Integrating Textual Knowledge and Formal Knowledge for Improving Traceability

Farid Cerbah[1] and Jérôme Euzenat[2]

[1] Dassault Aviation - DPR/DESA - 78, quai Marcel Dassault
92552 cedex 300 Saint-Cloud - France
farid.cerbah@dassault-aviation.fr

[2] Inria Rhône-Alpes - 655, avenue de l'Europe
38330 Monbonnot St Martin - France
Jerome.Euzenat@inrialpes.fr – http://www.inrialpes.fr/exmo/

Abstract. Knowledge engineering often concerns the translation of informal knowledge into a formal representation. This translation process requires support for itself and for its traceability. We pretend that inserting a terminological structure between informal textual documents and their formalization serves both of these goals. Modern terminology extraction tools support the formalization process where the terms are a first sketch of formalized concepts. Moreover, the terms can be used for linking the concepts and the pieces of texts. This is exemplified through the presentation of an implemented system.

1 Introduction

Knowledge management is concerned with the relationships between formal and informal knowledge. The informal knowledge is richer and familiar to any user while the formal one is more precise and necessary to the computer. Moreover, translating from informal to formal is a common task of knowledge acquisition and providing traceability information is a major requirement. Therefore, this task requires computational support.

Several attempts were made to provide tools supporting the linking of knowledge sources [8, 14, 11]. However, they provided only limited computational support. The links had to be established manually and were thus error-prone and time consuming. In the meantime, several works focused on the advantages of using corpus-based terminology extraction for supporting formal knowledge acquisition [3, 1, 2]. These contributions emphasize the central role of terminological resources in the mapping between informal text sources and formal knowledge bases.

We argue that technical terms can play a key role in traceability too. We put forth an architecture, centered around a terminology extraction and management tool, that enables to generate models from texts and navigate from one to the other through the terminological structure (§2). It has been fully implemented with existing software (§3) and provides high-level hypertext generation, browsing and model generation facilities.

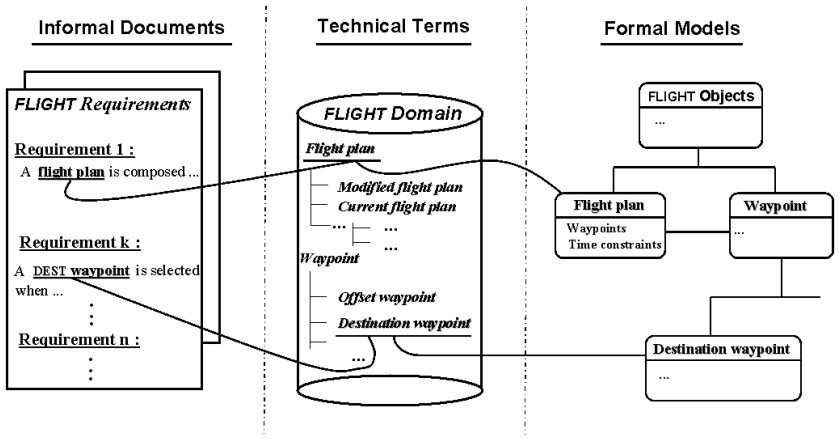

Fig. 1. Using terminological items to link textual requirements and object models

2 An architecture for traceability through terminological structures

When building a somewhat formal (or at least structured) repository from document sources, the concepts in the formal repository must be linked to their original sources in the texts. These traceability links are useful in many respects:

- Ensuring exhaustiveness: By following traceability links, the user or a program can easily identify the concepts which are not represented in the repository.
- Facilitating change propagation: At any time in the elaboration process, traceability information allows to find out the elements impacted by changes (upstream and downstream).
- Enhancing browsing capabilities of the overall repository, when traceability is established with hyperlinks.

