# The Second Language Problem in User Support[1]

**George R S Weir**
Department of Computer Science
University of Strathclyde
Glasgow G1 1XH
UK

e-mail: gw@cs.strath.ac.uk

**Giorgos Lepouras**
Department of Informatics
University of Athens
Athens 15771
Greece

e-mail: glepoura@di.uoa.ariadne-t.gr

Research Report No. RR-174-95

Abstract

Interaction between humans and computers is seldom perfect. Interactive software often includes aspects of user support to reduce the uncertainty and propensity to error experienced by many users. The present paper briefly reviews some of the common strategies for helping users and characterises the 'second-language problem' as a special case for support.

Since most software packages are written for an English-speaking audience and presuppose reasonable proficiency in English comprehension non-native speakers of English face the daunting task of interacting via a second-language. One can reasonably expect greater confusion and interactive difficulties in this class of users. Software implementors have gone some way toward addressing this problem with occasional provision of font and native language support (localisation). Some of these attempts are outlined here. The paper concludes by advocating a strategy of selective native language support based upon reasonable expectations of troublespots in interaction.

## 1. Introduction

Computer users rarely experience entirely trouble-free interaction. Whatever the merits of natural variety across individuals such variations virtually ensure that no software system yields constantly fluent interaction for all users. The design and implementation of a user interface remain inherently difficult tasks (Meyers, 1994).

Although current graphical user interfaces may be much easier for novice users to learn and use than earlier command-line (text-based) systems, the increased complexity of the tasks and applications that has to be addressed through the user interface, combined with the diversity of users, makes it impossible to guarantee trouble-free interaction. System designers must acknowledge the inevitability that some end-users will experience instances of 'cognitive dissonance' or 'user-system mismatch'.

Frequently, such troubles derive from inexperience on behalf of the user. Often they arise through poor attention to detail on the system design. For either reason, software developers are accustomed to ameliorate the situation by providing a number of different facilities — often as supplements to basic system functionality — that collectively constitute 'user support'.

[1] This work is part-supported by the British Council. Please address correspondence to the first-named author.

The range of possible difficulties in human-computer interaction is so extensive that one might regard meeting the full need for support as beyond human effort. On the other hand, the likelihood of patterns in the behavioural characteristics of computer users and system designers suggests that a wide spread of techniques should be the most effective approach to preventative and remedial assistance. A number of authors have sought to classify techniques for user support. Some of these schemes are outlined in the following section.

## 2. Varieties of user support

Current interactive systems often try to lighten the cognitive load of the user by providing feedback regarding the state of the application in the form of status information, error or warning messages (GUI Guide). Systems may also provide on-line help or even a tutoring system. These, and a number of other user support techniques are reviewed by Houghton (Houghton, 1984) who delineates the range of different types of assistance provided in interactive systems, viz.,

- Command assistance;
- Error assistance;
- Prompting;
- On-line tutor;
- On-line documentation

'Command assistance' characteristically provides a brief explanation of the command followed by a list of possible parameters that may be specified along with the command.

'Error assistance' aims to provide further information as follow-up to an initial system generated error messages.

'Prompting' assists the user in carrying out a task through stepwise support for its component actions. This strategy is often effective in preference to a simple error message, especially when an incomplete or incorrect goal can be detected by the interactive system.

User assistance is often taken further in the provision of an 'on-line tutor' as an adjunct to base system operation. This may aim to provide information that describes normal user-system interaction (e.g., the tutorial program included with Microsoft's Word for Windows) or may offer a training environment in which progressively more complex user-system interaction may be practiced.[2] Often, the latter allows the user to try commands without risk.

A simple but commonly adopted strategy is to make traditional hard-copy user documentation available for active consultation by the user. Such 'on-line documentation' may also support a retrieval mechanism which yields command assistance.

Although Houghton's original account of on-line help only addressed command-line interfaces, most of the varieties of assistance described therein appear in recent graphical user interface systems. Thus, the help facilities included in Microsoft Windows applications generally provide an on-line documentation system that includes a *search* option as a help retrieval mechanism.

[2] This is the 'training wheels' strategy described by Carroll et al, 1987

Being a prevalent software environment, MS-Windows provides a convenient reference point for the use of on-line support.

