

It's not Greek to me:

Terminology and the Second Language Problem

Giorgos Lepouras and George R S Weir

Abstract

The localisation of software applications is a common way of avoiding 'second-language' difficulties but can lead to new problems in user interaction. One determinant of this condition is the selection of appropriate terminology for local language use. The present paper reports on a comparison of terminology from three leading word processor packages, in their English and Greek language varieties. Diversity in terminology, the resort to transliteration, and scope for neologism are characteristic results of localisation that underline the risks of confusion in the target user population. This paper discusses the possibility of preventative measures and offers a remedial approach to this problem for use in the considered applications.

Keywords: human-computer interaction, standardisation, internationalisation, localisation, terminology, user support.

Introduction

Some problems in user-computer interaction arise under the influence of language. Whenever non-native speakers of English have to use English-based software there are inevitable risks that the user's degree of English competence or influence from cultural background will impact detrimentally upon their interaction with the computer system. A common strategy to avoid this 'second-language problem' is resort to the use of fully localised software applications. Ostensibly, this resolves the problem, but, as we have argued elsewhere [10], there is likelihood that new difficulties are introduced by this manoeuvre. Software localisation is principally a process of translation from an original language version (usually, English) to another 'local' language

version [5, 7, 9, 11]. Among the factors that may introduce problems in localisation is the lack of standardised terminology within a local language. In support of this thesis, our research on the second-language problem has revealed a range of linguistic anomalies that arise from a lack of standardised terminology. The present paper highlights some of the obstacles to standardisation of terminology and the effects of this phenomenon. This is illustrated by examples taken from three Greek localised versions of well-known word processors. Beyond discussion of these issues, we highlight a number of strategies designed to combat or alleviate this problem.

Standard Terminology

The rapid evolution and near ubiquity of interactive computer systems confronts users with a growing variety of terms which describe new or unfamiliar concepts. Since this evolution takes place mainly in the English speaking world most new terms or new applications of existing terms emerge in English. This has a knock-on effect for localisation with the need to translate or create 'equivalent' new terms in each local language. This is not always easily achieved and in operation the results are often far from ideal. Standardised local terminology would seem to offer relief for this condition.

Standardisation is the process of making regular (of one size, shape, quality, etc.), according to fixed or agreed standards. In the case of terminology, standardisation is the process that ensures that everyone uses the same term to denote a specific concept or action. Although this sounds straightforward, in practice it is difficult to achieve. Despite the trend toward direct manipulation interfaces, standardisation of terminology is still a major issue for human-computer interaction.

There are many reasons why standardisation of terminology is not easily secured. In the first place, standardisation takes a long time. Users cannot wait while the standards process trundles toward its conclusions. Where bodies bear responsibility for the standardisation of localised terminology there may be doubts as to their efficacy. As a result, other groups, usually with their own agenda, generate terminology as a matter of course.

Confusion is a predictable result when new terms are produced and disseminated by a variety of interested parties, more or less relevant to the field. For example, one readily finds instructional books dealing with a common topic, which employ different terms. While pluralism in other fields may be a healthy notion, the absence of any control or check on the diverse application of terminology easily generates confusion for the naïve user.

Clearly, producing a reference list of translated terms is not in itself enough. Terms have to be learned and used by the widest possible group of users or they remain simply proposed terms. Software producers usually hope to impose their own conventions. For such companies there is an advantage in having their terminology used widely in the market. Terminological inconsistency with applications from rival software companies discourages users from switching to less familiar alternative products.

For applications built upon Microsoft's Windows interface, a set of recommended terminology has been published as a guide for application developers working in European languages [8].

Localisation

Product localisation is not always the path chosen by international software houses. Many companies are content to offer English-only versions, despite the difficulties facing non-native speakers of English when confronted with such applications. Others support 'international' users by means of international settings that accommodate known differences, such as date syntax, paper size, sort order, etc [1]. While such 'culturalisation' may be desirable, we regard this as an aspect of internationalisation that is secondary to the issue of comprehension (cf. [2]).