In many information systems where both textual knowledge and formal knowledge are involved to describe related concepts, a terminological structure can play an intermediate role. Some of the technical terms found in the corpora represent concepts which will be subsequently introduced in the formal models. These terms can be seen as an intermediary level between the text found in documents and the formal models. (see figure 1).
In order to achieve both formalization and traceability, a system must articulate the following functions:

Terminology extraction. In technical domains, many precise and highly relevant concepts are linguistically represented by compound nouns. The multiword nature of the technical terms facilitates their automatic identification in

texts. Relevant multi-word terms can be easily identified with high accuracy using partial syntactic analysis [3, 9] or statistical processing [5] (or even both paradigms [6]). Terminology extraction techniques are used to automatically build term hierarchies that will play the intermediate role between documents and models.

Document and model indexing. The technical terms are used for indexing text fragments in the documents. Fine grained indexing, i.e paragraph level indexing, is required while most indexing systems used in information retrieval work at the document level. Besides, most descriptors used in this kind of indexing are multi-word phrases. The terms are also used for indexing the model fragments (classes, attributes...).

Hyperlink generation. The term-driven indexing of both texts and models with the same terminological structure is the basis of the hyperlink generation mechanisms. However, hyperlink generation should be controlled interactively, in the sense that the user should be able to exclude automatically generated links or add links that have not been proposed by the system.

Model generation. It is quite common that the concept hierarchies mirror the term hierarchies found in the documents. This property can be used to generate model skeletons which will be filled manually.

The integration of these functions within a single process results in a method for helping the acquisition and maintenance of formal knowledge from textual knowledge. It first extracts a terminological structure (which automatically indexes the document fragments). The terminological knowledge must be validated by the users which can generate a class taxonomy (also indexed by the terms).

3 A user support tool for improving traceability

The functions presented above are implemented by existing components (§3.1 and 3.2) which are linked in the appropriate way (§3.3).

3.1 Terminology extraction with XTerm

XTerm [4] is a natural language processing tool that performs terminology extraction from French or English documents and offers high level browsing capabilities through the extracted data and the source documents. Starting with a document collection, XTerm scans all document building blocks (paragraphs, titles, figures, notes) in order to extract the text fragments. These word sequences are then prepared for linguistic processing.

The first linguistic processing step is part of speech (pos) tagging. XTerm uses a rule based tagger based on the Multex morphological parser [13]. pos tagging starts with a morphological analysis which assigns to each word its possible morphological realizations. Then, contextual disambiguation rules are applied to choose a unique realization for each word. At the end of this process, each word is unambiguously tagged.

