A Knowledge-based Organisation of Lexical Resources to Support Multilingual Information Retrieval in Software Localisation

Vangelis Karkaletsis and Constantine D. Spyropoulos

National Centre for Scientific Research (N.C.S.R.) "Demokritos",
Institute of Informatics & Telecommunications, Aghia Paraskevi 15310, Athens, Greece
Tel: +301-6513110 ext. 520, Fax: +301-6532175
E-mail: {vangelis, costass}@iit.nrcps.ariadne-t.gr

Abstract

This paper discusses the writing and translation problems that multilinguality imposes to software industry. Inconsistent terminology, inefficient reuse of existing translation resources as well as lack of control of technical writing and translation process are such problems caused mainly by the inefficient organisation and retrieval of lexical resources. The paper proposes the use of a knowledge-based organisation in order to support the multilingual retrieval of terminological and translation resources by technical writers and translators. An application of the proposed organisation in a software localisation project is described.

Introduction

Localisation is the process of adapting a software product to a specific culture and market. Internationalisation, on the other hand, is the process of isolating the culture and language-specific parts of a software product during its development. An internationalised product can be easily localised by adding those features and elements that better match the target culture and marketplace. Internationalisation and localisation are becoming dominant tasks in the process of software engineering. Recent studies characterise localisation as the fastest growing area of the software manufacturing market (LISA 1994).

During software localisation, companies face problems such as the inconsistent use of terms, the inefficient reuse of existing translation material and the control of the writing and translation process (Vouros et al.). For instance, in the user interface or in the help text of a software product, the same concept may be represented by two different terms, or the same term may represent two different concepts. Inconsistent use of terms affects the quality of the software product and its localised versions considerably. When software evolves from one version to another, a lot of the phrases, sentences or even paragraphs are often used in the same or in a similar way. The retranslation of the same or similar text is something that must be avoided since it affects the cost and time required to localise a new version of the software product.

All the above problems are related to the inefficient organisation and retrieval of terminological and translation resources by technical writers and translators. The use of terminology management and translation memory systems for the organisation of terminological and translation resources and the retrieval of multilingual information from these resources, can help handle such problems (Vouros et al.).

A terminological database can help technical writers assign consistent meanings to terms and use consistent terminology across a product range or a range of platforms. It can also assist translators preventing different translations of the same term in different parts of the software, avoiding inconsistencies in the translation of terminology between the product and the localised environment in which it operates as well as improving group translation work. To achieve this, one needs to collect and organise terms used in the original software products and in their localised versions. A terminology management system can handle the creation and management of the corresponding multilingual terminological databases.

Translation is the often most resource-intensive process in localisation. During translation, you have to preserve consistency and uniformity, and resolve any possible ambiguities. Ishida (Ishida1994) noticed that 70% of text remains unchanged or undergoes only minor changes when software product versions are updated. A translation memory that contains already translated text from previous localised product versions, can considerably reduce time and cost for subsequent localisations. This reuse of existing translation resources can improve consistency between the writing style of translators, avoid inconsistencies with already translated software products, avoid the loss of formatting and hypertext information contained in the text, decrease the turn-around time for documents, and improve control over the translation process.

In this paper, we first discuss the current approaches for the organisation of terminological and translation resources in software localisation and identify their limitations that motivated us to examine the use of knowledge-based techniques. We propose the implementation of a repository of terminological and translation resources organised in a knowledge-based architecture, discuss the benefits of this architecture and present the techniques we use to reduce the cost and complexity of setting up and using such an organisation. Finally, we describe the application of the proposed approach in a software localisation project and we conclude summarising our work and outlining some areas of future research.

Organisation of Terminological and Translation Resources in Software Localisation

The set-up and management of terminological and translation resources is facilitated by the use of terminology management systems and translation memory tools. We examined the use of such resources in the context of software localisation and performed software localisation case studies, in which some widely used systems were examined (Karkaletsis et al.). Although these systems provided many facilities, we encountered problems which had to do with the amount, maintenance and formulation of these resources. This motivated us to examine other ways for organising and managing lexical resources in software localisation.

