Integrating Textual Knowledge and Formal Knowledge for Improving Traceability

Farid Cerbah[1] and **Jérôme Euzenat**[2]

Abstract. This article deals with traceability in knowledge repositories. More precisely, we concentrate on the role of terminological knowledge in the mapping between (informal) textual requirements and (formal) object models. We show that terminological knowledge facilitates the production of traceability links and model generation, provided that language processing technologies allow to elaborate semi-automatically the required terminological resources. The presented system is one step towards incremental formalization from textual knowledge. As such, it is a valuable tool for building knowledge repositories.

1 INTRODUCTION

Knowledge management has for long been preoccupied by the relationships between formal and informal knowledge. The informal is richer and familiar to any user while the formal is more precise and necessary to the computer. It is recognized that linking formal knowledge to informal knowledge has several benefits in the context of knowledge management including, (1) establishing the context for formalized knowledge and documenting it, and (2) providing a natural way to browse through formalized knowledge. A software tool for supporting link generation, like the one presented in this paper, is an opportunity to kick off incremental, corpus-driven formalization.

In the field of knowledge management, there have been attempts to provide tools supporting the linking of knowledge sources [11, 15, 18]. However, the computational support provided was quite limited. The links had to be established manually and thus were error-prone and time consuming (not only the initial setting of the links but, above all, the updating operations). Besides, the browsing capabilities from formal knowledge to the informal documents were minimal (e.g., the hyperlinks had only one target document). In the meantime, several works focused on the advantages of using a corpus-based terminology for supporting formal knowledge acquisition [4, 1, 2]. These contributions emphasize the central role of terminological resources in the mapping between informal text sources and formal knowledge bases. We put forth an architecture, centered around a terminology extration and management tool, which enables to generate models from texts and navigate from one to another through the terminology. We describe a fully implemented system that provides high-level hypertext generation, browsing and model generation facilities. From a more technical viewpoint, we introduce an original XML based model for integrating software components.

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. Finally, section 4 briefly compares our contribution to related works and the conclusion provides some directions for further research.

2 PRINCIPLES

2.1 Traceability in software engineering and knowledge repositories

In software engineering, it is often stressed that design and implementation decisions should be "traceable", in the sense that it should be possible to find out the requirements impacted, directly or indirectly, by the decisions. In a similar way, 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. This mapping is useful in many respects:

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

Moreover, in the context of generalized knowledge management, traceability of elaborated knowledge from raw text provides both grounding and arguments for decisions.

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

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

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

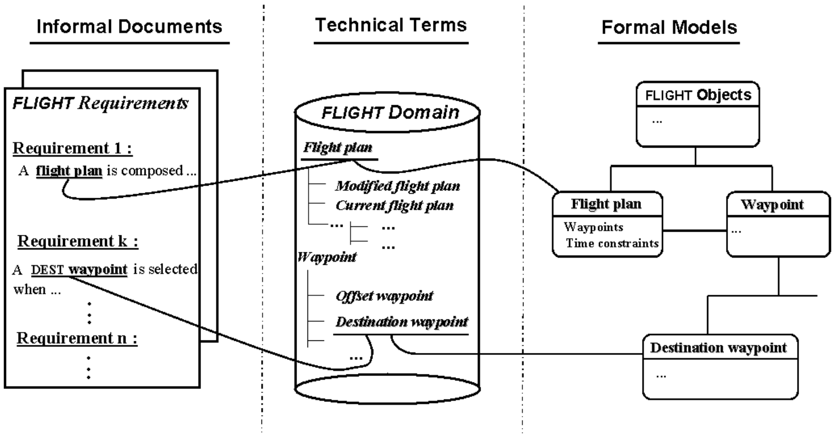


Figure 1. Using terminological items to link textual requirements and object models

is a tedious and time consuming task and current tools for requirement analysis or knowledge acquisition provide no significant help for doing that job (except [15]).

2.2 The role of terminological resources

In many information systems where both textual knowledge and formal knowledge are involved to describe related concepts, terminology can play an intermediate role. 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 in the incremental formalization processes from texts to formal models. There exists other demonstrative examples in related domains, such as product data management and software engineering.