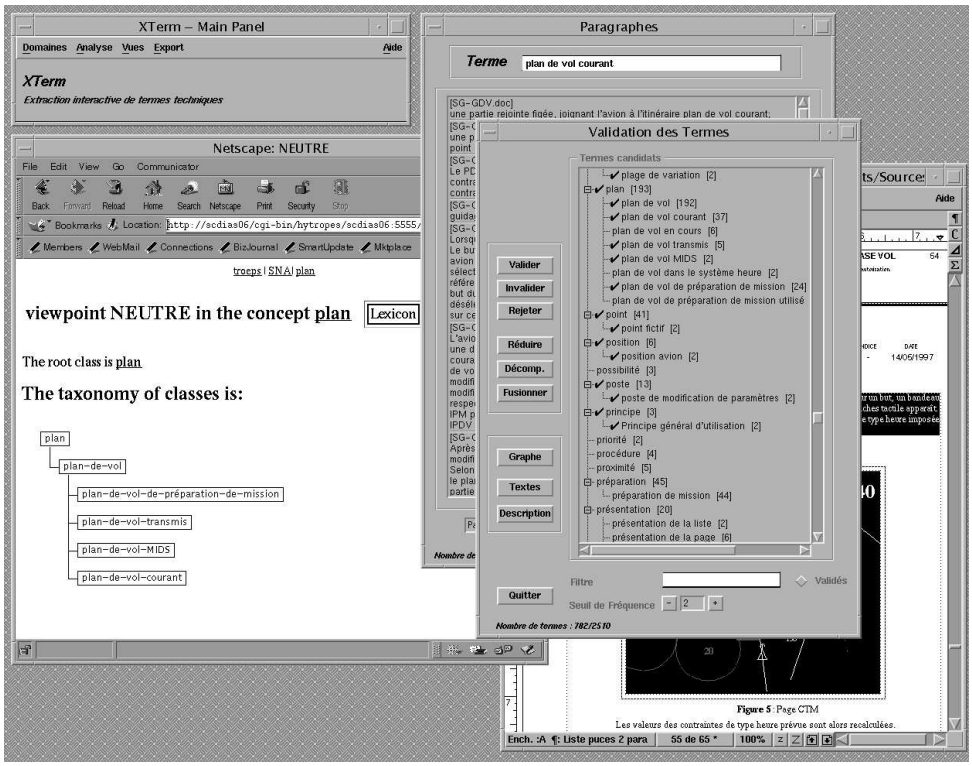

Fig. 2. The integrated system based on XTerm and Troeps.

As already mentioned, the morpho-syntactic structure of technical terms follows quite regular formation rules which represent a kind of local grammar. For instance, many French terms can be captured with the pattern "*Noun Preposition (Article) Noun*". Such patterns can be formalized with finite state automata, where crossing conditions of the transitions are expressed in terms of morphological properties. To identify the potential terms, the automata are applied on the tagged word sequences. A new potential term is recognized each time a final state is reached. During this step, the extracted terms are organized hierarchically. For example, the term "*flight plan*" ("*plan de vol*" in figure 2) will have the term "*plan*" as parent and "*modified flight plan*" as a child in the hierarchy.

The candidate set obtained after this step is still too large. Additional filtering mechanisms are involved to reduce it, including grouping rules that detect term variants (e.g. "*Waypoint page*" instead of "*page of the waypoints*").

Additionally, XTerm provides the mechanisms for indexing and generating hyperlinks from technical terms to document fragments. Hyperlink generation is a selective process: To avoid overgeneration, the initial set of links systematically established by the system can be reduced by the user.

3.2 Knowledge modeling with the TROEPS system

TROEPS [10, 15] is an object-based knowledge representation system, i.e. a knowledge representation system inspired from both frame-based languages and object-oriented programming languages. It is used here for expressing the models.
An object is a set of field-value pairs associated to an identifier. The value of a field can be known or unknown, it can be an object or a value from a primitive type (e.g. character string, integer, duration) or a set or list of such. The objects are partitioned into disjoint concepts (an object is an instance of one and only one concept) which determine the key and structure of their instances. For example, the "*plan*" concept identifies a plan by its number which is an integer. The fields of a particular "*plan*" are its time constraint which must be a duration and its waypoints which must contain a set of instances of the "*waypoint*" concept.
Object-based knowledge representation provides various facilities for manipulating knowledge including filtering queries (which find objects of a concept satisfying field and attachment constraints), similarity queries (function of field values or attachment classes) involving a distance measure, value inference (through default values, procedural attachment, value passing or filtering), position inference (classification and identification) in which the possible positions of an object or a class in a taxonomy are computed.
TROEPS knowledge bases can be used as HTTP servers delivering the knowledge to the world-wide web. These knowledge servers enable knowledge base browsing and editing from a HTTP client. Moreover, the knowledge is linked to other sources and can be manipulated through knowledge-based operations (e.g. filtering or classification). Lastly, TROEPS offers an XML interface which allows to describe a whole knowledge base or to take specific actions on an existing knowledge base.