The use of knowledge bases offers several advantages over existing terminological databases (Meyer et al. 1992). It reduces redundancy and facilitates the maintenance of terminological data. A change to the characteristics of a concept is automatically inherited by all its subconcepts. An inconsistent change in the characteristics of a concept can be automatically detected. In terminological databases conceptual information is encoded implicitly in the form of definitions, contexts, etc., whereas a knowledge base encodes this information explicitly through the concepts attributes and relations. The formalisation of terminological knowledge into a coherent system of concepts facilitates the lexicalisation of these concepts to the target languages and, in general, the translation of a software product to the target markets. Users with different linguistic and cultural background lexicalise the same system of concepts taking into account the linguistic and cultural peculiarities. Interlingual knowledge bases can be used to represent the language-independent characteristics. Local knowledge bases can be attached to the interlingual one through the classification facilities provided.

The effective organisation of translation resources from previous localised product versions in a translation memory requires the use of flexible storage techniques. The indexing of translation memory entries by the concepts of a knowledge base is such a technique. If the translation memory of a software product contains text segments describing the user interface functions (functional knowledge) and it is linked to a knowledge base that contains knowledge on the user interface components (structural knowledge), then the retrieval of the description of a user interface function will be guided by the concepts representing elements of the relevant user interface components. On the other hand, such a linking can result to a more complete description of a user interface term since it may provide not only structural but also functional knowledge about this term. In other words the linking of the translation memory with the knowledge base does not only provide a flexible storage technique for translation resources but also a richer terminological knowledge base.

However such an approach suffers of a problem common in knowledge-based approaches, the cost and complexity for setting up and using knowledge bases (Reiter and Mellish 1993, Meyer 1991, Oard and Dorr 1996). We took into account the characteristics of software localisation, in order to handle this problem. The next section describes our efforts in this matter.

A Knowledge-based Organisation of Lexical Resources

Our main objective is to reduce the cost and complexity of setting up and using a knowledge-based organisation of lexical resources. It is impractical to expect technical writers to create such a knowledge base by hand. The ideal solution would be to have the technical writers write text and then convert it into knowledge base structures with a NL understanding system. However, given the state of the art in NL understanding, it is difficult to reliably and unambiguously convert text into knowledge base structures.

This led us to an intermediate approach based on the types and sources of terminological knowledge in software products. The sources of knowledge for the user interface components and their functions include the on-line help texts, message catalogues, software product glossaries, and general domain glossaries. This knowledge can be classified into general domain knowledge, knowledge on the user interface components and knowledge on the user interface functions. The problem is how to extract each different type of knowledge from these sources. We propose a knowledge acquisition approach, based on the preprocessing of the knowledge sources through the use of

controlled and mark up languages (Karkaletsis 1995). The rules of a controlled language are used to restrict the syntax and vocabulary of the texts and the rules of a mark up language to mark up in the text those structures that are necessary for the extraction of knowledge. The controlled and marked up text segments can be exploited by a knowledge acquisition system to extract automatically the required structural and functional knowledge.

The organisation of knowledge in this knowledge base is depicted in Figure 1.