For example, in the DOCSTEP project [9], 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 [19, 12, 3]. One of the first step in the modeling process consists in a systematic identification of the technical terms (simple and compound nouns) in the documents, namely the terminology used to describe the problem. Some of these technical terms represent concepts which will be subsequently introduced in the formal models. These terms can be seen as an intermediary level between the textual requirements and the formal models. (see figure 1).

2.3 Functional view of a system that exploits terminology

A system that takes advantage of terminological resources may involve techniques pertaining to several technological areas, and particularly natural language processing, information retrieval and knowledge management:

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 [4], [13] or statistical processing [6] (or even both paradigms [8]). 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 ...).

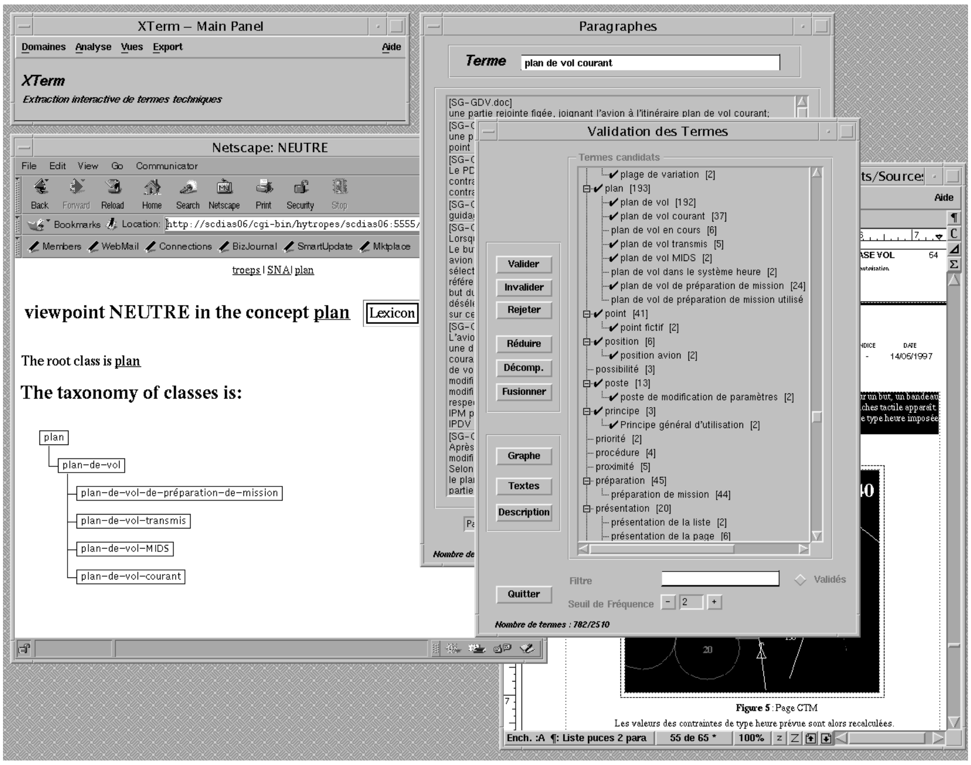
Hyperlink Generation. The terminology driven indexing of both texts and models with the same terminology is the basis of the hyperlink generation mechanisms. Futhermore, hyperlink generation should be controlled interactively, in the sense that the user should be able to exclude automatically generated 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 completed manually.

These features are implemented in the system presented in the next section.

