Traceability between Models and Texts through Terminology

Farid Cerbah [a], Jérôme Euzenat [b]

[a] *Dassault Aviation,* DPR/ESA,
78, quai Marcel Dassault, 92552 cedex 300 Saint-Cloud, France
[b] *INRIA Rhône-Alpes,*
655, avenue de l'Europe, 38330 Monbonnot St Martin, France

Abstract

Modeling often concerns the translation of informal texts into formal representations. This translation process requires support for itself and for its traceability. We pretend that inserting a terminology between informal textual documents and their formalization can help to serve both of these goals. Modern terminology extraction tools support the formalization process by using terms as a first sketch of formalized concepts. Moreover, the terms can be employed for linking the concepts and the textual sources. They act as a powerful navigation structure. This is exemplified through the presentation of a fully implemented system.

Key words: Terminology extraction, Traceability, Model generation, Hypertext, Object-oriented modeling, Natural language.

1 Introduction

The modeling activity 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. It is recognized that linking formal representations to their informal textual counterparts has several benefits including, establishing the context for formal structures and providing a natural way to browse through knowledge repositories. In one word, linking the formal and the informal is an enabling technology for traceability.

Email addresses: farid.cerbah@dassault-aviation.fr (Farid Cerbah), Jerome.Euzenat@inrialpes.fr (Jérôme Euzenat).

The benefits of integrating textual documents and formal representations in common repositories can be pointed out more specifically in three types of modern information systems:

Product data management. Design and production data are formalized in product trees. This formalization improves data consistency and evolutivity. The components of the product tree are related to documents, such as maintenance manuals or manufacturing notices. Connecting formal models to informal sources guarantees a better synchronization between technical data and documents.

Software engineering. Formal (or semi-formal) software models, and more particularly object oriented ones, are now widely used in software development. In this context, textual knowledge represents specification and design documents. These informal sources are used as a basis for building formal models. In an object-oriented framework, many traceability links aim at relating textual document fragments in natural language and model fragments.These links should enable to find out the requirements impacted, directly or indirectly, by the (re)design decisions.

Knowledge management. In the context of generalized knowledge management, traceability of elaborated knowledge from raw text provides both grounding and arguments for decisions. It is thus necessary to link the source documentation to the formal knowledge. In this area, several attempts have been made to provide tools supporting the linking of knowledge sources (1; 2; 3; 4).

Many previous works focused on the advantages of using corpus-based term extraction for supporting formal knowledge acquisition (5; 6; 7). These contributions emphasize the central role of terminological resources in the mapping between formal knowledge structures and the textual sources. In the same spirit, the present work demonstrates how an active support for traceability can be provided. We put forth an architecture, centered around a term extraction and management tool, for the generation and the management of traceability links between textual documents and formal object representations resulting from the modeling processes. It has been fully implemented with existing software and provides high-level hypertext generation, browsing and model generation facilities.

The rest of the paper is organized as follows. Section 2 introduces the main concepts of our approach and the basic tasks that should be performed by a user support tool which exploits terminological knowledge for improving traceability. Section 3 gives a detailed and illustrated description of the implemented system. , Section 4 compares our contribution to related works and we conclude with several directions for further work.

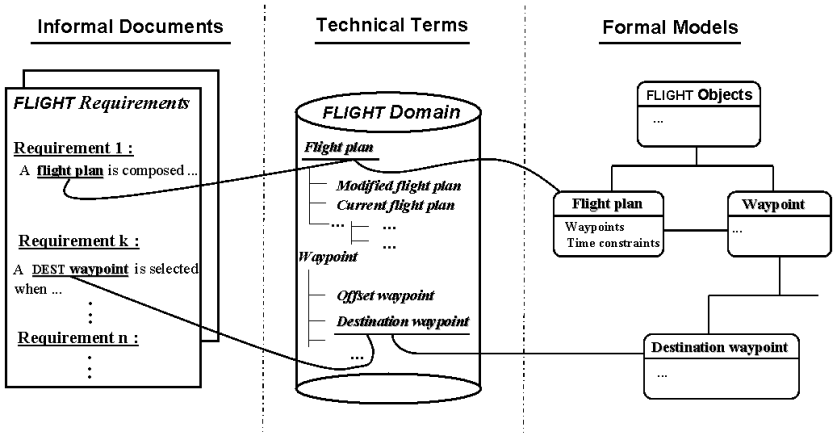

Fig. 1. Using terminological items to link textual requirements and object models.