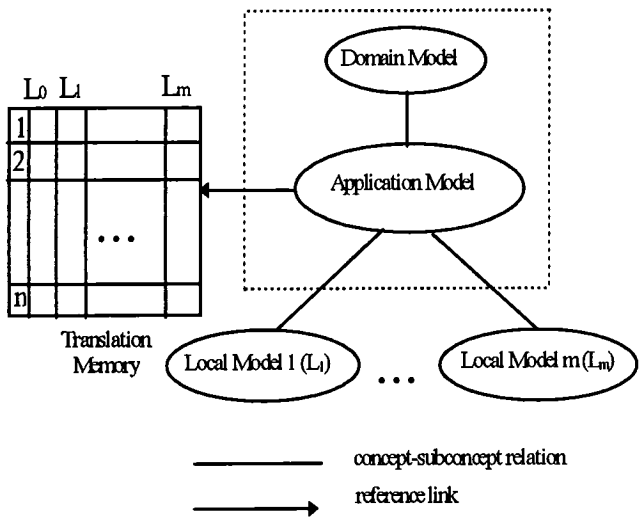

Figure 1. The knowledge-based organisation of lexical resources

This organisation is based on the linking of a translation memory with a knowledge base. The translation memory contains the 'n' controlled and marked up text segments describing the user interface functions (language L_0) and the corresponding translations in the localised versions (L_1, ..., L_m). The translation memory represents the functional knowledge of the software user interface. On the other hand, the knowledge base represents the user interface components and organises them in a taxonomic hierarchy, that is partitioned to:

- the **domain model**, which is the conceptual model of the supporting hardware and software environment,
- the **application model**, which describes the user interface components, and classifies them under the concepts included in the domain model. The domain concepts are linked to the appropriate translation memory entries via special attributes, named reference links. The application and domain concepts form the language independent part of the conceptual model.
- the **local models** which provide the lexicalisation of concepts in different languages. Local concepts are classified under the domain and application concepts and represent the language and culture specific knowledge.

The proposed knowledge representation is based on a KL-ONE-like knowledge representation formalism (Brachman et al. 1991, Vouros and Spyropoulos 1991) that involves the following formal objects:

- *Concepts.* Three types of concepts are used in the proposed representation: domain, application and local.
- *Relations.* Two types of conceptual relations are only required. Generic relations and partitive relations.
- *Attributes.* They correspond to properties of the concepts. An attribute value may be filled by another concept or by an individual. The reference link is an attribute.
- *Individuals.* They correspond to specific objects of the domain, i.e., numbers, text strings, or graphical symbols.

The following types of inference are provided for the management of those formal objects:

- *Inheritance*: properties of a concept are inherited by its subconcepts. This is done through the generic relations.
- *Inconsistency detection*: a change in a concept, or the addition of a new concept, may be inconsistent with already existing knowledge. The system detects such an inconsistency and does not permit the user to make the requested change or addition.
- *Concept classification*: all concepts that are more general or more specific than a given concept are found.
- *Subsumption*: questions about whether or not one concept is more general than another concept are answered. This type of inference is important for concept classification.

Multilingual Retrieval of Terminological and Translation Resources

To realise the proposed approach, we developed a research prototype based on a simple but efficient knowledge representation system along with a set of tools for acquiring functional and structural knowledge from help texts (Karkaletsis 1995, Karkaletsis et al. 1995).

A technical writer or a translator that wants to clarify the meaning of a term appearing in the user interface, provides this term to the prototype in his own language. The prototype retrieves first the corresponding local concept in the local model and its superconcept in the application model. It then collects all the necessary structural, functional and general domain knowledge from the translation memory and the application and domain models respectively and provides this knowledge to the user in his own language.

On the other hand, the translator that wants to express a user interface function in a language L_x, provides to the prototype some terms relevant to the corresponding user interface component. These terms correspond to local concepts in local model L_x. The prototype retrieves the appropriate application concepts and provides the technical writer with the corresponding translation memory entries in language L_x. These entries contain translations from previous versions of the software product.

Setting up and Using the Knowledge Base for a Software Localisation Project

To exemplify the proposed approach we give an example of setting up a knowledge base for the Style Manager module of the software product HP VUE (Hewlett Packard Visual User Environment). HP VUE 3.0 is a Windows Management System and User Interface, built by Hewlett Packard over X11R5 Windows for the HP-UX 9.00 operating system series. Style Manager is the application that enables the HP VUE users to change the appearance and behaviour of the screen colors, sound, keyboard, mouse, windows and sessions.