3 A USER SUPPORT TOOL FOR IMPROVING TRACEABILITY

The implemented system consists of two components, XTerm and Troeps. XTerm deals with the document management and linguistic processing functions, more particularly terminological extraction and the document indexing. Troeps



XTerm – Main Panel
Domaines Analyse Vues Export Aide
XTerm
Extraction interactive de termes techniques
Netscape: NEUTRE
File Edit View Go Communicator
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location:
Members WebMail Connections BizJournal SmartUpdate Mktplace
troeps | SNA| plan
Lexicon
Paragraphes
Validation des Termes
Termes candidats
Valider
Invalider
Rejeter
Réduire
Décomp.
Fusionner
Graphe
Textes
Description
Quitter
Validés
Figure 5 : Page CTM
Ench. :A ¶: Liste puces 2 para
55 de 65 *
100%

Location: http://scdias06/cgi-bin/hytropes/scdias06:5555/

viewpoint NEUTRE in the concept plan

The root class is plan

The taxonomy of classes is:

```
plan
└plan-de-vol
  ├plan-de-vol-de-préparation-de-mission
  ├plan-de-vol-transmis
  ├plan-de-vol-MIDS
  └plan-de-vol-courant
```

Terme: plan de vol courant

[SG-GDV.doc]
une partie rejointe figée, joignant l'avion à l'itinéraire plan de vol courant;

```
  └✔ plage de variation [2]
⊟✔ plan [193]
  ├✔ plan de vol [192]
  ├✔ plan de vol courant [37]
  ├ plan de vol en cours [6]
  ├✔ plan de vol transmis [5]
  ├✔ plan de vol MIDS [2]
  ├ plan de vol dans le système heure [2]
  ├✔ plan de vol de préparation de mission [24]
  └ plan de vol de préparation de mission utilisé
⊟✔ point [41]
  └✔ point fictif [2]
⊟✔ position [6]
  └✔ position avion [2]
├ possibilité [3]
⊟✔ poste [13]
  └✔ poste de modification de paramètres [2]
⊟✔ principe [3]
  └✔ Principe général d'utilisation [2]
├ priorité [2]
├ procédure [4]
├ proximité [5]
⊟ préparation [45]
  └ préparation de mission [44]
⊟ présentation [20]
  ├ présentation de la liste [2]
  ├ présentation de la page [6]
```

Filtre:

Seuil de Fréquence: 2

Nombre de termes : 782/2510

Les valeurs des contraintes de type heure prévue sont alors recalculées.

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

deals with knowledge management and model indexing. The model generation function is spread over both components.

3.1 Terminology extraction with XTerm

XTerm [5] is a natural language processing tool that provides two services to end users:

- Terminology acquisition from documents. It analyzes a French or English technical documentation in order to build a hierarchy of potential technical terms. The user can explore and filter the extracted data via a graphical interface.
- Terminology-centred hypertext navigation. XTerm can be seen as a hypertext browser. The extracted terms are systematically linked to their textual contexts in the documents. The user can easily access the textual fragments containing term occurrences.

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. Additionally, it 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.

The first linguistic processing step is POS tagging. We used a rule based tagger based on the Multex morphological parser [17]. POS tagging starts with morphological analysis which assigns to each word its possible morphological realizations. Then, contextual desambiguation rules are applied to choose a unique realization for each word. At the end of this process, each word is unambigeously tagged.

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 transition crossing conditions are expressed in terms of morphological properties. To identify the potential terms, the automata are applied on the tagged word sequences provided by the POS tagger. 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 that set. Grouping rules are used to identify term variants. For instance, in French technical texts, prepositions and articles are often omitted for the sake of concision (the term "*page des buts*" can occur in the elided form: "*page buts*)"[3]. Term variants are systematically conflated into a single node in the term

[3] Whose English literal translations are respectively: "*page of the waypoints*" and "*page waypoints*". A plausible equivalent term in English could be "*Waypoint page*".