2 An architecture for traceability through terminological knowledge

2.1 Tracing decisions in software development

In a software development process, design and implementation decisions should be "traceable", in the sense that it should be possible to find out the requirements impacted by the decisions. This mapping is useful in many respects:

- It helps to ensure completeness: By following traceability links, the user or a program can easily identify the requirements which are not satisfied by the software.
- It facilitates the propagation of changes: At any time in the development process, traceability information allows to find out the elements impacted by changes (upstream and downstream). For instance, the user can evaluate the incidence on the software design and implementation of a late change in the initial customer requirements.
- When traceability is established with hyperlinks, the browsing capabilities provided by design support tools are increased.

In an object-oriented framework, many traceability links aim at relating textual fragments of the documents in natural language and model fragments. Putting on these links manually is a tedious and time consuming task and current tools for requirement analysis provide no significant help for doing that job.

2.2 The role of terminological resources

As mentioned earlier, previous works in the fields of knowledge acquisition and natural language processing have shown that terminological resources extracted from corpora can help the incremental formalization processes from texts to formal models. One can find hints of these ideas in related domains:

- In the DOCSTEP project (8), which deals with product data management, terminological resources are used to connect multilingual technical documentation and items of product trees. Hyperlinks are established between term occurrences in documents and corresponding objects in product trees.
- In software engineering, the role of terminological knowledge in the modeling process has often been pointed out (9; 10; 11). One of the first step in the modeling process consists of a systematic identification of the technical terms (simple and compound nouns) in the documents, namely the terminology used to describe the problem.

It appears that, in many information systems where both textual documents and formal representations are involved to describe related concepts, a terminology can play a bridging 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).

One can think that the terminological resources are useful when constructing the models but not for ensuring traceability: once the units in the models are linked to the corresponding text fragments, the terms are no longer required. There are, however, several reasons in favor of preserving the terminological structures. First, even though it is a common practice in ontology building, the terms as linguistic units should not be identified with the concepts which are introduced to serve in well defined formal reasoning processes. A linguistic level is still necessary because of the inherent complexity of the terms and their relations. More particularly, this is where synonymy and graphical variation phenomena are handled. The synonymous terms that are merged into one referent concept can be useful for indexing and navigating through the documents and the models. On the matter of traceability, the terms that have been extracted but not selected for concept or class generation are design decisions that must be recorded for traceability.

2.3 Functional view

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 multi-word 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 (5; 12) or statistical processing (13; 14; 15) (or even both paradigms (16)). Terminology extraction techniques are used to automatically build term hierarchies.

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 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 descriptors is the basis of the hyperlink generation mechanisms. In this approach, The terminological structure is the cornerstone of the hypertext navigation capabilities.

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.

These four automated functions should be controlled by the user through selection and validation actions applied to the various potential knowledge units that have been automatically identified or constructed. The user has to refine the result of the term extractor in order to identify and validate the terms which are relevant with regard to the modeling task at hand. 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.

To sum up, the integration of these functions within a single process results in a method for helping the acquisition and maintenance of formal representations from textual documents.

3 A user support tool for improving traceability

To implement the functions presented above, we have used the existing components XTERM (§3.1) and TROEPS (§3.2) and have integrated them into an architecture that uses XML representations for exchanging structured data (§3.3). XTERM deals with document management and linguistic processing functions, more particularly terminological extraction and document indexing. TROEPS deals with knowledge management and model indexing. The model generation function is spread over both components.

