Proc SIGDOC 92, Ottawa., ACM.

BEYOND HYPERTEXT: KNOWLEDGE MANAGEMENT FOR TECHNICAL DOCUMENTATION

Timothy C. Lethbridge
Doug Skuce

Department of Computer Science
University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
(613) 564-8155 tcl@csi.uottawa.ca
(613) 564-5418 doug@csi.uottawa.ca

ABSTRACT

We describe the use of our *knowledge management* system, CODE2, as an aid to documenters of a complex software system called Telos. CODE2 was first used by the designers of Telos to clarify their design concepts and terminology. CODE2 served the following purposes: 1) to acquire the knowledge about the system; 2) to check the terminology in natural language documents; 3) as an on-line knowledge resource for documenters and end-users, and 4) to automatically generate parts of the printed documentation. We describe some features that make CODE particularly useful to documenters: its sophisticated user interface, its ability to handle both formal and informal knowledge, and its support for language. We also describe our vision for the future of such knowledge-based technology.

1. INTRODUCTION

Documenters and knowledge engineers have some common goals: To gather and perfect knowledge, and to present it in such a way that others can easily make use of it. Currently these two groups of professionals communicate very little: Outside the documenter's community, documentation is not yet appreciated as being important, despite that fact that it is still our main means of collecting and disseminating knowledge. Conversely, probably few people working in documentation are aware of developments in knowledge engineering. The work described here represents a merging of these areas.

We have developed a knowledge management system called CODE21 [Skuce, 1992a] that can be used by both of these groups and others2 [Skuce and Meyer, 1990] [Meyer and Skuce, 1992]. In this paper we first compare the tasks of knowledge engineers and documenters and then describe how CODE2 can help bridge the gap, based on practical experience in which CODE2 assisted documenters in a software-development environment. Finally, we describe our vision for how knowledge engineering and documentation technology should merge.

2. COMPARING THE TASKS OF KNOWLEDGE ENGINEERS AND DOCUMENTERS

We can consider there to be three fundamental steps in the documentation process: 1) initially gathering knowledge3, 2) verifying and perfecting the knowledge, and 3) packaging or formatting it for use (traditionally in printed form, but increasingly in an on-line form perhaps involving hypertext or CD-ROM).

A similar situation is faced by knowledge engineers. The knowledge-gathering phase is traditionally called 'knowledge acquisition', the verification phase is sometimes called 'knowledge debugging', and the packaging phase has traditionally involved developing software such as expert systems that permit a user to use the knowledge for some purpose.

The types of knowledge manipulated in the two fields tend to differ as follows: Documenters mainly use natural language and diagrams that cannot be readily understood by *computers*; knowledge engineers mainly use formal mathematical or computer representations that cannot be readily understood by many *humans*.

The above differences are beginning to narrow, and our work reflects a desire to merge both capabilities seamlessly in one tool:

• Documenters are increasingly interested in presenting information on-line4· This necessitates dividing the knowledge into small chunks (e.g. 'cards' or hypertext nodes), indexing it, and adding other features that allow the computer to make decisions about what to present and how. Additionally, some technical documentation is beginning to use a restricted English syntax and carefully controlled terminology [AECMA, 1989], so that humans can better understand it, particularly tbose with

---

1 CODE stands for Conceptually Oriented Design/Description Environment. The latest version of the system is called CODE4; this is a complete redesign of earlier systems (CODE2 and CODE3), incorporating many new ideas. The practical experience reported in this paper was gained using CODE2.

2 Terminologists are another group of professionals with similar goals for whom CODE has proved useful.

3 The terms 'knowledge' and 'information' could be used here somewhat interchangably. Practically all of what documenters refer to as *information* would be called *knowledge* by knowledge engineers.

4 The present conference attests to this.

limited knowledge of English or the subject matter. A benefit of this is that there is more potential for computers to be actively involved in semantic analysis.

• Knowledge engineers, on the other hand, are finding that it is important to ensure their computer systems can manipulate informal knowledge[5] [Lethbridge, 1992]. There are many reasons for this: 1) It allows more creativity and flexibility during knowledge acquisition (it is very difficult for humans to correctly formalize knowledge, at least initially when ideas are not yet clear), 2) It allows knowledge bases to be better understood by a wider range of humans, and 3) It allows for sharing of knowledge between systems that cannot accommodate each others' formal semantics. Knowledge engineers are also recognizing the importance of presenting knowledge in an appealing way [Lethbridge, 1991]: Some recent knowledge-based systems (including CODE2 and its successor, CODE4) can process or generate restricted natural language and have graphical user interfaces with outlining, hypertext, and graph-drawing capabilities.

3. FEATURES OF CODE2 THAT FACILITATE DOCUMENT CREATION AND USE

Figure 1 shows a typical view of CODE2 on a workstation screen. Knowledge is organized by *topics* in hierarchies; each topic is identified by either a node in a graph, or as an item in a textual browser that functions somewhat like an outline processor. The actual knowledge for a topic can be viewed in several ways; in Figure 1, a topic view is open for the topic 'actor specification'. The browser is set to the statement describing the property 'behaviour ports' of this topic.

