CONCEPT ANALYSIS AND TERMINOLOGY: A KNOWLEDGE-BASED APPROACH TO DOCUMENTATION

Douglas Skuce
Dept of Computer Science
University of Ottawa, Canada
doug@uotcsi2.bitnet

Ingrid Meyer
School of Translators and Interpreters
University of Ottawa, Canada
ixmal@uottawa.bitnet

ABSTRACT The central concern of terminology, a component of the general documentation process, is *concept analysis*, an activity which is becoming recognized as fundamental as term banks evolve into knowledge bases. We propose that concept analysis can be facilitated by knowledge engineering technology, and describe a generic knowledge acquisition tool called CODE (Conceptually Oriented Design Environment) that has been successfully used in two terminology applications: 1) a bilingual vocabulary project with the Terminology Directorate of the Secretary of State of Canada, and 2) a software documentation project with Bell Northern Research. We conclude with some implications of computer-assisted concept analysis for terminology.

1. TERMINOLOGY AND CONCEPT ANALYSIS

Terminology, the discipline concerned with the formation, description and naming of concepts in specialized fields of knowledge, is a key component of the general documentation process: it is normally preceded by knowledge acquisition (usually not formalized), and followed by document preparation. While it is still very common for one person to be responsible for all stages of the documentation process, terminological activities are increasingly becoming a distinct specialization, due to 1) the exponential growth of technical concepts, and consequent interest in terminology banks, electronic dictionaries, and various other computer aids for terminology; 2) the growing need for efficient transfer of highly specialized knowledge across national and linguistic boundaries, and associated demand for regulating the terminology of specialized domains; 3) the increasing recognition by corporations that high-quality documentation, which presupposes high-quality terminology, is an important factor in the success of a product.

Concept analysis involves 1) the description of concepts through an enumeration of their characteristics, or properties, and 2) the description of relations that hold within systems of concepts. It is generally agreed, and particularly stressed by the Vienna School of Terminology (Wüster 1985), that concept analysis is the central concern of terminology, essential to delimiting and partitioning nomenclatures, constructing definitions, distinguishing quasi-synonyms, dealing with neology, carrying out multilingual terminological analysis, and communicating with subject-field experts. Despite its importance, however, concept analysis is still done in an ad hoc fashion: to date, no developed methodology exists. Only rarely does one find graphical or structured textual presentations of concept systems in terminological publications: rather, one normally detects only traces of conceptual structures in the definitions of certain terms, "somewhat like a puzzle that no one can put together because there are pieces missing, and there is no picture of the whole that can serve as a guide" (translated from Kukulska-Hulme and Knowles 1989:382).

Apart from the lack of established methodology, a number of factors contribute to the difficulty of formalized concept analysis: 1) the terminologist is often not an expert in his subject fields, and thus faces all the knowledge elicitation and representation problems that characterize the knowledge engineering process (Skuce *et al* 1989); 2) since any partitioning of reality is arbitrary to some degree, concept relations often occur in complex "layerings" (Sowa 1984:349); 3) consistency and conceptual clarity are difficult to maintain in fields that are large, multidisciplinary, or rapidly evolving (Meyer and Skuce 1990).

We belive that these problems cannot be solved adequately using "paper-and-pencil" or "do-it-all-in-my-head" methods. The need for computer assistance is becoming all the more crucial as term banks evolve into multifunctional knowledge bases (Budin *et al.*), with various applications becoming dependent on them - for example, management information, training, expert systems and machine translation. With the increasing focus on the knowledge component of terminological research comes a need for sophisticated documentation workstations that include a knowledge support tool.

2. CODE: A KNOWLEDGE SUPPORT ENVIRONMENT

CODE (for *Conceptually Oriented Design Environment*, Skuce *et al* 1989, Skuce 1989a, 1990) is a generic knowledge acquisition environment, written in Smalltalk, that runs on a UNIX, Macintosh or 386 machine. The system has been developed at the Artificial Intelligence Laboratory of the University of Ottawa, Canada, and a prototype has been tested in 2 terminology applications (described below). CODE's associated methodology

(Skuce 1989b) integrates knowledge representation ideas from artificial intelligence, and includes a logical and a natural language analysis component. It was also influenced by experience with major expert system tools like KEE and ART. CODE may be thought of as a "spreadsheet for ideas", the intended user being any person faced with the task of systematically organizing expert knowledge. This knowledge, whether obtained verbally or textually, is rarely presented as precisely as terminologists would like: conceptual and terminological confusion are the rule rather than the exception. CODE employs a flexible knowledge representation which permits considerable variety in style and degree of formality. It includes mechanisms for catching many conceptual and terminological errors and guidance towards correcting them.