Elsewhere [10], we have recommended a second-language explication approach to the support of such users. This accords with Belge on the desirability of supporting users 'at the task level' [1, p.24], but focuses support at the linguistic level rather than considerations of cultural convention. In our view, the worst approach to localisation entails full translation of system terminology. Our subsequent discussion addresses this native language localisation as an approach that is problematic on a number of counts.

By its nature, localisation of existing software applications introduces a further level of potential terminological confusion. Even where common usage is established across an existing range of English language products, there remains scope for variation in the choice of local (translated) terms.

Variability in language usage is not unique to English. Fernandes notes that differing dialects may result in comprehension difficulties [3]. This will be equally true across dialects of non-English languages.

The localisation process is also constrained in its selection of terms by limitations introduced at the user interface. Translated terms may strive to be accurate, but may also be limited in length if required to fit the look and feel of an existing user interface.¹ For this reason, using multiple words to achieve accurate translation of a single non-local term is unusual and constrained by available space. Figure 1 illustrates the 'View' menu from the English and Greek versions of Word6.

Notably, this example retains a one-to-one correspondence for menu items and employs multiple Greek word renditions for several commands. Obviously, this trade-off between conciseness of expression and accuracy of translation is problematic and a potential threat to the integrity of an original application design, since it may impact upon the usability of a localised version.

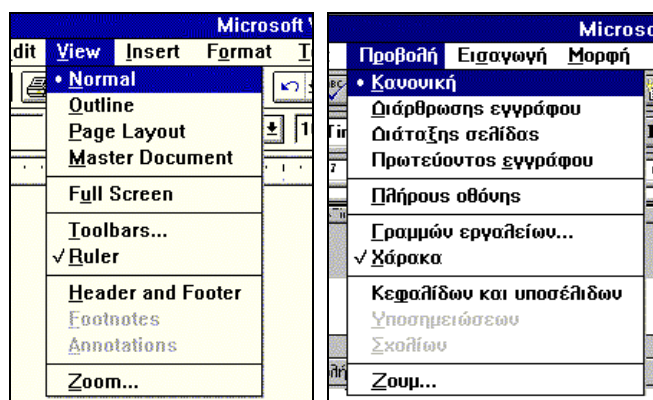


Figure 1: The 'View' menu from Word 6 (English and Greek versions)

Translation is also a costly process. A satisfactory rendering of an application's language from English to some native variety may require the services of several experts. Significantly, the required expertise must span the source and destination languages as well as the technical know-how for competent application use. Employing such a conjunction of talents can be expensive.² Less expensive options risk less adequate solutions.

While the appeal of localised software is undeniable, our belief is that the process of localisation may damage the

¹ This problem is especially pronounced in translations from English to Greek since Greek words are often lengthy.

² Software houses occasionally employ academics from computer-related disciplines in the country of localisation, to provide the necessary local language expertise. This must be more economical than maintaining in-house staff but the use of non-professionals is more likely to compromise the integrity of the finished product.

integrity of an application by introducing user difficulties absent in the non-localised version. Translation and terminology play a major part in this scenario, which may include incompatible applications and confusion among users. In the following, we describe our survey of three localised software applications, with a comparison of their terminology in Greek (the localised version) and English (the original version).

Cross-Language Comparisons

In a comparison between three well known and widely used, localised word processors (Greek Word 6.0 for Windows, Greek Lotus Ami Pro 3.0 for Windows, Greek WordPerfect for Windows, and their English counterparts³), we found that the English versions used the same core set of terminology, but this was not true for their Greek versions. As shown in Table 1, the total number of commands that can be found in the menus (including submenus) vary among the three word processors. The three word processors in their English versions share a total of 32 identical commands (such as New, Open, etc.). When it comes to the equivalent Greek versions the number of common commands is reduced to 21 (30% less than the original language set). These figures (Table 2) indicate greater variation in terms across the Greek language versions in contrast with their English counterparts.

