and outside) a company considerably, since documents can be produced on demand, always reflecting the current state of the know-how. Further, documentation is constantly available in different languages for enterprises operating internationally and can be tailored to different users, levels of detail and situations.

A major benefit from using knowledge bases rather than documents for preserving a company's know-how is that this knowledge is *sharable*, i.e. the knowledge bases can be exploited by further applications in addition to the generation of multilingual technical documents. This holds for the abstract technical model; representing that model in a sharable and reusable way facilitates the realization of other types of applications. For instance, one could employ the knowledge for the qualitative simulation of instruction steps in order to check for completeness and applicability of a plan. Further, we are experimenting with the automatic derivation of hints and warnings. For example, if an action has an undesired side-effect, one might want to communicate potential dangers to the reader. Both applications are described in more detail in [Rösner, 1994] and [Liebig and Rösner, 1996b].

5.2 Costs and Possible Optimization

The benefits of having access to corporate know-how on the domain are partially balanced by the costs of acquiring such knowledge. An initial approach to knowledge acquisition by analysing natural language document has been described in section 2. At first glance, this seems to be a tedious, costly and time-consuming task, which is hardly outweighed by the benefits of the resulting knowledge bases. However, we can only judge the acquisition effort correctly when taking a long-term perspective on this matter. There is no doubt that knowledge acquisition does not pay-off if the knowledge bases are only used for one particular application and for a short period of time, and if knowledge sources constantly undergo significant changes.

When taking a closer look at the nature of the different knowledge sources under consideration, we notice the following: Most of the *linguistic knowledge*, like grammars and lexica, will remain stable over time. The macrostructure of a certain text type will not change either, it just might have to be adapted to changes in DIN (or ISO) norms. Similar document types can be defined as variants of the initial text types; additional analysis only has to be performed if new text types are considered. Thus, where the initial acquisition phase is executed carefully, this large effort only has to be performed once, and pays off in later use and reduced maintenance costs.

Concerning the *nonlinguistic resources* we noted the following: Since the *abstract technical model* describes general domain knowledge, it will only change as much as the understanding of the technical foundations as such changes. In other words, only if the company's know-how on technical matters undergoes substantial transitions will the abstract technical model have to be modified. Hence, the maintenance effort for this knowledge can be neglected, even in the most innovative companies. In contrast, the *product specific knowledge base* is subject to frequent change. To reduce maintenance efforts and to assure consistency, the maintenance of the product specific knowledge should at best be carried out by product engineers as part of their activities during the product life cycle. Once the underlying product knowledge is modelled in a formal way, changes to the product can be described formally by the product designer within this formal representation. As all the different realizations and multilingual variants are generated from this formal representation, the resulting documents necessarily reflect these changes. Thus, documentation will be tightly integrated into the overall engineering process.

Current practise in industrial companies, however, somewhat blurs this vision: In early stages of the product life cycle, design decisions are frequently not formalized, and product (knowledge) modelling begins only at a later design phase in combination with the definition of the geometric model. As a consequence, the output from this phase—in the worst case only technical drawings—does not reflect the motivations underlying certain design decisions (e.g. functional requirements). But these decisions and reasons are relevant for later stages like production planning. We believe that modelling this knowledge right from the start of the design phase and generating the documentations automatically would considerably improve the cooperation between different departments and engineers of a company.

In brief, the costs of building and maintaining knowledge bases mainly stem from the product specific knowledge base, since it is subject to frequent change, whereas all other knowledge bases stay constant over long periods of time. However, even the costs of maintaining product specific information can be significantly reduced by integrating this process into the product life cycle and the operating environment of product engineers.

5.3 A Practical Solution: The Authoring Tool

We noted that the main maintenance effort concerns the product specific knowledge. This calls for an optimization of the management of this resource. One way to achieve this is by integrating the maintenance task into the product life cycle, and by ascribing this task to product engineers and designers. Since they are usually not familiar with techniques for knowledge representation, this task has to be supported by special tools for the addition and modification of product knowledge. In an ideal case, these tools are integrated into software systems which are already in use (like engineering data managements (EDM) systems or CAD-software). In addition, the final creation of documentation should be supported by an authoring tool (which could be used, for instance, by the technical editors), where knowledge specifying the content of a particular document can be selected and combined into an abstract representation of that document.