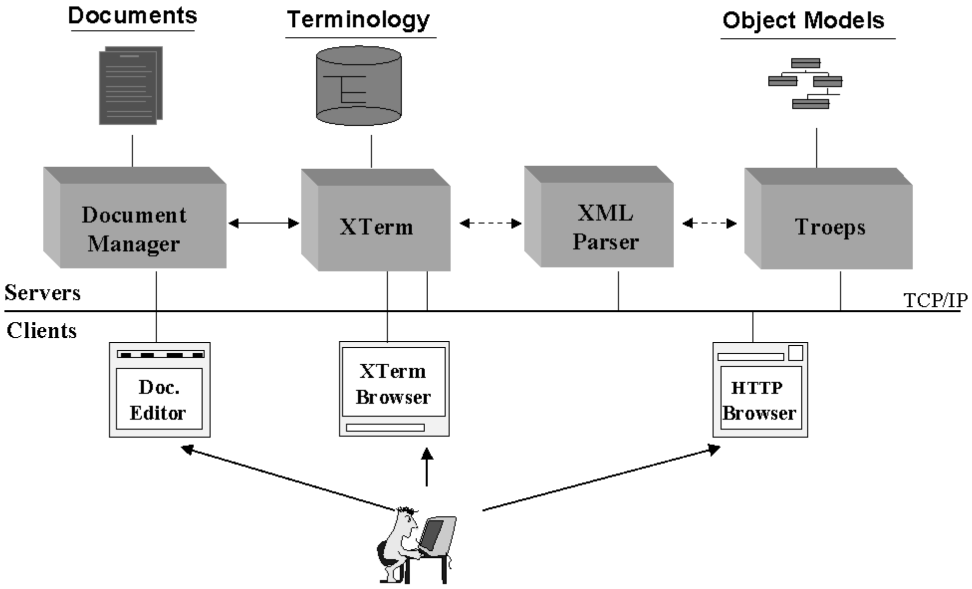


Figure 3. System architecture

hierarchy.

XTerm is highly interactive. Many browsing facilities are provided to facilitate the manipulation of large data sets (extracted terms + text fragments). XTerm can be used as an access tool to documentation repositories.

3.2 Knowledge modeling with the Troeps system

Troeps [14, 21] 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 determines the key and structure of its 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.

Objects can be seen under several viewpoints, each corresponding to a different taxonomy. An object can be attached to a different class in each viewpoint. For instance, a particular plan is classified as a "*flight plan*" under the nature viewpoint and as a "*logistic plan*" under the functional viewpoint. This is unlike other object systems, which usually allow only one class hierarchy.

Object-based knowledge representation provides various facilities for manipulating knowledge among which filtering queries (which find objects of a concept satisfying fields 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 whose skeleton is the structure of formal knowledge (mainly in the object-based formalism) and whose flesh consists of pieces of texts, images, sounds and videos tied to the objects. Turning a knowledge base into a HTTP server is easily achieved by connecting it to a port and transforming each object reference into an URL and each object into a HTML page. If HTML pages already document the knowledge base, they remain linked to or integrated into the pages corresponding to the objects. The Troeps user (through an Application Programming Interface) can explicitly manipulate each of the Troeps entities. The entities can also be displayed on a HTTP client through their own HTML page. The Troeps program generates all the pages on demand (i.e. when a URL comes through HTTP). The pages make numerous references to each others. They also display various documentation (among which other HTML pages and lexicon) and give access to Troeps features. From a Troeps knowledge server it is possible to build complex queries grounded on formal knowledge such as filtering or classification queries. The answer will be given through a semantically sound method instead of using a simple full-text search. Moreover, it is possible to edit the knowledge base. The system presented here takes advantage of this last feature.