Table 1: Comparison of total menu command set

Package	Total Commands
Word	110
Lotus Ami Pro	140
WordPerfect	219

Table 2: Comparison of core command set

	English versions	Greek versions
Common commands	32	21

The process of localisation begins with a set of original English command terms and makes a series of decisions on the appropriate conversion, translation or replacement for these terms within the local language [6]. Every term can be handled in one of three ways, viz., translation, neologism, or original retention.

Translation maps the meaning of the original onto a local expression with the same or similar connotation. This may

entail selection from one of several alternative meanings for the original expression.

Neologism is an option whenever translation is problematic or undesirable. Here the original word is not equated to an existing local term with similar meaning but is mapped to a newly coined local expression with no established usage.

The only remaining means of dealing with an original application term is to retain its original form either by transliteration into the local language (keeping the original sound with a localised written form) or by ‘borrowing’ the English original for use within the local language. Particularly in the latter case, this ‘original retention’ results in a mix of ‘converted’ local and ‘unconverted’ English terms in the localised version of the application. One reason for adopting this strategy may be assumed existing familiarity with specific English terms or the need to render a newly coined English expression, such as a Trade Mark.

This range of options in the localisation process is illustrated graphically in Figure 2 (shaded boxes indicate a local language construct). The possible procedures that map from original terms (source) to localised terms (result) is given by the intermediate tree level.

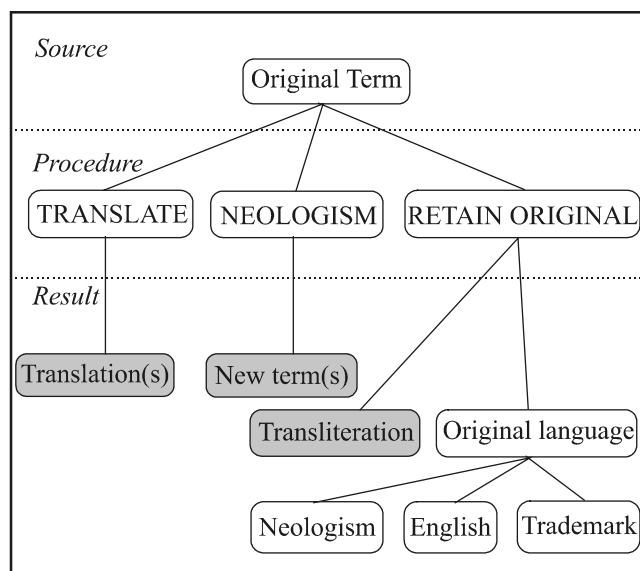


Figure 2: Localisation as Term Conversion

Through analysis of command menu items we distinguished two major sources of inconsistency between the terms used among the Greek versions of the word processors. These were:

- Terms with more than one possible translation;
- Terms with no equivalent Greek expression.

³ While MS-Word 95 and MS-Word 97 were localised for the Greek user community, WordPerfect's later version and Word Pro (the successor of Lotus Ami Pro) were not localised. For balanced comparison we have adhered to Greek Word 6.0 for Windows, Greek Lotus Ami Pro 3.0 for Windows and Greek WordPerfect for Windows.

Terms with multiple translations

This is the most common source of inconsistency between the Greek language applications. In such cases, terms that were shared across the English versions of the word processors differ when represented in their Greek equivalents. Most often the diverse terms, although semantically related, convey different and sometimes confusing meanings. Several examples of such variation are noted in Table 3, below.

Table 3: Comparison of core command sets

English	Word 6.0	Lotus	WordPerfect
Header	κεφαλίδα	υπέριτιτλος	τίτλος σελίδας
Footer	υποσέλιδο	υπότιτλος	υποσέλιδο
Tile	(not present)	χωρίς επικά-	εμφανή
cascade	(not present)	ορατοί τίτλοι	επικαλυπτόμενα
ruler	χάρακας	σηλογόμονας	χάρακας
bar	γραμμή	(not present)	στήλη
column	στήλη	(not present)	κολόνα
bullet	κουκκίδα	σύμβολο	σφαίρα
break	αλλαγή	διακοπή	αλλαγή
undo	αναίρεση	ακύρωση	ακύρωση
cut	αποκοπή	κοπή	κοπή
view	προβολή	όψη	θέα

The term 'header' may be familiar to native speakers of English and is used across each of the English-based word processors, but the choice of Greek equivalent is not straightforward.