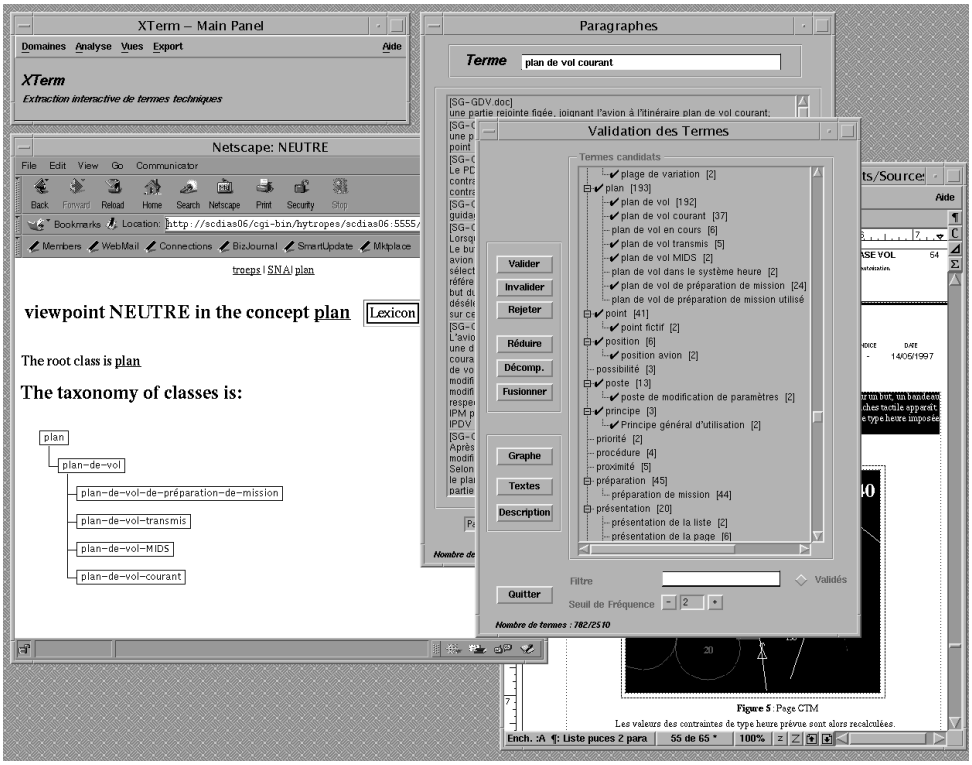

Fig. 2. The integrated system based on XTERM and TROEPS.

3.1 Terminology extraction with XTERM

XTERM 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 tagging. We used a rule based tagger built upon MMORPH (17) which is a morphological parser developed in the MULTEXT project. The MMORPH parser assigns to each word its possible morphological descriptions by looking up in lexicons and applying morphological decomposition rules. Each possible word analysis is expressed as a combination of attribute-value pairs. Then, contextual disambiguation rules are applied to choose a unique description for each word. At the end of this process, each word is unambiguously tagged.