The published guidelines on building Windows compliant interfaces (Windows Interface, 1991, p.96ff) specify that every application should include a help menu with the following items :

- *Contents*;
  When selected this item opens a Help window and displays a list of the main (application-specific) topics.

- *Index*;
  This optional item should yield a full alphabetical index of application-specific terms.

- *Search for Help on*;
  This menu option opens a Help window and displays a dialog box that allows the user to search for Help topics containing specific keywords.

- *Tutorial*;
  A further optional item, *Tutorial* starts an on-line tutorial.

- *How to Use Help*;
  This item yields 'help on help' by opening a Help window and displaying instructions for using the Windows Help system.

While the first three items provide alternative access mechanisms to the on-line documentation, the latter two serve as further information resources.

Although the operation of Windows Help is geared to providing general access at any stage in user-interaction there are attempts to provide context sensitive help. This is most obvious in support of system messages or warnings which are accompanied by a 'help button' (e.g., Figure 1).

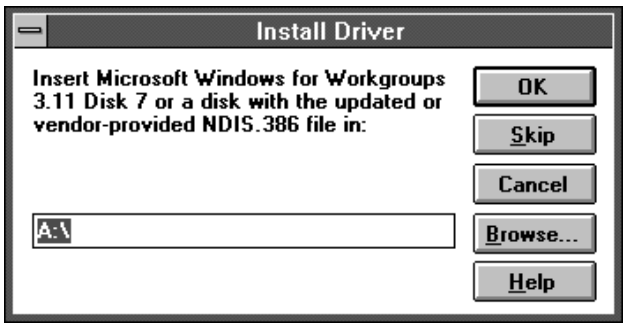



**Figure 1 : System message with context specific *help* option.**

Clicking on *help* puts the user into the Help System at an apposite point (cf. Figure 2).

This approach to context specific user-assistance also indicates that support may be further categorised, e.g., according to whether the user or the application is the initiating agent (cf. Aaronson & Carrol, 1987; So & Travis, 1994).[3]

[3] In the latter case the user support can be further classified by the method used to access help information. These may include the use of keywords, the use of a list of topics or an index (Campagnoni & Ehrlich, 1989), or the use of natural language (as in the Unix Consultant (Wilensky et al, 1984)).

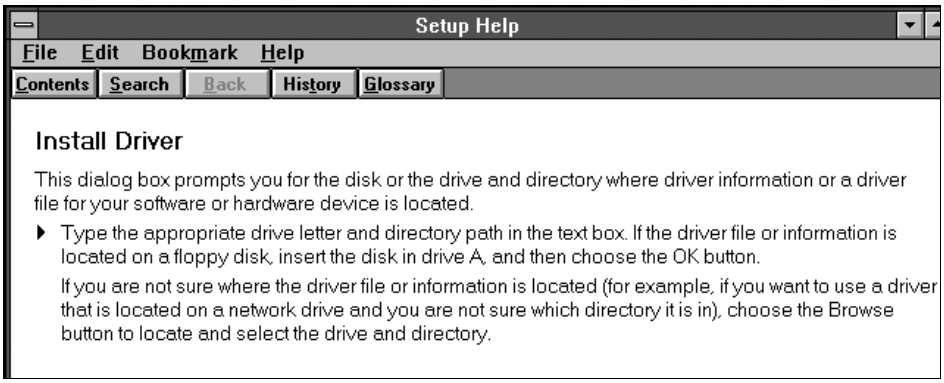



**Figure 2 : Context specific help information.**

Further characteristics of user support systems are addressed by Sukaviriya (Sukaviriya, 1991) who considers aspects such as the help initiator (user vs. computer), the access method (keyword based, natural language, etc.), the help content (conceptual, procedural), the medium of presentation (text, audio, etc.) and the adaptiveness of presentation.

Clearly, there is considerable diversity in the focus of user support and many possible dimensions may be highlighted in its classification. Our simple purpose in this section was to draw attention to this variety and to note that most make no explicit reference to the needs of users whose mother tongue is different from the delivery language of the computerised system. Although steady globalisation in the use of computer applications inevitably generates problems for users with differing native languages, the need for local language support, as an added feature in existing English language systems, is a rarely addressed aspect of help systems. In what follows, we discuss specific approaches to this (and related) problems.