Microsoft has replaced 'header' with 'κεφαλίδα'. Although this is the accurate translation of the English term, it is an uncommon expression in Greek with no familiar connotation. In contrast, Lotus has replaced 'header' with 'υπέριτιτλος'. This Greek term means 'supertitle'. Finally, in Greek WordPerfect we find that 'header' has been replaced with 'τίτλος σελίδας'. This Greek phrase literally means 'page title'. There can be little doubt that Greek speakers face confusion over this range of terminology, particularly since the header bears no obvious connection to the title of a document.

The selection of a Greek term for 'footer' is also problematic. Microsoft and WordPerfect use 'υποσέλιδο' which is an accurate but (for most Greek users) unfamiliar translation, whereas Lotus uses 'υπότιτλος' which means 'subtitle' and bears no clear relation to footer.

The Greek equivalent of the English term 'bullets' appears in Microsoft Word as 'κουκκίδες'. The commonest interpretation of this Greek term corresponds to the meaning of 'bullets' (as found in a document format).

WordPerfect, in contrast, uses a literal translation of 'bullets' ('σφαίρες'), which usually refers to bullets as found in a gun (or more generally to 'spheres'). Finally, Lotus Ami Pro substitutes 'bullet' with 'σύμβολο' which means 'symbol'. Again, the lack of standard terminology promises scope for misunderstanding.

Another example can be drawn from the translation of 'view'. Once more, the choice of a Greek equivalent is difficult. Microsoft Word replaced 'view' with 'προβολή' (i.e. projection), a choice that will puzzle most users since projection is slightly related to the appearance of the document. Lotus replaced 'view' with 'όψη', which means appearance or view, while WordPerfect used 'θέα' which again means view, but rather like a room with a view.

Some of the difficulties faced during the translation process of terms occur because the direct translation for the English term does not convey the original meaning. One such example is 'tile'. Although English users may be familiar with the connotation of tile in the context of windows applications, most Greek users will be confused if the equivalent Greek word was to be used. Instead, Ami Pro uses 'χωρίς επικάλυψη' (i.e. non-overlapping) and WordPerfect uses 'εμφανή' (i.e. visible).

The same range of difficulties is encountered in the translation of 'cascade'. Since, the equivalent Greek term does not convey the original meaning, alternative terms have been employed. Lotus Ami Pro uses 'ορατοί τίτλοι' (i.e. visible titles) and WordPerfect uses 'επικαλυπτόμενα' (i.e. overlapping). Once again, choosing a word to convey the original meaning is not easy. In both examples, the English term is being substituted by a phrase (i.e. 'tile' becomes 'χωρίς επικάλυψη' and 'cascade' becomes 'ορατοί τίτλοι') that describes the command. Aside from their adequacy in communicating the original meaning, a multi-word replacement can prove problematic in user interfaces where space is usually limited.

Terms with no equivalent Greek expression

The second major category is comprised of terms that have no equivalent expression in Greek. This lack can be remedied by inventing a new Greek term (neologism), or by retaining the original English term in some form. In the case of the localised versions of the three word processors, both approaches have been adopted, with English retention being the more common.

The term 'μαρκάρισμα' is a neologism. This term uses the root of the original English term (i.e. mark) with a Greek suffix. Other terms like 'Small Caps', 'textart' or 'kerning' remain in the localised versions the same as in the English versions. Terms like 'ζουμ', 'στυλ' or 'εφέ' (i.e. zoom, style and effects) appear in a transliterated form. Finally, trademarks like SmartIcons™ and ToolBar™ appear unchanged in the localised versions.