In a follow-up project to TechDoc, a prototypical *document workbench* has been realized which provides interactive support in managing knowledge bases and creating documents [Grote *et al.*, 1997]. The three main components of this workbench are an authoring tool for the interactive planning and structuring of a document, a document generator which takes this specification as input and generates formatted multilingual documents, and a module to administer knowledge bases and support the integration of existing resources.

The interactive creation of a document using the document workbench is performed in the following way: The starting point is an abstract skeletal plan, which represents the macrostructure of the desired text type. Optional elements have to be selected by the author, while the system provides the obligatory elements. These are then successively specified and refined by the author, drawing on various knowledge sources. By selecting the desired content from the knowledge bases the author creates an instances of a `plan`, which is fed to the multilingual generation system (TechDoc) for realization.

6 Discussion

Most of the work relevant to our approach falls into one of the following three categories:

- theories and tools for (multilingual) natural language generation,
- work on instructional texts, especially in technical documentation,
- sharable and reusable knowledge sources.

Some of the results of other projects have been directly integrated into our work either as components (e.g. the PENMAN grammars for surface generation) or as part of the theoretical framework (e.g. RST or description logic). In the following we will discuss commonalities and differences with our work.

Multilingual text generation. Early work on natural language generation from conceptual structures has already demonstrated the feasibility and attractiveness of multilingual delivery of knowledge structures [Goldman, 1974]. Nevertheless, multilingual generation has only recently received broader attention in the natural language generation community. This may be partly due to the fact that in a generation system with a single target language (i.e. English in many cases), issues of language independence of the semantic representations do not play such a decisive role.

There are a number of applications other than technical documentation in which the use of multilingual generation as an alternative to (human or machine) translation of the respective texts is both attractive and feasible, e.g. generation of weather reports (in English and French) from meteorological data or generation of statistical reports from a body of statistical data (cf. e.g. [Bourbeau *et al.*, 1990]). These successful applications have in common the fact that the domains of discourse are rather limited and therefore there is no need for elaborate modelling. Further, the 'raw material' for generation can be obtained by evaluating the data. In short, the aspect of domain modelling is an issue which has so far received only little attention in the area of multilingual generation; an issue, however, which is central to the generation approach pursued in the TECHDOC project.

Generation of instructional text. Other projects in the generation of instructional text are more elaborate in some specialised topics. But to our knowledge, none of them aims at such an integrated and holistic treatment of document generation as we have proposed, covering aspects of generation with the same effort as those of knowledge acquisition, knowledge representation and knowledge sharing and reuse.

The projects WIP [Wahlster *et al.*, 1993] and PPP [André *et al.*, 1996], for example, which aim at the automatic generation of multimodal technical documentations, have developed sophisticated techniques for coordinating text and graphics. Even though there exists a multilingual version of the system, they have payed only little attention to issues of generality in domain modelling and to reusing or extending the knowledge bases.

DRAFTER [Paris and Vander Linden, 1996] is another project in automatic generation of multilingual instruction texts that employs support for authoring. Their emphasis has been on the role of the technical author in the production of technical documentation and on providing tools for drafting document structures. Only recently have they given more attention to issues and principles of domain modelling.

Multi-purpose generation of personalised patient instruction and information is the aim of the HEALTHDOC [DiMarco *et al.*, 1995] project. They try to avoid domain modelling and start generation as a selection process from a so-called 'master document' already geared towards English as a target language. It is difficult to interpret this structure as formal knowledge representation and to imagine that it could serve as a basis for multilingual generation.

Knowledge representation. The need for reuse and exchange of knowledge bases has led to a number of research projects and standardisation efforts. An example of the latter is work on KIF [Genesereth and Fikes, 1992], a knowledge interchange format that serves as interlingua for the translation between different knowledge representation formalisms, and thus supports the incorporation of existing knowledge bases into other knowledge sources.