3.3 Communication between the components

The communication between the linguistic processing environment and the model manager is bidirectional: Upon user request, XTERM can call TROEPS to generate class hierarchies from term hierarchies. Conversely, TROEPS can call XTERM to display the textual fragments related to a concept (via a technical term).
For instance, figure 3 illustrates the class generation process from a hierarchy of terms validated by the user (a hierarchy rooted in the term "*Plan*"). The class hierarchy constructed by TROEPS follows the hierarchy of the validated terms.
At the end of the generation process, the created classes are still linked to their corresponding terms, and so the terminology-centered navigation capabilities offered by XTERM are directly available from the TROEPS interface. As illustrated by figure 3, the TROEPS user has access to the multi-document view of the paragraphs where the term "*flight plan*" and its variants occur. From this view, the user can consult the source documents if required.
Data exchange between XTERM and TROEPS is based on the TROEPS XML interface. XTERM sends to TROEPS short XML statements corresponding to the action performed by the user: creation of a new class or a subclass of an existing

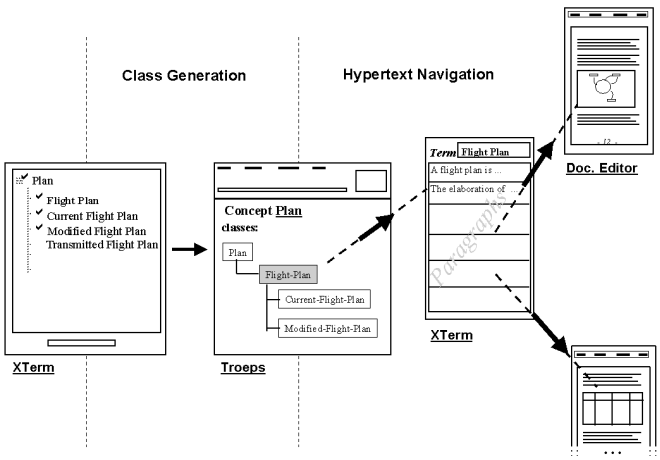

Fig. 3. Class generation and traceability through hyperlinks

class and the annotation of a newly created class with textual elements such as the outlined definition of the term naming the class.
This XML interface has the advantage of covering the complete TROEPS model (thus it is possible to destroy or rename classes as well as adding new attributes to existing classes). Moreover, it is relatively typical of object-based representation languages so that it will be easy to have XTERM generating in other languages (e.g. XMI [12] or Ontolingua) which share the notion of classes and objects.
More details about this approach of XML-based knowledge modeling and exchange can be found in [7].

4 Related work

Terminology acquisition is one of the most robust language processing technology [3, 9, 6] and previous works have demonstrated that term extraction tools can help to link informal and formal knowledge. The theoretical apparatus depicted in [3, 1, 2] provides useful guidelines for integrating terminology extraction tools in knowledge management systems. However, the models and implemented systems suffer from a poor support for traceability, restricted to the use of hyperlinks from concepts and terms to simple text files. On this aspect, our proposal is richer. The system handles real documents, in their original format, and offers various navigation and search services for manipulating "knowledge structures" (i.e., documents, text fragments, terms, concepts...). Moreover, the management services allow users to build their own hypertext network.
With regard to model generation, our system and Terminae [2] provide complementary services. Terminae resort to the terminologist to provide a very precise

description of the terms from which a precise formal representation, in description logic, can be generated. In our approach, the system does not require users to provide additional descriptions before performing model generation from term hierarchies. Model generation strictly and thoroughly concentrates on hierarchical structures that can be detected at the linguistic level using term extraction techniques. For example, the hierarchical relation between the terms "*Flight Plan*" and "*Modified Flight Plan*" is identified by XTERM because of the explicit relations that hold between the linguistic structures of the two terms. Hence, such term hierarchies can be exploited for class generation. However, XTERM would be unable to identify the hierarchical relation that holds between the terms "*vehicle*" and "*car*" (which is the kind of relations that Terminae would try to identify in the formal descriptions). As a consequence, the formal description provided by our system is mainly a hierarchy of concepts while that of Terminae is more structural and the subsumption relations is computed by the description logic system.