Although there were no examples in the considered applications, Figure 2 also indicates the possibility of retaining an original neologism from the English versions. Presumably, such terms would be equally new to their English as their Greek user population.

Tactical Response

Evidence from three localised word processors indicates significant variations in the application of terminology. This situation is worrying from a user perspective and should also concern the developers of localised systems. Two types of response to this situation are envisaged. On the one hand, some preventative steps may be considered in order to minimise future variability in terminology and thereby reduce the risk of adverse impact upon the end-user. On the other hand, given the prevalence of such discrepancies across software applications, a remedial response to this situation is also appropriate.

Preventative measures

Ideally, terminological variation would be minimised through use of established standards for command vocabulary, but this is unlikely to occur, unless the role of standardisation body is improved. We have already described the tendency for diverse groups to formulate idiosyncratic vocabulary, with little or no control. Minimising this would require some mechanism(s) for registration and control of terminology which should also be able to cater for the (terminology) user's needs [4]. This could conceivably be addressed through one or more standardisation bodies but would require the following facilities:

Fast response to the evolution of new terms

In the rapidly evolving area of informatics, new terms are produced almost daily. This means that the standardisation body should be able to assimilate new terms and process them quickly enough to meet the industry's needs. Optimally, institutions, companies, etc. producing or requiring new terms would register these with the standardisation body.

Use a board of experts

To be able to produce translated terms that are both grammatically and semantically correct, experts in the field of informatics as well as the field of language are needed. This board of experts will have the responsibility of introducing new terms. Since the number of persons affected by the introduction of new terms is wide, the group of experts cannot function without the approval of people concerned. Therefore, the standardisation body should establish tight links with institutions and companies. To assure the successful standardisation of terminology, new terms have to get the broadest possible acceptance.

Disseminate new terms

The existence and timely dissemination of recommended terminology is necessary to ensure that people concerned are being informed. To this end electronic media of information dissemination can be used to minimise the time needed to notify all parties concerned. If the terms introduced have the approval of the users' majority and the terms are being timely disseminated then the tendency of individuals to produce their terms will be alleviated.

Remedial measures

An alternative tactic for addressing the variability of terminology in localised systems is to make such variety explicit to the end-user. This type of approach can be especially useful to experienced users who already have a good knowledge of the terminology and are confused by the choice of terms in new software. Of course, this is itself a challenge, since any explicit advice to users must serve their benefit whilst not impeding their use of the software application.

For MS-Windows applications, like those described in our comparison, we have considered several alternative techniques as vehicles for delivery of terminology advice to end-users. As with all user support activities, two components are crucial. First, the content of any advice must be selected to maximise the benefit while minimising extra load on the user. Second, a means of delivery must be devised that is both effective and unobtrusive. A further factor is the need to address the problem in a generic fashion, i.e., by means that can be applied to every application.

Content

The potential content of advice for the programs considered in our survey is readily assembled from our comparison of these applications. Of greatest import is information on terms that have been localised to more than one meaning, followed by details of original English terms that strictly have no local equivalent. Apart from the equivalent localised terms that could be delivered to the user, additional support could be given in the form of definitions, examples, annotations, etc. Information on more subtle aspects, such as transliterations, trademarks and neologisms, are an optional extra.

Delivery

Depending on the content of information that has to be delivered, its presentation may assume different forms. In the case of equivalent localised terms and experienced users, the user should be able to select the terminology set that is most familiar. For example, let's assume that a user is familiar with MS-Word and wants to work with Lotus Ami Pro. In that case, each time the user selects a command in Ami Pro that has an equivalent but with different translation in MS-Word, the familiar MS-Word command will be presented as a subtitle under the Ami Pro command. In cases where additional support (such as definitions) has to be delivered to the user, a separate window, in the form of a status bar, can be employed.

Implementation

The implementation of a terminology presentation application can be based on the client-server nature of current window systems, in which applications communicate with the system, and with each other, by message passing. This facility enables us to monitor and filter these messages and so determine the actions carried out by users. In turn, this provides a basis for additional user support.