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

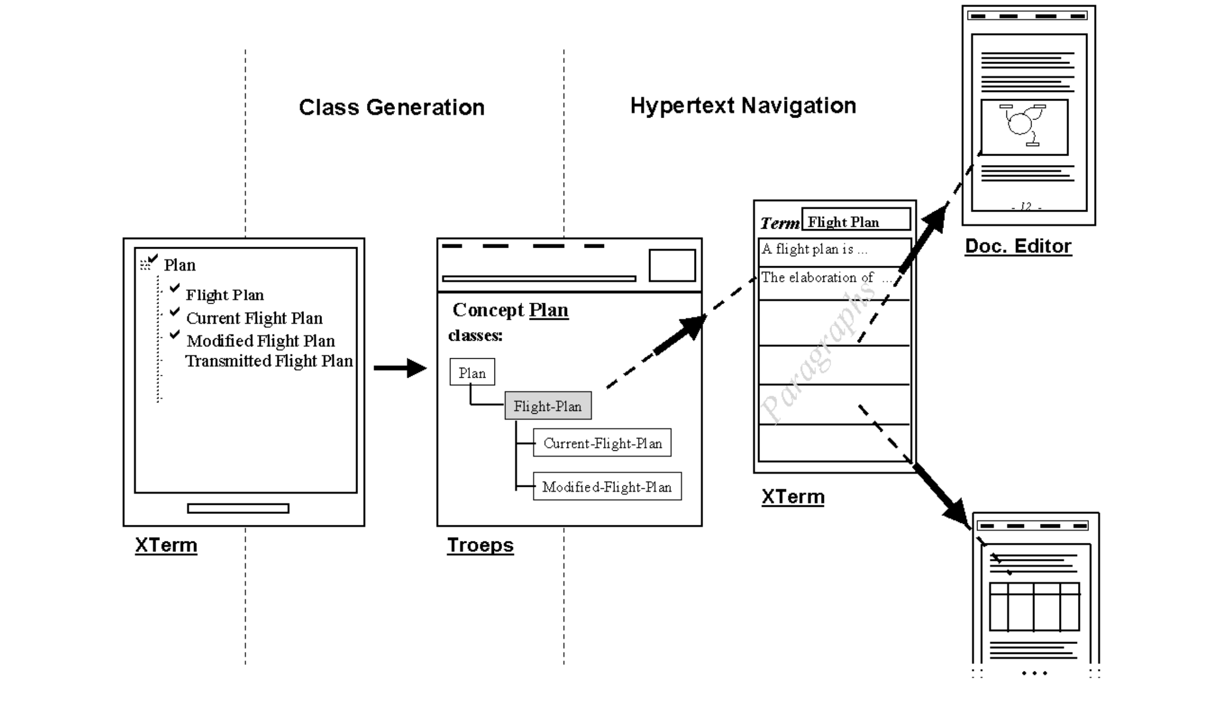


Figure 4. Class generation and traceability through hyperlinks

from term hierarchies. Conversely, Troeps can call XTerm to provide the textual fragments related to a concept (via a technical term).

For example, figure 4 illustrates the class generation process from a hierarchy of terms carefully validated by the user (a hierarchy rooted in the term "*Plan*"). The class hierarchy constructed by Troeps mirrors the hierarchy of the validated terms (under the root "*Plan*").

At the end of the generation process, the created classes are still linked to their corresponding terms, which means that the terminology-centred navigation capabilities offered by XTerm are directly available from the Troeps interface. As illustrated by figure 4, the Troeps user has access to the multi-document view of the paragraphs which concern the "*Flight-Plan*" concepts[4]. From this view, the user can consult the source documents if required.

Data exchanges between XTerm and Troeps are based on the XML language (see figure 3). Troeps offers an XML interface which allows to describe a whole knowledge base or to take punctual actions on an existing knowledge base. This last feature is used in the interface where XTerm sends to Troeps short XML statements corresponding to the action performed by the user. These actions correspond to the 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. XML representation of object models . We give below an extract of this sequence, corresponding 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 term definition filled out in the XTerm description of the term is added as a textual annotation in the class description. After these automated steps, the classes can be completed manually.

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 standard in the definition of formalized knowledge so that it will be easy to have XTerm generating other formats (e.g. XMI [16] or Ontolingua) which share the notion of classes and objects.

More details about this approach of XML-based knowledge modeling and exchange are given in [10].

[4] More precisely, this view displays the paragraphs where the term "*flight plan*" and its variants occur.

4 RELATED WORK

Terminology acquisition is one of the most robust language processing technology [4, 13, 8] and previous works have demonstrated that term extraction tools can help to link informal and formal knowledge. The theoretical apparatus depicted in [4], [1] and [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 hold 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.

A recent contribution in the field of knowledge management is that of [20] which provides automatic indexing of mail messages in a corporate context. However, the indexing mechanisms do not involve terminological resources.