Ontologies [Uschold and Gruninger, 1996] are an attempt to ease the construction of knowledge based systems by providing reusable knowledge sources about recurring topics. Some ontologies are now available in the ontology library [Gruber, 1993]. We are investigating how to make better use of these resources and how sources taken from this library may be harmonised with our abstract technical model.

On the other hand, our approach to knowledge acquisition through the analysis of multilingual documents guided by abstraction principles may very well be interpreted as a methodology (among others) for the principled construction of ontologies. However, one needs to employ (semi-)automated tools for that task in in order to reduce the costs of building knowledge bases. There have recently been efforts on knowledge acquisition by constructing semantic knowledge bases (ontologies) semi-automatically, primarily within the PANGLOSS project on knowledge-based machine translation [Knight and Luk, 1994]; and, in the area of knowledge based systems, on terminology extraction [Dagan and Church, 1994], [Justeson and Katz, 1995]. Efforts within the natural language generation community on knowledge acquisition are described in [Reiter *et al.*, 1997] and [Paris and Vander Linden, 1996].

7 Conclusions and Future Work

We presented a novel approach to knowledge management by

- introducing the notion of documents as views on particular knowledge structures in a particular context;
- describing techniques for the acquisition of knowledge contained in documents;
- illustrating how to create knowledge bases which can be used for automated document generation and which are reusable for other applications (e.g. qualitative simulation of actions proposed in a plan);
- presenting suggestions on how to integrate such an approach into the engineering process and discussing potential consequences for the whole product life cycle.

The development of our work has continually been accompanied by and has profited from a lot of discussions with experts from companies of various size (SMEs as well as internationally operating concerns) and from different industrial sectors. There has been much support and acceptance for the general approach but some critical questions raised demand further investigation and evaluation of the approach 'in the field':

- What is the relation between cost and benefit, i.e. how much effort has to be initially invested into knowledge acquisition before the potential advantages of such a corporate memory are at least balanced?
- Will engineers accept the approach and use the authoring tools offered?
- Is convergence and cross-fertilization with other approaches to modelling of product data and product knowledge (e.g. STEP) possible?

In addition to such evaluation issues, our future work will include investigating into ways to make the knowledge acquisition phase more efficient and less time-consuming, for instance by employing semi-automated tools for the analysis of large text corpora. Here, we will draw on existing work on terminology extraction and semantic clustering [Dagan and Church, 1994], [Justeson and Katz, 1995].

Furthermore, future work has to focus on the integration of our approach with existing engineering data management systems. These systems contain product knowledge such as geometrical data and configuration of components, but lack, for example, knowledge on instructions for use, which are, however, represented in our approach. Hence, we need to investigate into a tight integration of both approaches.

A major concern will be if and how our approach will be transferable to other text types and their underlying knowledge structures. Concerning technical documentation, this will include further analysis of, for instance, explanatory manual texts (i.e. explaining the normal functioning and usage of technical objects rather than their repair and maintenance) and guides for trouble-shooting, and their relation to function models and failure models respectively. Explanatory texts for use in intelligent tutoring systems (ITS) and for interactive advisory systems seem to be other promising candidates for further elaboration of our approach.

References

[André *et al.*, 1996] E. André, J. Müller, and T. Rist. The PPP Persona: A Multipurpose Animated Presentation Agent. In *Advanced Visual Interfaces*, pages 245–247. ACM Press, 1996.

[Artale *et al.*, 1996] A. Artale, E. Franconi, N. Guarino, and L. Pazzi. Part-Whole Relations in Object-Centered Systems: An Overview. *Data & Knowledge Engineering*, 20(3):347–383, December 1996.

[Bateman *et al.*, 1994] J. A. Bateman, B. Magnini, and F. Rinaldi. The Generalized *italian,german,english* Upper Model. In *Proc. of the ECAI'94 Workshop on Comparison of Implemented Ontologies*, Amsterdam, 1994.