Notably, this type of support is not hard coded in the applications it supports, but it is implemented as a separate application. This affords a system wide support technique that can be applied generally, not solely to support for variations in the local language terminology.

For our purposes, a 'filtering' component determines when the user selects a particular command and sends this information, with the name of the application, to a separate data processing and user presentation module. Depending upon the user's preferences, the support module checks for an equivalent alternative term and presents it to the user. Figure 3 depicts this approach.

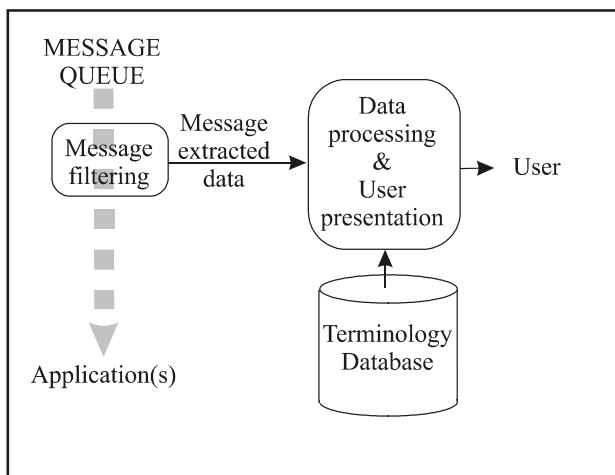


Figure 3: Generic terminology support process

Using this approach, we have developed a generic application that runs on MS-Windows to provide terminology support to users. The application employs a terminology database containing pairs of terms (sets of Greek alternatives) from the three word processors.

Figure 4 shows the Lotus Ami Pro word processor with support for MS-Word users. Although this type of terminology support may benefit all users, it was envisaged that it could be of special value to users already familiar with the terminology used by a 'rival' software package. To this end, an experiment was set up to explore the possible users' gains by this type of support.

Experimental testing

For the experimental testing of the support application two groups of ten subjects were employed. Both groups consisted of experienced Greek MS-Word users that had never worked with Lotus Ami Pro (although they were familiar with other word processors).

Both groups were asked to complete eleven different tasks with Lotus Ami Pro. Each task required users to select appropriate menu commands. Two of the tasks were based on commands that not only remained the same between Word and Ami Pro, but also remained in the same menu. Two other tasks were based on commands that remained the same between Word and Ami Pro, but appeared in different menus.

The remaining seven tasks employed commands that use different translations in the two word processors. Two of these were found in the same menu (although the menu titles used different translations) and the rest were found in different menus. All tasks were presented to the subjects without using terminology from any of the word processors.

Prior to the experiment, all subjects had a few minutes to acquaint themselves with the application, without external aid. Furthermore, they were advised that while interacting with the word processor they could meet differences in menu categorisation or command terminology.

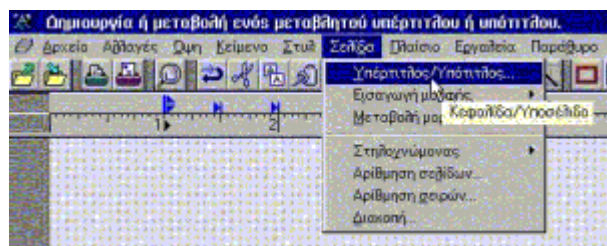


Figure 4: Lotus Ami Pro with support for MS-Word users

The first group of subjects had no access to terminology support while the second did. Each subject was asked to complete all of the tasks, while an evaluator recorded the time required.⁴ All subjects were advised of the nature of the experiment. Subjects with access to the dynamic support were advised of its presence and purpose. Both groups consisted of graduate students. The results of this experiment are summarised in Table 4.