## 3. Support for international users

In a context in which non-native speakers of English may be increasingly obliged to operate with English-based computerised information systems, difficulties in user-interaction are inevitable. Our standpoint on such problems is the reasonable assumption that language itself will introduce particular scope for difficulties and misunderstanding in the use of such information systems. This situation and attempts to alleviate the problem characterise the 'second language problem' for human-computer interaction.[4]

An obvious redress to this problem is to provide local language support. Indeed, this could be carried to the extreme of full language localisation of software. This radical approach calls for re-implementation of the system so as to provide solely mother-tongue information. The main drawback of this solution is its likely cost in terms of time and effort. So this strategy is usually reserved for cases where the user community (and the market potential) can justify this cost. Furthermore, there may be other obstacles in the way of full local language implementation. Lack of access to the software source code can be a major disincentive. Breach of software copyright is a further danger, especially if the existing software is in use solely under license.

The absence of standardised cross-language terminology can be another obstacle. If a standardised terminology does not exist, then the localisation can create confusion among the

[4] In recent years some attention has focused on the issue of 'internationalisation' for computer systems (e.g., Neilson, 1990; Russon & Boor, 1993).

potential users, who are probably accustomed to different terms (this point is considered further in Section 4.2, below).

Finally the full localisation approach may not make sense in cases where the user community is diverse (e.g. communities in universities sometimes consist of users with different mother-tongues). There is an evident need for other ways to address the requirement for other language support.

# 4. Strategies for second language user support.

A range of possible strategies would address the second-language issue. The most obvious being:

1. use a single language common to all;

2. replicate the original language in one or more required tongues;

3. focus other language support on the most troublesome system areas.

## *4.1 Use of a common language*

The first idea that one might consider as a way of lessening the impact of the second-language problem would be the use of a single communication format that is equally comprehensible to all regardless of native language. This is easier to imagine than to achieve.

Perhaps the closest approach to viability would rely on graphical modes of expression. After all, this 'non-linguistic' technique is already commonly employed in direct manipulation user interfaces, where icons represent objects well known to the user, such as the wastebasket or the diskette. The increasing international use of icons (airports and railway stations provide many examples), lends credence to the possibility of this universal picture language approach, but there are serious doubts about it as an adequate solution to the whole second language problem.

A number of potential difficulties confront this strategy. In the first place, it seems unlikely that there are sufficient universally understood signs to convey all that may be required of any single interactive system, especially if it bears any reasonable degree of complexity or sophistication. In the second place, signs often generate problems of ambiguity and discriminability. For example, the iconic pushbuttons illustrated in Figure 3 clearly rely upon a common visual metaphor, yet their application is quite different. The binoculars (Figure 3a) appear in Microsoft Word (version 6.0) as a pushbutton which is used to find a specified text or format. The glasses (Figure 3b) appear in Microsoft Excel (version 5.0) as a pushbutton which is used to display the value of an expression.

(a) (b)

**Figure 3 : Icons related by metaphor**

There is no obvious reason why binoculars should signify a 'search' operation rather than the 'reveal' operation of the glasses (nor *vice versa*). Clearly, there is considerable scope for misinterpretation of such non-verbal expressions. Seemingly well-known and widely used user

interfaces may also nurture problems in symbol recognition. Even apparently obvious iconic signification may be misleading. The warning sign (Figure 4a) is often used to denote a critical message but for Greeks this represents an offensive gesture. To avoid misinterpretation, in this context, a stop sign (Figure 4b) may be used instead.[5]


(a) (b)

**Figure 4 : Warning Signs**

Simple characteristics may also carry unintended connotations, e.g., 'black is a color associated with death and funerals in the West, while for the Chinese, white is the color associated with funerals and red with marriages' (Uren et al, 1993).

## *4.2 Second language duplication*

A straightforward solution to the second language problem lies in providing a dual set of information. English may be the 'standard' output language, but is rarely common currency for non-native speakers of English. Hence for such users, who might be phased by unfamiliar English expressions, we could afford the luxury of 'switching' at any time to output in their native language.