A screen of Style Manager, containing the "Color-Dialog" window is depicted in Figure 2.

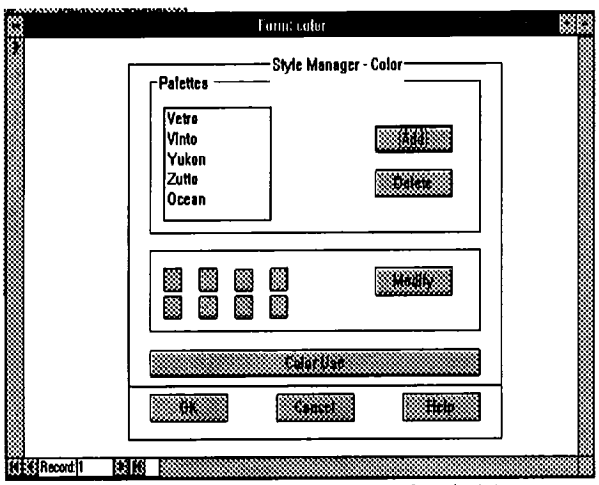

Figure 2. The "Color-dialog" screen of Style Manager

The sources of knowledge are the Style Manager on-line help, Style Manager Glossary, Style Manager Message catalogue and general glossaries on the domain. The set-up of the knowledge base follows the stages described below.

(a) *Handling of the help texts and application glossaries.* The help text and the definitions of the application glossary were rewritten using Bull's Global English rules (Bull 1993):

Rule 1. Make positive statements.
Rule 2. Keep sentence length 21 words.
Rule 3. Avoid false nomenclature.
Rule 4. One thought per sentence.
Rule 5. Use simple sentence structures.
Rule 6. Use active voice and parallel construction.
Rule 7. Avoid conditional tenses
Rule 8. Avoid abbreviations and colloquialisms.
Rule 9. Use correvt punctuation.
Rule 10. Use standardised nomenclature.

The help text and the glossary definitions were then marked up using a prespecified set of mark-up tags. This tagset includes mark-up tags for headings (chapters, sections, sub-sections, topics), tasks, glossaries terms, messages from the message catalogue, domain concepts, application concepts, generic and partitive relations, concepts attributes. Some of the mark-up tags used in our example along with their meaning are presented in Table 1. The <tag-name> denotes the beginning of the marked-up text, whereas the </tag-name> denotes the end of the marked-up text.

<task>	Textual description of a task or sub-task
<UI component>	Textual description of a user interface component
<head>	The head of a description
<body>	The body of a description
<n>	A number, that denotes a task
<part>	Part of a user interface component
<gloss>	A glossary term
<domain>	A domain concept
<appl>	An application concept
<message>	A message from the message catalogue

Table 1. Mark-up tags

Samples of controlled and marked up text from Style Manager help along with their original forms are depicted in Figures 3a, 3b.

Controlled and marked-up sample

<task><head>Modify a palette.</head>
<body>
<task><n>1.</n> Select a palette in the Color <gloss>dialog</gloss>.</task>
<task><n>2.</n> Open the modify <gloss>dialog</gloss>.</task>
<task><n>2.1.</n> Select a color <gloss>button</gloss>.</task>
<task><n>2.2.</n> Choose the Modify <gloss>button</gloss>.</task>
<task><n>3.</n> Modify the values of the color <gloss>slider</gloss>.</task>
<task><n>4.</n> Choose the OK <gloss>button</gloss> in the Modify <gloss>dialog</gloss>.</task>

<task><n>5.</n> Choose the OK <gloss>button</gloss> in the Color <gloss>dialog</gloss>.</task>
</body></task>

Original sample

To modify a palette.
1. Select a palette in the Color dialog.
2. Click a color button, then click Modify to open the Modify dialog.
3. Adjust the settings.
4. Choose OK in the Modify dialog.
5. Choose OK in the Color dialog.