[Bourbeau *et al.*, 1990] L. Bourbeau, D. Carcagno, E. Goldberg, R. Kittredge, and A. Polguère. Bilingual Generation of Weather Forecasts in an Operations Environment. In H. Kargren, editor, *Proc. of the 13th International Conference on Computational Linguistics*, pages 318–320, Helsinki, Finland, 1990.

[Brachman and Schmolze, 1985] R. J. Brachman and J. Schmolze. An Overview of the Kl-ONE Knowledge Representation System. *Cognitive Science*, 9(2), 1985.

[Brachman *et al.*, 1991] R. Brachman, D. McGuinness, P. Patel Schneider, L. Resnick, and A. Borgida. Living with CLASSIC: When and How to Use a KL-ONE-Like Language. In J. Sowa, editor, *Principles of Semantic Networks.* Morgan Kaufmann, 1991.

[Dagan and Church, 1994] I. Dagan and K. Church. Termight: Identifying and Translating Technical Terminology. In *Proc. of the European Chapter of the Association for Computational Linguistics (EACL-94)*, pages 34–40, 1994.

[DiMarco *et al.*, 1995] C. DiMarco, G. Hirst, L. Wanner, and J. Wilkinson. HealthDoc: Customizing Patient Information and Health Education by Medical Condition and Personal Characteristics. In *Workshop on Artificial Intelligence in Patient Education*, Glasgow, August 1995.

[Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2:198–208, 1971.

[Genesereth and Fikes, 1992] M. E. Genesereth and R. E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[Goldman, 1974] N. M. Goldman. *Computer Generation of Natural-Language from a Deep Conceptual Base.* PhD thesis, Department of Computer Science, Yale University, 1974.

[Grote *et al.*, 1993] B. Grote, D. Rösner, and M. Stede. From Knowledge to Language—Three Papers on Multilingual Text Generation. Technical report, Forschungsinstitut für angewandte Wissensverarbeitung (FAW) (Research Institute for Applied Knowledge Processing), Ulm, Germany, 1993.

[Grote *et al.*, 1997] B. Grote, M. Moll, and T. Rose. Ein Arbeitsplatz zur teilautomatisierten Erstellung mehrsprachiger technischer Dokumentation (Dokumentationsarbeitsplatz). Abschlußdokumentation, Forschungsinstitut für angewandte Wissensverarbeitung (FAW) (Research Institute for Applied Knowledge Processing), Ulm, Germany, April 1997.

[Gruber, 1993] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

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

[Heinson *et al.*, 1994] J. Heinson, D. Kudenko, B. Nebel, and H.-J. Profitlich. An Empirical Analysis of Terminological Representation Systems. *Artificial Intelligence*, 2(68):367–397, 1994.

[Justeson and Katz, 1995] J. S. Justeson and S. M. Katz. Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text. *Natural Language Engineering*, 1:9–27, 1995.

[Kasper, 1989] R. T. Kasper. A Flexible Interface for Linking Applications to PENMAN's Sentence Generator. In *Proc. of the DARPA Workshop on Speech and Natural Language*, Marina del Rey, USA, 1989. USC/Information Sciences Institute.

[Knight and Luk, 1994] K. Knight and S. Luk. Building a Large Knowledge Base for Machine Translation. In *Proc. of the American Association of Artificial Intelligence Conference (AAAI-94)*, Seattle, WA, 1994.

[Kühn and Höfling, 1994] O. Kühn and B. Höfling. Conserving Corporate Knowledge for Crankshaft Design. In *Seventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-AIE'94)*, Austin Texas, 1994.

[Liebig and Rösner, 1996a] T. Liebig and D. Rösner. Description Logic as Core Machinery for the Automatic Generation of Multilingual Technical Documents. In *Collected Papers of the 1996 International Description Logics Workshop*, Boston, MA, USA, November, 2–4 1996. AAAI Press.

[Liebig and Rösner, 1996b] T. Liebig and D. Rösner. Modelling of Reuseable Product Knowledge in Terminological Logics – a Case Study. In *Proc. of the First International Conference on Practical Aspects of Knowledge Management (PAKM'96)*, Basel, Switzerland, October, 30–31 1996.