The following features of CODE2 (and its successor, CODE4) make it particularly suited to use by documenters and subsequently by users of the knowledge. The same features, however, have significant benefits for the knowledge engineer, a fact many of the latter do not yet appreciate [Lethbridge, 1992]:

• The underlying knowledge representation schema

CODE2's knowledge representation has a linguistic orientation: Facts are expressed as statements about topics. Rules exist for how the components of these statements should be expressed, but they need not be followed if they are deemed too restrictive. Thus statements can contain informal natural language or a machine-interpretable subset of (almost) natural language we call ClearTalk[6]. There is also a lexicon that permits queries about meanings and uses of terms (e.g. "show me how this term is defined", or "is this term used in more than one sense?", so that terminology can be better controlled (Figure 2).

• Acquisition facilities

CODE2 is able to assist the documenter to structure knowledge: 1) It can guide him or her to organize the knowledge being entered, particularly through the use of generalization ('isa') hierarchies, in which more general statements "inherit" to more specific subjects[7], 2) It can assist in detecting errors of various kinds, particularly misused terms, and 3) it can assist in the knowledge-review process by recording information such as who has agreed with each statement.

• Text scanning

CODE2 has a facility to scan a natural language document looking for terms it does not know, which can be added to the lexicon. It can be set to prompt the user to verify that certain terms, known to be problematic, are being used in an appropriate sense (Figure 3).

• Documentation generation

The user can ask CODE2 to generate documents containing both diagrams and text. Such documents typically include glossaries and 'concept primers', which CODE2 can deliver in a highly readable format. As explained in the next section, these can be prepared for publishing with little modification, relieving the documenter of an arduous task. CODE2 has sophisticated facilities for selecting and arranging knowledge: This permits generated documents to be tailored to a variety of uses. New documents can be easily produced almost automatically whenever the knowledge base is changed

Figure 4 shows a topic description as output from CODE2 into a publishing tool. Figure 5 shows an example of a "semantic net" diagram, that shows relationships between concepts, also produced automatically.

• On-line browsing

CODE2 can be directly used by the end user as an on-line documentation facility. The user can navigate around knowledge using a hypertext-like facility. CODE2's browsing facilities are considerably more powerful than typical hypertext though: The experienced user can use masking, dynamic querying and diagram drawing capabilities to generate entirely new presentations of the knowledge.

Figure 6 shows a browser set to show only the hierarchy of topics involving 'reference'. A mouse click will bring up a complete view of any description, such as that shown in Figure 7. A complete textual listing of any topic or topics can be obtained as in Figure 8, for off-line reference.

---

5 Documenters generally express themselves in natural language, which may well be highly structured, but which is nevertheless 'informal' in the sense that today's computers cannot understand it. Knowledge engineers have traditionally worked at the other extreme, where knowledge must be expressed in a formal, mathematical language.

6 Most of the examples in this paper use ClearTalk

7 e.g. a statement such as "cars have wheels" inherits to specializations of cars such as "sports cars"

4. THE BNR EXPERIMENT

At BNR (Bell Northern Research) we worked with a group of some fifty people who were developing a large, complex piece of software called Telos [Skuce, 1992b]. Our involvement lasted about eighteen months. The first few months were largely spent with software engineers and their managers reverse engineering the system as it existed, i.e. obtaining the basic knowledge about the system. Later, we actively assisted the engineers to design new features using CODE2 as a "designer's assistant".

Towards the end of the project the documenters became interested in CODE2 as they realized that much of the knowledge they would need to obtain and then describe in technical documentation was already captured in CODE2, avoiding the need for frequent discussions with the engineers to explain unclear concepts and terminology (one of CODE2's main purposes). Furthermore they realized the advantage of having one common source for documentation that could be used for the verification and generation of documents. This contrasted sharply with the common situation where the documenters have to work with a group of engineers who have difficulty agreeing on a common set of concept descriptions and appropriate terminology. Typically, knowledge is dispersed in many incomplete documents prepared without adequate attention to terminology or correctness. Often, much of the knowledge is still only in the heads of the engineers, the situation known to knowledge engineers as the "knowledge acquisition bottleneck".

Once CODE2 was recognized as the central repository for knowledge (the group officially designated it as such), the following types of documents or other information resources could be produced from the knowledge base:

• The on-line help system.

• A user's reference manual.

• Much of the official ‘concepts’ guide, that explains the whole system and its components at a conceptual level (Figure 4 is an example of such a concept description.)

• System maintenance information.

The documentation team found the knowledge base to be such a valuable resource that they took charge of maintaining it. Indeed, they became what every project will hopefully someday have: the resident knowledge engineers.

Suggestions were also made that CODE2 could be integrated directly into Telos, both to provide a next-generation on-line help system about Telos itself, and to allow for the management of knowledge that Telos *itself* would generate, i.e. knowledge about software designed using Telos.

5. A VISION OF THE FUTURE: THE MERGING OF TECHNOLOGIES

In future we see a convergence of the present assortment of functionally diverse systems such as knowledge-based systems, hypertext systems, documentation systems, terminology support systems, and on-line help systems. At present, each of these is still being designed and deployed to meet the needs of a particular group of users in a particular kind of situation.