Fig. 3a. The controlled, marked up and the original versions of a sample of Style Manager help (description of a User Interface function)

Controlled and marked-up sample

<UI component><head>Style Manager - Color.</head>
<body>
<part>Palettes</part>: The system lists the color palettes that are available to your display.
<part>Add <gloss>button</gloss></part>: The system displays the Add Palette <gloss>dialog</gloss> to create your own palette.
<part>Color <gloss>buttons</gloss></part>: The system displays the colors that form the selected palette.
<part>Modify <gloss>button</gloss></part>: The system displays the Modify <gloss>dialog</gloss> to modify the selected color.
<part>HP VUE Color Use <gloss>button</gloss></part>: The system displays the Color Use <gloss>dialog</gloss> to limit the amount of colors that HP VUE uses.
</body></UI component>

Original sample

Style Manager - Color
Palettes: Lists the color palettes available to your display.
Add: Displays the Add Palette dialog for creating your own palette.
Color buttons: Show you the colors that make up the currently selected palette.
Modify: Displays the Modify dialog for modifying the currently selected color button.
HP VUE Color Use: Displays the Color Use dialog for limiting the amount of colors used by HP VUE.

Fig. 3b. The controlled, marked up and the original versions of a sample of Style Manager help (description of a User Interface component)

(c) *Creation of translation memory.* The controlled and marked-up descriptions of tasks and sub-tasks are extracted, labeled and organised hierarchically, in order to form the translation memory. Table 2 depicts a part of the English(L_0) - Greek(L_1) translation memory, whereas Figure 4 depicts the marked up translation memory entries in language L_0.

(d) *Creation of application model.* All the noun phrases in the help text and the application glossary that contain domain concepts are found and extracted. They are normalised in order to form the first set of application concepts. They are then classified under the domain concepts. Using the mark-up tags for the user interface components and their parts we add the partitive relations. We repeat this process using now the application concepts collected. In this way we can find and store everything appearing on the screen (i.e. menus, labels, buttons, text fields, etc.) semi-automatically. The final step is to specify the concepts attributes (e.g. reference links). The collected application concepts are marked-up in the help text, glossary and translation memory entries (see Figure 4). A part of the application concepts system is depicted in Figure 5.

	L_0	L_1
12	Modify a palette.	Τροποποιήστε μία παλέττα.
121	Select a palette in the color-dialog.	Διαλέξτε μια παλέττα στο διάλογο-χρώμα.
122	Open the modify dialog.	Ανοίξτε το διάλογο-τροποποίησης.
1221	Select a color-button.	Διαλέξτε ένα κουμπί-χρώμα.
1222	Choose the modify-button.	Επιλέξτε το κουμπί-τροποποίηση.
123	Modify the values of the color-slider.	Τροποποιήστε τις τιμές του ρυθμιστή-χρώμα.
124	Choose the ok-button in the modify-dialog.	Επιλέξτε το κουμπί-ok στο διάλογο-τροποποίηση.
125	Choose the ok-button in the color-dialog.	Επιλέξτε το κουμπί-ok στο διάλογο-χρώμα.

Table 2. Part of the English-Greek Translation Memory

<task><n>1.2.</n>Modify a palette.</task>
<task><n>1.2.1.</n> Select a palette in the <appl>Color dialog</appl>.</task>
<task><n>1.2.2.</n>Open the <appl>modify dialog</appl>.</task>
<task><n>1.2.2.1.</n>Select a <appl>color button</appl>.</task>
<task><n>1.2.2.2.</n>Choose the <appl>Modify button</appl>.</task>
<task><n>1.2.3.</n> Modify the values of the <appl>color slider</appl>.</task>
<task><n>1.2.4.</n> Choose the <appl>OK button</appl> in the <appl>Modify dialog</appl>.</task>
<task><n>1.2.5.</n> Choose the <appl>OK button</appl> in the <appl>Color dialog</appl>.</task>

Fig. 4. Part of the marked-up translation memory entries in L_0

(e) *Creation of local models.* Two local concepts systems were created for English and Greek respectively (see Figure 5).