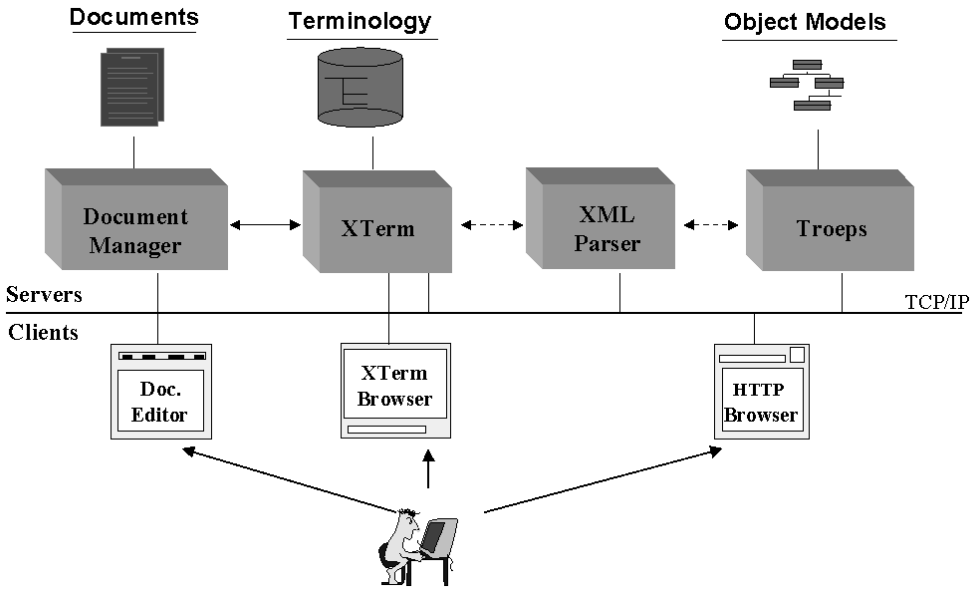

Fig. 3. System architecture

For example, the sentence "*les plans de vol restent affichés* [Lit. *The flight plans remain displayed*]." is parsed in the following way:

```
"Les"       [Lemma=le,Cat=art,Gender=masc,Num=plur,Quant=def]
"plans"     [Lemma=plan,Cat=noun,Gender=masc,Num=plur]
"de"        [Lemma=de,Cat=prep]
"vol"       [Lemma=vol,Cat=noun,Gender=masc,Num=sing]
"restent"   [Lemma=rester,Cat=Verb,Tense=present,Mood=ind,Num=plur]
"affichés"  [Lemma=afficher,Cat=Verb,Tense=past,Mood=part,Num=plur]
```

As mentioned in section 2.3, the morpho-syntactical 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. Figure 4 gives an example of a simplified automaton (state 2 is the unique final state). 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.

Actually, term extraction with automata is just the first filtering step of the overall process. The candidate set obtained after this step is still too large. Additional filtering mechanisms are involved to reduce it, such as grouping rules that identify term variants. For example, in French technical texts, prepositions and articles are often omitted for sake of concision. The term "*page des*

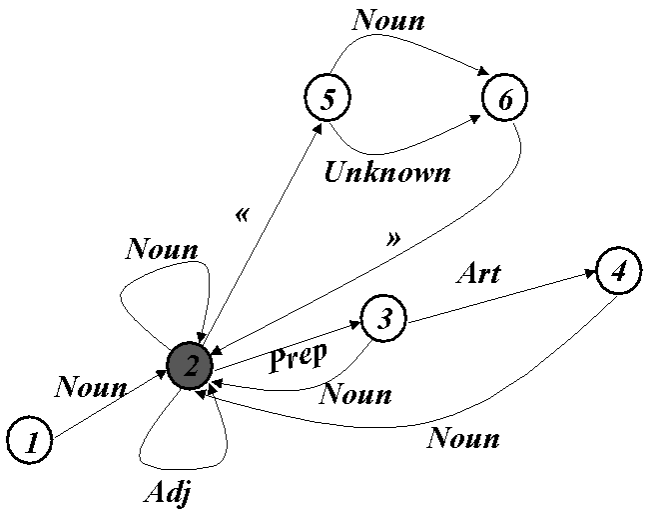

Fig. 4. A term extraction automaton.

buts" ("*page of the waypoints*") can occur in the elided form "*page buts*" ("*waypoint page*"). Term variants are systematically conflated into a single node in the term hierarchy.

Additionally, XTERM provides the mechanisms for indexing and hyperlink generation 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 Concept modeling with the TROEPS *system*