[LOOM, 1991] The LOOM Knowledge Representation System. Documentation Package, USC/Information Sciences Institute, Marina Del Rey, CA., 1991.

[Mann and Thompson, 1987] W. C. Mann and S. A. Thompson. Rhetorical Structure Theory: A Theory of Text Organization. In L. Polanyi, editor, *The Structure of Discourse*. Ablex, Norwood, N.J., 1987.

[Mann, 1983] W. C. Mann. An Overview of the PENMAN Text Generation System. In *Proc. of the National Conference on Artificial Intelligence*, pages 261–265. AAAI, August 1983.

[Matthiessen and Bateman, 1991] C. Matthiessen and J. A. Bateman. *Text Generation and Systemic-Functional Linguistics: Experiences from English and Japanese.* Frances Pinter Publishers and St. Martin's Press, London and New York, 1991.

[Paris and Vander Linden, 1996] C. Paris and K. Vander Linden. DRAFTER: An Interactive Support Tool for Writing Multilingual Instructions. *IEEE Computer, Special Issue on Interactive Natural Language Processing*, July 1996.

[Reiter and Mellish, 1992] E. Reiter and C. Mellish. Using Classification to Generate Text. In *Proc. of the 30th Annual Meeting of the Association for Computational Linguistics (ACL'92)*, 1992.

[Reiter *et al.*, 1997] E. Reiter, A. Cawsey, L. Osman, and Y. Roff. Knowledge Acquisition for Content Selection. In *Proc. of the 6th European Workshop on Natural Language Generation*, Duisburg, 1997. Gerhard-Mercator Universität.

[Rösner and Stede, 1992] D. Rösner and M. Stede. Customizing RST for the Automatic Production of Technical Manuals. In R. Dale, E. H. Hovy, D. Rösner, and O. Stock, editors, *Aspects of Automated Natural Language Generation – Proc. of the 6th International Workshop on Natural Language Generation*, pages 199–214. Springer, Berlin, 1992.

[Rösner and Stede, 1994] D. Rösner and M. Stede. Generating Multilingual Documents from a Knowledge Base: The TECHDOC Project. In *Proc. of the 15th International Conference on Computational Linguistics (COLING'94)*, pages 339–346, Kyoto, Japan, 1994.

[Rösner, 1994] D. Rösner. *Automatische Generierung von mehrsprachigen Instruktionstexten aus einer Wissensbasis.* Habilitationsschrift, Fakultät für Informatik der Universität Stuttgart, 1994.

[Roth, 1982] K. Roth. *Konstruieren mit Konstruktionskatalogen.* Springer, Berlin, Berlin, 1982.

[Scott *et al.*, 1991] A. Scott, B. Clayton, and F. Gibson. *A Practical Guide to Knowledge Acquisition.* Addison-Wesley, New York, 1991.

[Simon, 1996] G. Simon. Knowledge Acquisition and Modelling for Corporate Memory: Lessons Learned from Experience. In *Proc. of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW 96)*, 1996.

[Stede, 1996] M. Stede. Lexical Semantics and Knowledge Representation in Multilingual Sentence Generation. Doctoral disseration CSRI-347, Computer Systems Research Institute, University of Toronto, June 1996.

[Uschold and Gruninger, 1996] M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review*, 11(2):93–136, 1996.

[van Heijst *et al.*, 1996] G. van Heijst, R. van der Spek, and E. Kruizinga. Organizing Corporate Memories. In *Proc. of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW 96)*, 1996.

[Wahlster *et al.*, 1993] W. Wahlster, E. André, W. Finkler, H.-J. Profitlich, and T. Rist. Plan–Based Integration of Natural Language and Graphics Generation. *Artificial Intelligence*, 63:387–427, 1993.

[Winston *et al.*, 1987] M. E. Winston, R. Chaffin, and D. Herrmann. A Taxonomy of Part-Whole Relations. *Cognitive Science*, 11:417–444, 1987.