The transition from informal to formal models is also addressed in [16]. The approach allows users to express the knowledge informally (within texts and hypertexts) and more formally (through semantic networks coupled with an argumentation system). In this modeling framework, knowledge becomes progressively more formal through small increments. The system, called "Hyper-object substrate", provides an active support to users by suggesting formal descriptions of terms. Its integrated nature enables to make suggestions while the users are manipulating the text, and to take advantage of already formalized knowledge to deduce new formalization steps. Our system, whose linguistic processing component is far more developed, could be coherently embedded in this comprehensive modeling framework.

5 Conclusion

We have presented a fully implemented system that produces class hierarchies out of textual documents, taking advantage of term hierarchies automatically built with natural language processing techniques. This system, by integrating document, terminology and knowledge management, provides traceability links through technical terms.

The system is robust but generates only taxonomies. Further work will address the automatic generation of more complex knowledge structures such as attributes and relations between classes.

This work has considered source documents with a low degree of formality: text in paragraphs. Further investigation will adress the problem of link generation from semi-structured sources. Link generation might significantly be improved when the sources are semi-structured. In particular, XML (and SGML) tagging provides useful information about the content structure that allows to accurately identify the potential link anchors.

Acknowledgments

This work has been partially realized in the GENIE II program supported by the French ministry of education, research and technology (MENRT) and the DGA/SPAé.

References

1. N. Aussenac-Gilles, D. Bourigault, A. Condamines, and C. Gros. How can knowledge acquisition benefit from terminology ? In *Proceedings of the 9th Knowledge Acquisition for Knowledge Based System Workshop (KAW '95)*, Banff, Canada, 1995.
2. B. Biébow and S. Szulman. Une approche terminologique pour la construction d'ontologie de domaine à partir de textes : TERMINAE. In *Proceedings of 12th RFIA Conference*, pages 81–90, Paris, 2000.
3. D. Bourigault. Lexter, a terminology extraction software for knowledge acquisition from texts. In *Proceedings of the 9th Knowledge Acquisition for Knowledge Based System Workshop (KAW '95)*, Banff, Canada, 1995.
4. F. Cerbah. Acquisition de ressources terminologiques – description technique des composants d'ingénierie linguistique. Technical report, Dassault Aviation, 1999.
5. K. W. Church and P. Hanks. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.
6. B. Daille. Study and implementation of combined techniques for automatic extraction of terminology. In J.L. Klavans and P. Resnik, editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. MIT Press, Cambridge, 1996.
7. J. Euzenat. XML est-il le langage de représentation de connaissance de l'an 2000 ? In *Actes des 6ème journées langages et modèles à objets*, pages 59–74, Mont Saint-Hilaire, CA, 2000.
8. B. Gaines and M. Shaw. Documents as expert systems. In Cambridge University Press, editor, *Proceedings of 9th British society expert systems conference*, pages 331–349, 1992.
9. J. S. Justeson and S. M. Katz. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27, 1995.
10. O. Mariño, F. Rechenmann, and P. Uvietta. Multiple perspectives and classification mechanim in object-oriented representation. In *Proceeding of 9th ECAI*, pages 425–430, Stockholm, 1990.
11. P. Martin. *Exploitation de graphes conceptuels et de documents structurés et hypertextes pour l'acquisition de connaissances et la recherche d'information.* PhD thesis, Université de Nice-Sophia Antipolis, 1996.
12. OMG. XML Metadata Interchange (XMI). Technical report, OMG, 1998.
13. D. Petitpierre and G. Russell. MMORPH – The Multext Morphology Program. Technical report, Multext Deliverable 2.3.1, 1995.
14. F. Rechenmann. Building and sharing large knowledge bases in molecular genetics. In *Proceedings of 1st International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, pages 291–301, Tokyo, 1993.
15. Projet Sherpa. Troeps 1.2 reference manual. Technical report, Inria, 1998.
16. F. Shipman and R. McCall. Supporting incremental formalization with the hyper-object substrate. *ACM Transactions on information systems*, 17(2):199–227, 1999.