TROEPS (18; 19) 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

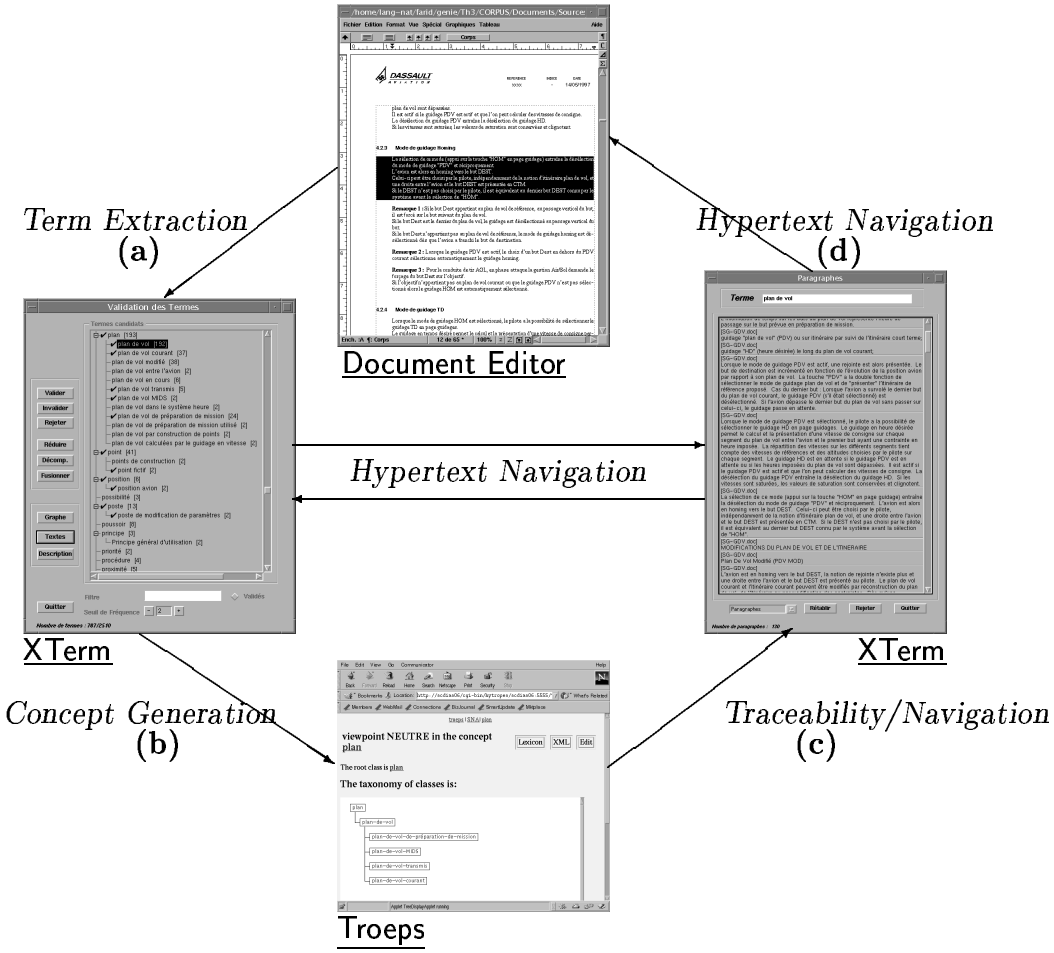

Fig. 5. Functional interactions between XTERM and TROEPS.

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 (in input and output). This interface can describe a whole knowledge base or can apply specific operations on an existing knowledge base (such as adding a new class in a hierarchy, classifying or destroying an object or sticking an annotation to an existing structure).

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).

Figure 5 illustrates a typical use scenario of the overall system. Terminology extraction (fig. 5–a) builds a term hierarchy containing a subtree rooted in the term "*Plan*". Class generation (fig. 5–b) can then be applied in order to construct a class hierarchy that 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 term-centered navigation capabilities offered by XTERM are directly available from the TROEPS interface. The TROEPS user has access to the multi-document view of the paragraphs where the term "*flight plan*" and its variants occur (fig. 5–c). From this view, the user can consult the source documents if required (fig. 5–d).

On a more technical ground, XTERM sends an XML stream to an XML parser which, in turn, invokes TROEPS as an HTTP server (see figure 3). 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 class and the annotation of a newly created class with textual elements such as the outlined definition of the term naming the class. For example, to generate classes from the term hierarchy rooted at the term "*plan*", XTERM sends to TROEPS an XML stream containing a sequence of class creation and annotation statements. The following XML fragment corresponds to the creation of classes "*Flight-Plan*" and "*Current-Flight-Plan*":

```
<trp:ADD>
  <trp:CLASS>
    <trp:CLASSDSC name="Flight-Plan">
      <trp:CLASSREF name="Plan"/>
    </trp:CLASSDSC>
  </trp:CLASS>
</trp:ADD>
<trp:ADD>
  <trp:CLASS>
    <trp:CLASSDSC name="Current-Flight-Plan">
      <trp:CLASSREF name="Flight-Plan"/>
    </trp:CLASSDSC>
  </trp:CLASS>
</trp:ADD>
<trp:ANNOTATE label="comment">
  <trp:CLASSREF name="Flight-Plan"/>
    <trp:CONTENT>
        A flight plan is a sequence of waypoints...
    </trp:CONTENT>
</trp:ANNOTATE>
```

The TROEPS 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 (20)) which share the notion of classes and objects).