Table 4: Experimental results

Task	With Terminology Support		Without Terminology Support		Significance test (Z statistic) 1-tailed P
	Mean	Std.	Mean	Std.	
1.	6.6	2.0	6.6	1.8	0.4847
2. *	50.5	21.6	76.6	24.2	0.0126
3. *	19.9	6.3	20.0	5.8	0.4248
4.	10.0	4.7	13.6	4.8	0.0647
5. *	22.9	12.4	36.6	8.2	0.0036
6. *	20.4	11.0	25.0	9.5	0.1814
7. *	24.4	12.7	24.1	11.0	0.4698
8.	11.2	5.2	27.8	10.7	0.0011
9. *	19.0	5.3	20.1	6.0	0.2847
10. *	13.9	5.2	20.4	5.7	0.0104
11.	4.4	1.3	4.5	1.2	0.4063

Tasks shown in gray employ commands with different translations between the two word processors. Tasks with an asterisk employ commands located in different menus in each word processor.

Our experimental results show that each group with access to terminology support application performed significantly

⁴ Using a chronometer, timing was performed manually on each task, from its outset until the user selected the correct menu command.

better in tasks using commands with different translations between the two word processors. Notably, there was no major variation between groups for the tasks that relied on common commands. Mean times for task performance are show in Table 5.

The results were further analysed using the Mann-Whitney non-parametric significance test, estimating significance level using the Z statistic. The first noteworthy point is that the times measured for the tasks that relied on the same commands come from a common distribution. Furthermore, there is a significant difference for the times measured for tasks 2, 5, 8, and 10 (with significance levels far less than 0.05), proving that there is an improvement in response times when local language support is deployed.

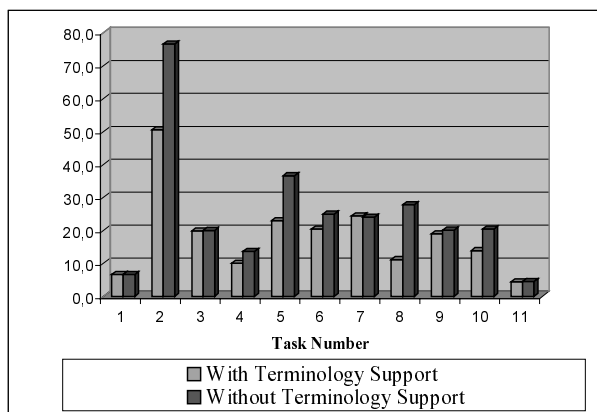


Figure 5: Mean time (seconds) needed to complete each task

For task number 4 the significance test is inconclusive, while there is no apparent difference between the two sets of measurements for task number 6. The latter may be attributed to the fact that the alternative term used in Lotus Ami Pro is similar to that used in MS Word ('page break' in the first case and 'page change' in the second). It may also be affected by the easy identifiability of the command in Lotus Ami Pro (there is a menu named 'Page' where one can find the command 'Break').

Task number 9 is a further special case. Although the Ami Pro terminology in this task differs from MS-Word, subjects were able to infer the meaning of the unfamiliar command from the rest of the commands in the group. Specifically, since the task was to create justified alignment and the rest of the commands in the submenu were left, centre and right, subjects could readily deduce that the last command was the one they required.

Subjects were also asked to comment on their experience. From their answers it is worth noting that all subjects expressed some confusion and dissatisfaction with the choice of terms. Although in some cases they could guess what the actual term meant, in other cases they had to think in terms of the likely English equivalent in order to find out.

Of course the difference in the use of terms was not the only source of confusion. Some of the subjects (eight in number)

were also puzzled with the categorisation of commands into menus, which was based on a different philosophy from MS-Word. The experimental results endorse this point, since commands that remained in the same relative position were found in less time than other commands. This is further supported by the evaluator's notes, which show that subjects seeking an unfamiliar command usually started searching from the relative position of the command in MS-Word.

Summary And Conclusions

The experimental results show, as one would expect, that timings are comparable where the command terms are common across the applications. Significantly, timings for other tasks show an enhanced performance for those users with terminology support. Our starting point suggests benefit through use of such selective language support. Our experiments endorse this assumption.