CODE is organized around the two fundamental notions of *concept* and *property* . Concepts can be of two types: *class concepts* and *instance concepts*. For example, 'university' is a class concept with instances such as 'University of Ottawa'. A property is a unit of information that characterizes a concept, corresponding roughly to a succinct declarative sentence. CODE organizes knowledge into units called *conceptual descriptors* (CDs), which are analogous to frames in artificial intelligence or objects in object-oriented programming. CDs can be arranged in inheritance hierarchies, so that more specific concepts may inherit properties from more general ones. Inheritance is controlled by a system of "flags", which define the inheritance behaviour as a function of the kind of property and the kind of inheritance link.

CODE offers the following useful features for terminology:

1. Detection of inconsistencies. A well-developed system for controlling inheritance of properties helps the terminologist maintain conceptual clarity and consistency. For example, the logical behaviour of properties can be flagged as "necessary", "sufficient", "optional" or "typical"; the modifiability of properties (in subconcepts) can be flagged as "not permissible", "free", or "if logically consistent"; etc. When a change is made to a property at a high conceptual level, one is queried as to whether the change also applies to subconcepts. Similarly, when a concept is moved from one branch of the network to another, one is queried about the properties that will be affected. These and other mechanisms for checking inconsistencies allow the terminologist to do "what-if" experiments and obtain quick feedback about the desirability of changes

2. Flexible means of specifying relations and properties. CODE is not tied to any particular theory of concepts: the user can specify any properties and relations he wishes. As well as *hierarchical* relations (e.g. generic-specific, part-whole), the terminologist may also specify any number of user-defined *associative* relations (in the general sense of non-hierarchical).

3. Graphical and textual representation. The knowledge base can be visualized either by a graphical display, in the form of a directed graph, or by textual units, called *CD Views*. Any changes made on the graph are updated automatically in the corresponding CD Views, and vice versa. The graphical display is highly developed, offering features for managing large graphs, viewing multiple graphs (essential for multilingual terminology), indicating concepts and relations of special interest, and displaying hierarchical and associative relations.

4. Representation of multiple partitioning of reality. A subject field can often be partitioned in several ways, depending on which properties of concepts are emphasized. Since terminologists frequently need to take such multiple partitions into account, CODE offers two features of interest: 1) multiple inheritance is permitted, and certain properties can be blocked if necessary; 2) concepts can be assigned various keys, so that one can focus on only certain concepts within the knowledge base, or work with all conceptual partitions simultaneously.

5. Hypertext-like browsing capability. CODE's browsing facility, the Property Browser, allows the terminologist to "navigate" easily between concepts, between properties, and between concepts and properties. A multiple windowing capability allows simultaneous viewing of any number of graphs, CD views, and Property Browsers.

3. APPLICATIONS OF CODE

A. Bilingual terminology. During the fall of 1989, CODE was tested by Meyer in the Terminology Directorate of the Department of the Secretary of State of Canada. The Terminology Directorate practises terminology as a discipline in its own right. Its efforts are largely geared towards translation needs. Knowledge acquisition is a vital part of the terminology work at the Secretary of State: most of it is done from documents, although subject-field experts are frequently consulted as well. The amount of knowledge acquisition depends on the type of project: it is most important for *thematic* research of the *vocabulary* type, i.e. research aiming at a complete coverage of a specialized field, leading to a published work that includes definitions, and not simply bilingual lists of equivalents.

CODE was used in a vocabulary project for typesetting. The system served two purposes: 1) to formally represent knowledge that had already been acquired in the field, and that was reflected to some degree in a previous vocabulary - it was found that the formal representation lead to improvements on the previous definitions; 2) to systematize knowledge on emerging concepts in the field, particularly regarding the role of computerization.

B. Software documentation. Documentation is an essential aspect of the software production process, but unfortunately it is often not treated with sufficient care. Part of the problem is that careful conceptual analysis and terminological control are often not part of the design and development phases that precede documentation. Indeed, one of the goals in designing CODE was to help software engineers organize knowledge for themselves and for documentation. Ideally, knowledge (and hence terminology) should be systematized, edited, verified, maintained, and then distributed to those who need it throughout the whole software cycle. Typically, however, this is left to the documentalists, who, like the terminologists described above, must try to piece together a consistent description of the system after the fact.