This technique is often valuable where the font characteristics of the local language are significantly different from the ISO Latin standard.[6] Thus, interfaces such as X[7] and Microsoft Windows have been supplemented (or revised) to enable non-English speakers to handle mother-tongue fonts such as Thai, Japanese, Arabic or Greek (Figure 5).[8]

Generally, these systems provide users with 'local' fonts and a means of switching between the 'standard' English and the local keyboard layout. A similar technique has recently been applied to the X version of NCSA Mosaic , although full 'bi-directional'[9] language facilities are presently incomplete.

Less sophisticated techniques can be employed in applications that are not bi-directional. For example, the GWIC system (Weir & Ni, 1993) was written to operate in an environment with no language adaptations or extensions. Chinese text was prepared in a DOS-based Chinese

[5] Neilson cites a further cultural difficulty in which check boxes as interpreted in Japan may convey the converse meaning to that intended by the American designers (Nielsen, 1993).
[6] For most widely used operating systems there exist a number of modified versions that support a second (native) language along with the standard (English), e.g., NK Dos: a modified DOS that recognises and supports Chinese characters (Archer88); IAS (Integrated Arabic System): an enhanced DOS with Arabic command processor and an Arabic command interface (Tayli & Al-Salamah, 1990).
[7] A subset of a layered multilingual input/output system (Kataoka et al, 1992) was implemented in the X Window system as a feature of X11R5. Aktypis has designed and implemented a bilingual enhancement for X Windows, that allows for the Greek character set to be used along with the standard Latin (Aktypis et al, 1993).
[8] Greek MS-Windows is an enhanced version that includes several Greek true-type fonts as well as a keyboard driver for Greek/English/European keyboards.
[9] Bidirectional multilingual systems can handle more than a single language both as output (for display) and as input (for user entry).

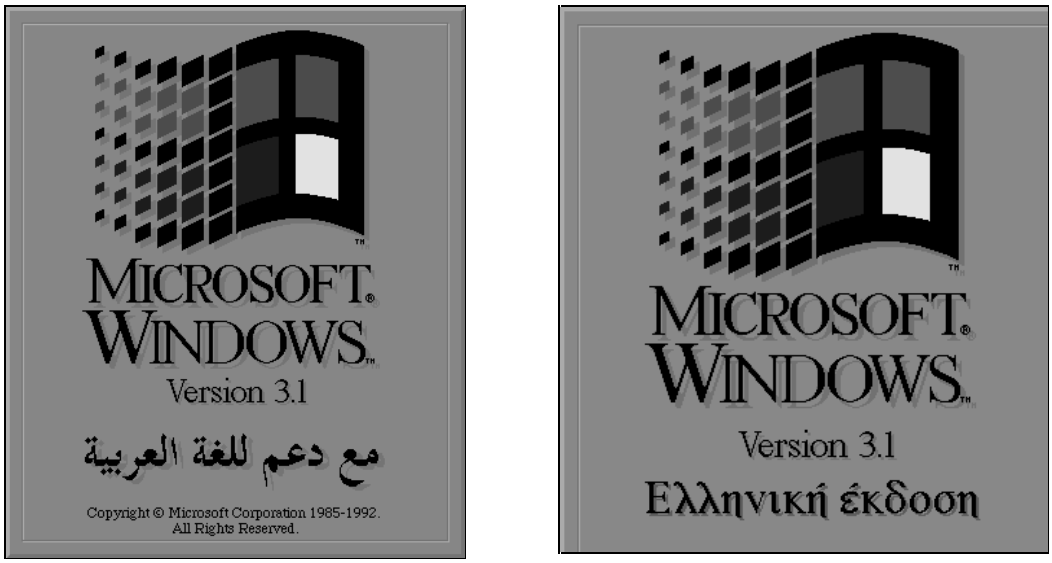



**Figure 5 : Microsoft Windows—Arabic and Greek versions**

word processor and exported as pictures for display in the final application. For exotic font families it may be more convenient to employ digitised handwritten text (Figure 6).[10]