Our claim, however, is that these users and situations have much in common: they all need rapid access to the kinds of knowledge that today are being captured and disseminated using document production tools. But today's document production tools have no knowledge management functions; they are not designed by or for knowledge engineers. In the future, both groups will merge their expertise in designing tools that meet both kinds of needs. CODE2 represents an early step in this direction that has already found practical application in industry.

In our vision of the future we expect that engineers, managers, educators, documenters and others will all contribute to a comprehensive knowledge base describing many aspects of a topic. The knowledge will be carefully managed with the help of sophisticated software, to ensure that it is correct, complete and consistent. The knowledge base will be treated as a large hypertext document and directly browsed by some users. Other users will run programs that make computational use of the knowledge. Still other users will choose to generate documents that fulfil special needs (e.g. beginners tutorials, context-sensitive on-line help). These documents will be automatically updated when the knowledge base changes.

In this scenario, the documenter's main role will change from that of writing to that of coordinating the gathering of knowledge and structuring the knowledge base so that useful documents can be automatically generated.

ACKNOWLEDGEMENTS

Support for this research has been provided by Bell-Northern Research and the Natural Sciences and Engineering Research Council of Canada. We wish to thank the Telos group at BNR, particularly Roy MacLean and his documentation team who so enthusiastically embraced our technology and provided many suggestions. We also wish to thank Yves Beauvillé who was responsible for much of the programming.

REFERENCES

AECMA (1989) *ACEMA Simplified English*, Association Europeenne des Constructeurs de Materiel Aerospatial, PSC-85-16598

Lethbridge, T. C. (1991). "Creative Knowledge Acquisition: An Analysis". *6th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff

Lethbridge, T. C. and D. Skuce (1992). "Informality in Knowledge Exchange". *AAAI Workshop on Knowledge Representation Aspects of Knowledge Acquisition*, San Jose

Lethbridge, T. C. and D. Skuce (1992). "Integrating Techniques for Conceptual Modeling". *7th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff

Meyer, I., D. Skuce, et al. (1992). "Towards a New Generation of Terminological Resources: An Experiment in Building a Terminological KB". *13th International Conference on Computational Linguistics (COLING)*, Nantes

Skuce, D. (1992a). “A Wide Spectrum Knowledge Management System.” *To appear in: Knowledge Acquisition* : 49 pp.

Skuce, D. (1992b) Managing Software Design Knowledge: A Tool and an Experiment. Submitted with reviewer's changes to: *IEEE Transactions of Knowledge and Data Engineering*. 45pp.

Skuce, D. and I. Meyer (1990). "Concept Analysis and Terminology: A Knowledge-based Approach to Documentation.". *Proc. 13th International Conference on Computational Linguistics (COLING)*, Helsinki

Figure 1. An overall view of CODE2 on a workstation screen. There are three main windows open. In the background is a graph labelled ‘TelosKBase76’ showing topics arranged in a generalization hierarchy. At the right is a ‘Property Browser’. This also shows a generalization hierarchy in its left subwindow. Its top-right two subwindows list properties of the selected topic. Below these are subwindows giving details of the selected topic and property. The window in the bottom left of the figure shows one way that a collection of properties of a topic can be presented. The knowledge in any of these windows can be edited, and the user can navigate in a hypertext-like mode. The complexity of the knowledge in this figure is attributable to the subject matter.

Figure 2. A lexicon browser. Words can have many meanings and can perform as various parts of speech. The lexicon browser permits the editing of this type of knowledge. Once lexical information is entered into the system, the ClearTalk parsing mechanism is better able to validate knowledge. In this figure the left subwindow contains a hierarchy of lexicons (of varying importance). The middle subwindow contains words known in the current knowledge base. The right subwindow contains lexical information about one of the senses of the selected word.

Figure 3. A text scanner. Using the existing knowledge base, the system scans an natural language document for words it does not recognize. The system then presents various prompts so that the knowledge can be entered into the system.

Figure 4. An example of a page produced directly from CODE2 for prototype Telos documentation. The graph fragment at the top relates the topic ‘Binding Specification’ to related concepts. The rest of the page contains pseudo-natural-language generated by CODE2 and then post-processed by a word processor.

Figure 5. An example of a semantic net diagram produced directly from CODE2. The boxes represent topics in the knowledge base, and the arcs show how these various concepts are related to each other.

Figure 6. A detailed view of a property browser. The browser has been focussed on the concept 'reference'. ‘Fixed actor reference’ (a specialized kind of reference) has been selected and its properties are displayed on the top-right subwindow. The property ‘equivalence

Figure 7. A screen view of part of a topic description. The top left subwindow shows how this topic relates to others. The top center subwindow contains knowledge management information. The top right subwindow contains linguistic and commentary information. Other subwindows contain categories of properties particular to this topic.

Figure 8. An example of a textual listing of a topic description. Details of properties of ‘actor specification’ divided up into categories. The terms in angle brackets refer to the most general topic to have this property.