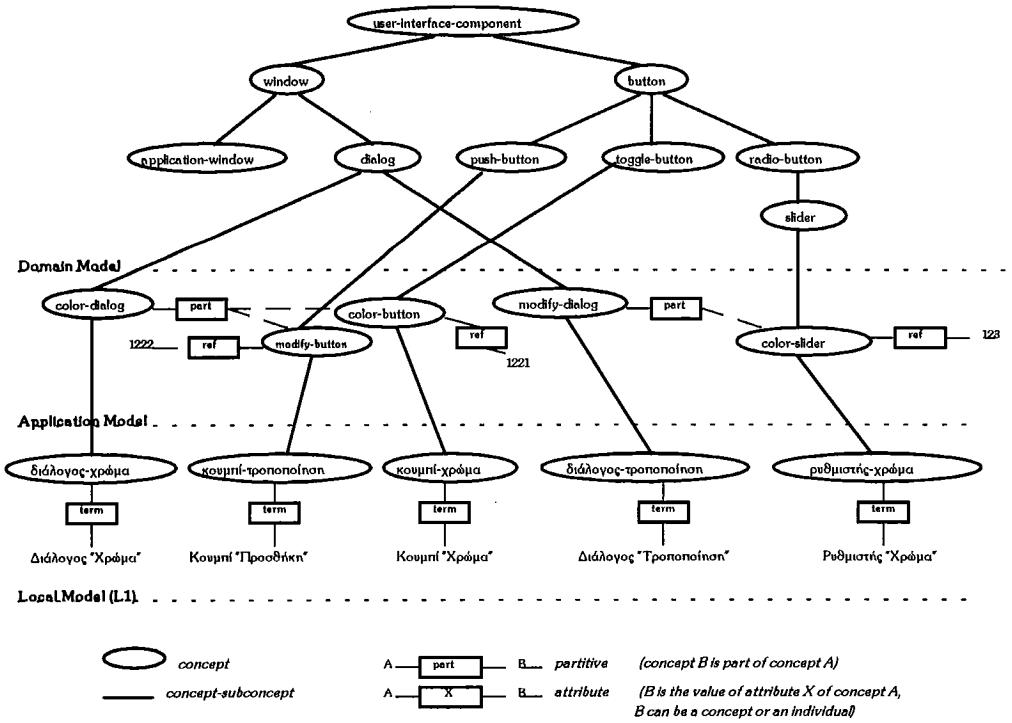

Fig. 5. Part of the knowledge base

Let's suppose that the technical writer or the translator needs information from the terminological knowledge base in order to clarify the meaning of the term "modify-button" that appears in the user interface. The prototype can provide him the information that "modify-button" is a "push-button" type of "button" "user interface component" and a part of the "color-dialog". It is also used in the task "Choose the modify-button" to "Open the modify dialog" in order to "Modify a palette" . This information can be provided in the language that the technical writer or translator has chosen.

Let's also suppose that the translator wants to express the task where "modify-button" is involved, in Greek, In that case, he has to provide the prototype with the term "modify-button". The prototype will retrieve the appropriate application concept from the terminological knowledge base and through the "ref" attribute will retrieve from the translation memory the relevant task "1222 Choose the modify-button" and its Greek translation "1222 Επιλέξτε το κουμπί-τροποποίηση".

6 Discussion and Future Plans

In our work for software localisation, we examined several existing terminology management systems and translation memory tools. Most of these systems present limitations due to the fact that they do not adequately support the definition of classes and they do not provide inheritance mechanisms. As an alternative approach for the organisation of terminological and translation resources in order to support their multilingual retrieval, we look at knowledge-based techniques and propose a hybrid knowledge representation scheme. This scheme mixes conceptual structures for the structural knowledge, and text segments for the functional knowledge. The proposed scheme facilitates the knowledge base set-up. The use of only two types of relations, the limited number of attributes, and in general the limited size of conceptual structures reduces the complexity of setting up and managing the knowledge base.