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 [19, 12, 3]. 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 "Hyper-Objet substrate", provides an active support to users by suggesting formal descriptions of terms. The integrated nature of this system allows to make suggestions while the users are manipulating the text, and to exploit 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 [7] 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). However, some learning techniques developed in this context could be adapted for model generation.

5 CONCLUSION

Structured knowledge repositories are by nature highly relational and the various relations that hold between knowledge fragments are often expressed through hyperlinks. However, hypertext editing is an expensive and time-consuming activity which, nowadays, is hardly processed automatically, even partially. Our approach emphasizes the need for an active support to hypertext editing. We have presented a fully implemented system that helps users to link formal models to their informal sources.

We assumed in this work that the sources had a low degree of formality, roughly documents with a poorly structured content. Further investigation will adress the problem of link generation from semi-formal sources such as SGML and XML documents. With the success of XML, the availability of such semi-formal sources tends to increase. We think that link generation can be significantly improved when the sources are semi-formal. In particular, XML tagging provides useful information about the content structure that allows to accurately identify the potential link anchors.

We also adressed in this work the issue of model generation from informal sources. We proposed robust class generation mechanisms that take advantage of term hierarchies automatically built with NLP techniques. Further work will adress automatic generation of more complex knowledge structures such as relations between classes and attributes.

ACKNOWLEDGEMENTS

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*, pp. 81–90, Paris, (2000).

[3] G. Booch, *Object-Oriented Analysis and Design with Applications*, Addison-Wesley, 2d edn., 1994.

[4] 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).

[5] F. Cerbah, ‘Acquisition de ressources terminologiques – description technique des composants d’ingénierie linguistique’, Technical report, Dassault Aviation, (1999).

[6] K. W. Church and P. Hanks, ‘Word association norms, mutual information and lexicography’, *Computational Linguistics*, **16**(1), 22–29, (1990).

[7] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery, ‘Learning to construct knowledge bases from the World wide Web’, *Artificial Intelligence, Special Issue on Intelligent Internet Systems*, **118**(1-2), 69–113, (2000).

[8] B. Daille, ‘Study and implementation of combined techniques for automatic extraction of terminology’, in *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, eds., J.L. Klavans and P. Resnik, MIT Press, Cambridge, (1996).

[9] K. Elavaino and J. Kunz, ‘Docstep — technical documentation creation and management using step’, in *Proceedings of SGML ’97*, (1997).

[10] Jérôme Euzenat, ‘XML est-il le langage de représentation de connaissance de l’an 2000 ?’, in *Actes des 6eme journées langages et modèles à objets*, pp. 59–74, Mont Saint-Hilaire, CA, (2000).

[11] B. Gaines and M. Shaw, ‘Documents as expert systems’, in *Proceedings of 9th British society expert systems conference*, ed., Cambridge University Press, pp. 331–349, (1992).

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

[13] 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).

[14] O. Mariño, F. Rechenmann, and P. Uvietta, ‘Multiple perspectives and classification mechanim in object-oriented representation’, in *Proceeding of 9th ECAI*, pp. 425–430, Stockholm, (1990).

[15] P. Martin, *Exploitation de graphes conceptuels et de documents structurés et hypertextes pour l’acquisition de connaissances et la recherche d’information*, Ph.D. dissertation, Université de Nice-Sophia Antipolis, 1996.

[16] OMG, ‘XML Metadata Interchange (XMI)’, Technical report, OMG, (1998).

[17] D. Petitpierre and G. Russell, ‘MMORPH – the Multext morphology program’, Technical report, Multext Deliverable 2.3.1, (1995).

[18] 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*, pp. 291–301, Tokyo, (1993).

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

[20] D. Schwartz, ‘When email meets organizational memories’, *International journal of human-computer studies*, **51**(3), 599–614, (1999).

[21] Projet Sherpa, ‘Troeps 1.2 reference manual’, Technical report, Inria, (1998).

[22] F. Shipman and R. McCall, ‘Supporting incremental formalization with the hyper-object substrate’, *ACM Transactions on information systems*, **17**(2), 199–227, (1999).