Terminological variations are evident across a range of localised software packages. The origins of such variety have been discussed in this paper as a component of the 'second language problem'. A remedial approach to user support has been described, based on a separate dynamic help system, which can be used across a range of such applications. The experimental results show that, especially for experienced users who want to use a new application with a different terminology set, this approach can help them overcome the obstacles set by terminology inconsistencies.

This experimental work has addressed instances of terminological variety within localised Greek software. If command language variability across localised software applications cannot be avoided, users should at least be advised and empowered in order to minimise its detrimental impact.

The work described here underpins our general view of the need for selective language support, especially where users are faced with a non-native language software application. Such user support will find a place in a wide variety of user-system interaction (not only when changing between applications in the same area but also when installing a new application with varying terminology). Notably, terminological consistency across a single software producer's product range offers no advantage when using a word processor from this company A and a design package from another software producer.

Further evaluation is underway to illustrate the scope for enhanced interaction through second language support in such contexts.

References

1. Belge, M., The next step in software internationalization, *Interactions*, 21-25, Jan 1995.

2. Bourges-Waldegg and Scrivener, S. R. A., Meaning, the central issue in cross-cultural HCI design, *Interacting with Computers* (9), 287-309, 1998.
3. Fernandes, T., Global interface design, *Proceedings of CHI'94*, 373-374, ACM, 1994.
4. Humbley J, Summary of results of terminology surveys in France, 1995. <http://www.mcs.surrey.ac.uk/Research/CS/AI/pointer/enqfr95a.html>
5. Karat J. and Karat C. M., Perspectives on Design and Internationalization, *SIG CHI Bulletin*, vol. 28, no 1 Jan. 1996, p. 39.
6. Khursid A, Language Engineering and the Processing of Specialist Terminology, Department of Mathematical and Computing Sciences, University of Surrey, 1996.
<http://www.mcs.surrey.ac.uk/Research/CS/AI/pointer/paris.html>
7. Russon P. and Boor S., How fluent is your interface? Designing for international users. *Proceedings of InterCHI '93*, ACM Press, 342-347, 1993
8. The GUI Guide, International Terminology for the Windows Interface, Microsoft Press, European Edition, 1993.
9. Uren E., Robert H. and Perinotti T. Software Internationalization and localization. An introduction. Van Nostrand Reinhold, New York, 1993.
10. Weir, G.R.S., Lepouras, G. and Sakellaridis, U., (1996), Second Language Help for Windows Applications, in M.A. Sasse, R.J. Cunningham and R.L. Winder (eds.), *People and Computers XI*, *Proceedings of HCI '96*, Springer, London, pp. 129 – 38. 1996,
11. Yeo A. and Barbour R. H., Software for the rest of the world, Working Paper 96/2, Computer Science Department, University of Waikato, New Zealand, 1996.

About the Authors

Giorgos Lepouras is a PhD candidate at Department of Informatics, University of Athens, Greece. His research interests include internationalisation, localisation issues and user support issues. He has an MSc in Information Technology Systems from University of Strathclyde, Glasgow and a first degree in Mathematics from University of Athens.

George Weir is a lecturer in Computer Science at the University of Strathclyde, in Glasgow, Scotland where he teaches Human-Computer Interaction and Multimedia. He has a special interest in user support issues and has written and edited several books and papers in HCI. He has a doctorate in Philosophy from Edinburgh University (1983) and a Master of Arts from the University of Glasgow (1976).

Author's Addresses

Giorgos Lepouras
Dept. of Informatics,
University of Athens,
Athens 157 71,
Greece
Tel: (301) 7275220
Fax: (301) 7275214
G.Lepouras@di.uoa.gr

George R. S. Weir
Dept. of Computer Science,
University of Strathclyde,
Glasgow G1 1XH,
Scotland, UK
Tel: (44141) 552 4400 (ext. 3915)
Fax: (44141) 552 5330
gw@cs.strath.ac.uk