An experiment in this application of CODE was completed in the fall of 1989 at Bell Northern Research, where Skuce spent some 60 days working closely with the designers of a new design environment for communications systems. The conceptual structure and terminology of this system were worked out in many long knowledge acquisition sessions. The resulting knowledge base is now being used to drive documentation production, on-line help, and subsequent design extensions.

4. CONCLUSIONS

A knowledge base produced with CODE can be seen as a "blueprint" for a documentation project, in that it clarifies the conceptual structure and terminology of the project from the outset. Just as one does not construct buildings without blueprints, systematic and computer-assisted concept analysis should be a prerequisite for the documentation process in general, particularly the terminology component. The following are just a few examples of the positive implications we foresee for our approach:

1. **Greater quality and multifunctionality of terminological data**, through a large and well-structured knowledge component.
2. **Terminological consistency between all phases of the documentation process**, (i.e. less of the "pass-my-confusion-onto-the-next-person" phenomenon).
3. **Enhanced communication among terminologists, other documentalists, and experts** through a shareable knowledge base. The documentalist's and terminologist's better understanding of the conceptual structure of an expert's field is also bound to enhance his credibility with the expert.
4. **Efficient training.** A knowledge base that offers efficient, on-line retrieval of information ensures conceptual continuity, and prevents a new documentalist from having to relearn his predecessor's field "from scratch".
5. **Improved transfer of knowledge across linguistic borders.** The translation process is a great bottleneck for efficient inter-linguistic knowledge transfer: both human and machine translation are greatly enhanced by correct terminology and conceptual clarity.

ACKNOWLEDGEMENTS

The projects described above were greatly enriched by discussions with C. Leonhardt, D. Michaud, L. Paradis, S. Pavel, and P. Weissenhofer at the Secretary of State, and with R. Aubin, I. Engelberg, M. Javor, J. McGee and B. Selic at Bell Northern Research. Skuce's research has been supported by COGNOS Inc., the National Sciences and Engineering Research Council of Canada (NSERC), the University Research Incentives Fund (URIF) of the Government of Ontario, and Research Services of the University of Ottawa. Meyer's research has been supported by Research Services of the University of Ottawa.

REFERENCES

BUDIN, G., Galinski, C., NEDOBITY, W., THALLER, R. 1988. "Terminology and Knowledge Data Processing". *Terminology and Knowledge Engineering* (Proceedings of the International Congress on Terminology and Knowledge Engineering, Trier, 1987), Ed. H. Czap and C. Galinski. Frankfurt: INDEKS Verlag.

KUKULSKA-HULME, Agnes and KNOWLES, Frank. 1989. "L'organisation conceptuelle des dictionnaires automatiques pour textes techniques". *META*, Vol. 34, No. 3.

MEYER, Ingrid and SKUCE, Douglas. 1990. "Computer-Assisted Concept Analysis: A Knowledge-Based Approach to Terminology". Paper presented at the EURALEX Fourth International Conference, Malaga.

SKUCE, Douglas. 1990 (in press). "A Language and System for Making Definitions of Technical Concepts". *Journal of Systems and Software.*

SKUCE, Douglas. 1989a. "A Generic Knowledge Acquisition Environment Integrating Natural Language and Logic". Proceedings IJCAI Workshop on Knowledge Acquisition (Detroit, Aug. 1989).

SKUCE, Douglas. 1989b. "Beyond Objects: A Synthesis of Objects, Logic, and Natural Language for Software Production". Submitted to IEEE Transactions on Knowledge and Data Engineering.

SKUCE, Douglas, WANG, S., and BEAUVILLE, Y. 1989. "A Generic Knowledge Acquisition Environment for Conceptual and Ontological Analysis". Proceedings Knowledge Acquisition for Knowledge-Based Systems Workshop (Banff, Canada, Oct. 1989).

SOWA, John. 1984. *Conceptual Structures.* Reading, MA: Addison-Wesley.

WÜSTER, Eugen. 1985. *Einführung in die Allgemeine Terminologielehre und Terminologische Lexikographie.* Vienna: Infoterm.