The proposed knowledge-based organisation is expected to ensure terminological consistency and facilitate the reuse of existing multilingual resources since it supports the multilingual retrieval of information by technical writers and translators during software localisation. It may also be extended for the implementation of a central repository in a software company that will contain all the terminological and translation resources of the company's software products and localised versions organised in the proposed knowledge-based structure according to the products types and platforms.

The use of such an organisation may also facilitate other software localisation activities. We have already exploited it in the area of multilingual message generation for software applications (Spyropoulos et al. 1995).

References

Brachman, R.J.; McGuiness, D.L.; Patel-Schneider, P.F.; Resnick, L.A.; and Borgida, A. 1991. Living with Classic: When and How to Use a KL-ONE-like language. *Principles of Semantic Networks-Explorations in the Representation of Knowledge*, Sowa, J. ed. 1991, 401-456.

Bull, 1993. Bull Global English. Bull ILO, Paris, 1993.

Ishida, R. 1994. Future Translation Workbenches: Some Essential Requirements. *LISA Forum Newsletter*, vol II, July 1993.

Karkaletsis, V. 1995. Exploitation of Terminological Knowledge Bases in Multilingual Software Products. Ph.D. diss., Dept. of Computer Science, Athens Univ. (in Greek).

Karkaletsis, V.; Spyropoulos, C.D.; Vouros, G.; and Halatsis, C. 1995. Organisation and Exploitation of Terminological Knowledge in Software Localisation. *TermNet News - Journal of the International Network for Terminology*, no 48, 42-48.

Karkaletsis, V.; Kokkotos, S.; Spyropoulos, C.D.; Vouros, G.; and Hudson, R. Commercial Tools to Support Localisation. *Software without frontiers*, Hall, P.A. and Hudson, R. eds.: J.Wileys&Sons, London (in press).

LISA, 1994. Software Localisation in Europe, Findings from the OVUM Study. *LISA Forum Newsletter*, vol III, Febr 1994, 3-12.

Meyer, I. 1991. Knowledge Management for Terminology-Intensive Applications: Needs and Tools. In Proceedings of 1st SIGLEX Workshop, June 1991, Lecture Notes in Artificial Intelligence, vol. 627, 21-37.

Meyer, I.; Skuce, D.; Bowker, L.; and Eck, K. 1992. Towards a New Generation of Terminological Resources: An Experiment in Building a Terminological Knowledge Base. In Proceedings of COLING'92, Nantes.

Oard, D., and Dorr, B. 1996. A Survey of Multilingual Text Retrieval, Technical Report, UMIACS-TR-96-19 CS-TR-3615, Dept. of Computer Science, Maryland Univ.

Reiter, E., and Mellish, C. 1993. Optimising the Costs and Benefits of Natural Language Generation. In Proceedings of IJCAI 1993, 1164-1169.

Spyropoulos, C.D.; Karkaletsis, V.; Vouros, G.; Honkela, T.; Lagus, K.; and Lehtola, A. 1995. Investigating On Line Message Generation in Software Applications: The GLOSSASOFT Results. In Proceedings of the ERCIM Workshop "Towards User Interfaces for All: Current Efforts and Future Trends", Heraklion, Crete, Greece, 30-31 October 1995.

Vouros, G.; Karkaletsis, V.; and Spyropoulos, C.D. Documentation and Translation. *Software without frontiers*, Hall, P.A. and Hudson, R. eds.: J.Wileys&Sons, London (in press).

Vouros, G., and Spyropoulos, C.D. 1991. The PHOS conceptual language for knowledge representation. *Engineering Systems with Intelligence: Concepts, Tools and Applications*, Tzafestas S.G. ed. 1991, 43-50: Kluwer Academic Publications.