More details about this approach of XML-based knowledge modeling and exchange can be found in (21).

Concerning the implementation of the links, there could be several possible ways to connect the concepts to the terms. The one used in the current system consists for TROEPS of invoking XTERM as an HTTP server with a URL and the concept name as parameter (considered as an implicit link). XTERM will then take care of finding the corresponding concepts thanks to its indexing capabilities. Another implementation consists in communicating to TROEPS the term to pass back to XTERM in order to activate it when the user wants to access the terminology (this is an explicit link). Both approaches can be implemented in the current framework. In fact the best option would be to generate a key that can be used by XTERM for efficiently retrieving the term (this can be useful for retrieving homonymous terms or terms corresponding to concepts whose name has been changed on the TROEPS side).

4 Related work

Terminology acquisition is one of the most robust language processing technology and previous works have demonstrated that term extraction tools can help to link informal texts and formal structures. The theoretical apparatus depicted in (5; 6; 7) provides useful guidelines for integrating term 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 (7) provide complementary services. Terminae helps the terminologist to build a precise description of the terms from which a 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 instance, the hierarchical relation between the terms "*Flight Plan*" and "*Current 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 "*aircraft*" and "*fighter*" (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.

In the field of software engineering, object-oriented methods concentrate on the definition of formal or semi-formal formalisms, with little consideration for the informal-to-formal processes (9; 10; 11). However, to identify the relevant requirements and model fragments, designers should perform a deep analysis of the textual specifications. The recommendations discussed in section 2.2 on the use of terminological resources can be seen as a first step.

The transition from informal to formal models is also addressed in (22). 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 "Hyperobject substrate", provides an active support to users by suggesting formal descriptions of terms. Its integrated nature allows 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.

Our work is also related to the WEB→KB system (23) whose goal is to automatically build large knowledge bases by analyzing the World Wide Web. The system starts with a predefined domain model, composed of classes and relations between them. Potential instances are identified on the Web using machine learning techniques. "Informal instances" of predefined classes and relations may correspond to Web pages, hyperlinks or text fragments. Our approach concentrates on the extraction of model fragments whereas this work focuses on instance identification. No linguistic processing is involved in this system. Textual material is simply viewed as bag of words (without stemming or lemmatization). However, some learning techniques developed in this context could be adapted for model generation. This also shows that our system can be used for querying a corpus of documents through its formalized content.

ClearType (4) is a subset of English grammar and lexicon with a very precise interpretation. It enables to perform knowledge acquisition form a controlled

natural language fragment and can be used for incremental formalization (22). This approach constrains the writers to adopt that language but enables a completely automatic generation of the concepts. The approach presented here does not constraint the writing but requires the user intervention for selecting the relevant concepts.

5 Conclusion

Translating from informal to formal is a common task of knowledge acquisition or requirement elicitation and providing traceability information is now a major requirement for any kind of system design.

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

The system has been presented to potential users and, particularly, to software designers in aeronautics. Their feedback showed that the approach is consistent with actual specification and design practices.

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. To that extent several kind of work can be undertaken:

- Improving knowledge generation by automatically detecting potential attributes and their types (the same could be possible for events, actions...);
- Adding new capabilities for combining the compound term structuring with a "primitive term" structuring. There are two non exclusive tracks for doing so: introducing semantic definitions in the lexicon used by XTERM so that the system can be aware that a a term like "*fighter*" denotes a kind of aircraft and should introduced under the aircraft taxonomy, or using a system like Terminae for annotating terms with their formal definitions and deducing the hierarchical relations. Implementing definition detection in texts could be another track.
- Using text structure for a better indexing. The present work has considered source documents with a low degree of formality: roughly, text structured in paragraphs. Further investigation will address the problem of link generation from semi-structured sources. Link generation might be significantly 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] B. Gaines, M. Shaw, Documents as expert systems, in: C. U. Press (Ed.), Proceedings of 9th British society expert systems conference, 1992, pp. 331–349.

[2] 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, Tokyo, 1993, pp. 291–301.

[3] P. Martin, Exploitation de graphes conceptuels et de documents structurés et hypertextes pour l'acquisition de connaissances et la recherche d'information, Ph.D. thesis, Université de Nice-Sophia Antipolis (1996).

[4] D. Skuce, T. Lethbridge, CODE4: a unified system for managing conceptual knowledge, International Journal of Human-Computer Studies 42 (4).

[5] 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.

[6] N. Aussenac-Gilles, D. Bourigault, A. Condamines, 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.

[7] B. Biébow, S. Szulman, Une approche terminologique pour la construction d'ontologie de domaine à partir de textes : TERMINAE, in: Proceedings of 12th RFIA Conference, Paris, 2000, pp. 81–90.

[8] K. Elavaino, J. Kunz, Docstep — technical documentation creation and management using step, in: Proceedings of SGML '97, 1997.

[9] J. Rumbaugh, Object-Oriented Modeling and Design, Prentice-Hall, 1991.

[10] I. Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, 1992.

[11] G. Booch, Object-Oriented Analysis and Design with Applications, 2nd Edition, Addison-Wesley, 1994.

[12] J. S. Justeson, S. M. Katz, Technical terminology: Some linguistic properties and an algorithm for identification in text, Natural Language Engineering 1 (1) (1995) 9–27.
[13] I. Dagan, K. Church, *Termight*: Identifying and Translating Technical Terminology, in: Proceedings of 4th Conference on Applied Natural Language Processing (ANLP'94), Stuttgart, 1994, pp. 34–40.
[14] C. Enguehard, L. Pantéra, Automatic Natural Acquisition of a Terminology, Journal of Quantitative Linguistics 2 (1) (1994) 27–32.
[15] K. T. Frantzi, S. Ananiadou, The C-Value/NC-Value domain independent method for multi-word term extraction, Journal of Natural Language Processing 6 (3) (1999) 145–179.
[16] B. Daille, Study and implementation of combined techniques for automatic extraction of terminology, in: J. Klavans, P. Resnik (Eds.), The Balancing Act: Combining Symbolic and Statistical Approaches to Language, MIT Press, Cambridge, 1996.
[17] D. Petitpierre, G. Russell, MMORPH – The Multext Morphology Program, Tech. rep., Multext Deliverable 2.3.1 (1995).
[18] O. Mariño, F. Rechenmann, P. Uvietta, Multiple perspectives and classification mechanism in object-oriented representation, in: Proceeding of 9th ECAI, Stockholm, 1990, pp. 425–430.
[19] Projet Sherpa, Troeps 1.2 reference manual, Tech. rep., Inria (1998).
[20] OMG, XML Metadata Interchange (XMI), Tech. rep., OMG (1998).
[21] J. Euzenat, XML est-il le langage de représentation de connaissance de l'an 2000 ?, in: Actes des 6eme journées langages et modèles à objets, Mont Saint-Hilaire, CA, 2000, pp. 59–74.
[22] F. Shipman, R. McCall, Supporting incremental formalization with the hyper-object substrate, ACM Transactions on information systems 17 (2) (1999) 199–227.
[23] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery, Learning to construct knowledge bases from the World wide Web, Artificial Intelligence, Special Issue on Intelligent Internet Systems 118 (1-2) (2000) 69–113.

Dr **Farid Cerbah** is researcher at Dassault Aviation (Paris, France). He has worked on various areas of computational linguistics, including natural language generation, computational terminology and machine learning applied to language processing. His current research focuses on terminology acquisition and classification techniques based on partial parsing and statistical methods. His research interests also include the use of linguistics resources for enriching hypertext documents.

Dr **Jérôme Euzenat** is researcher at INRIA Rhône-Alpes (Montbonnot, France). He has worked and published about reason maintenance systems, object-based knowledge representation, temporal granularity and knowledge-base cooperative editing. His all time interests are tied to the relations holding between various representations of the same situation. This is applied to work on property-preserving transformations of representations for an accurate communication.