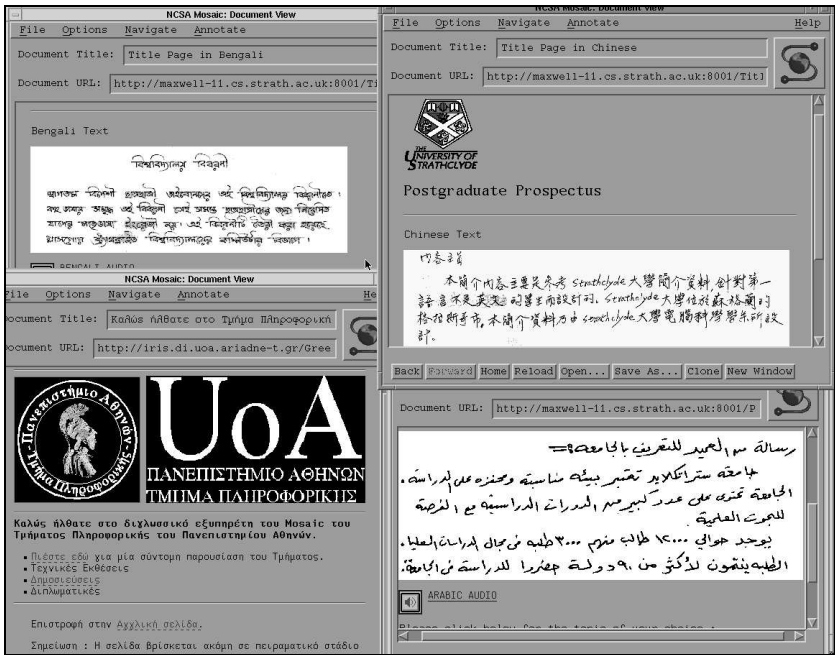



**Figure 6 : Scanned handwritten fonts in NCSA Mosaic**

[10] In this illustration the Arabic, Bengali and Chinese texts are digitised handwriting, whereas the Greek text uses installed Greek system fonts.

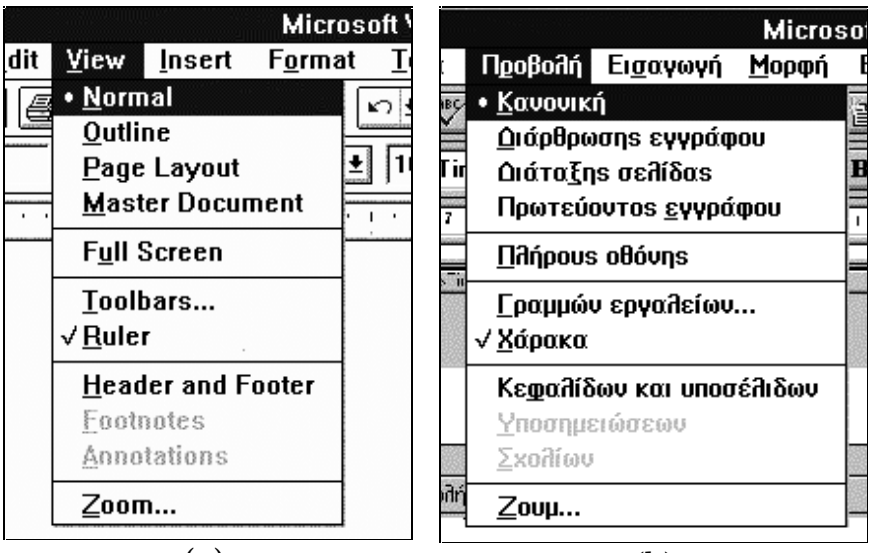
Although such strategies help by supporting local character sets, they do not fully address the second-language problem. The availability of local language fonts leaves untouched the problem of conveying the content of an original English-based application in the local language. Fonts are a vital component but are less than half the story. On the other hand, a complete native language version offers no second language obstacles to the local language user, but would amount to complete localisation of the application. Of course, if there is no second language, there is no second language problem.

As noted above, inherent difficulties may appear when 'converting' existing English-language software to localised versions. The absence of standardised cross-language terminology can mitigate against the merits of local language facilities. One example appears in popular word processor packages that have been localised into Greek.

Figure 7(a) illustrates the *View* menu from Microsoft Word in its conventional (English) version while Figure 7(b) shows the Greek equivalent. The terms 'header' and 'footer' may be familiar to native speakers of English but the choice of Greek equivalent is problematic. Microsoft replace 'header' with Κεφαλιδα and 'footer' with υποσελιδο. Although the former does means 'header', it is an uncommon expression in Greek with no familiar connotation. Native Greek speakers indicate difficulties in making sense of Κεφαλιδα.

The Microsoft choice to replace 'footer' is more familiar to Greek speakers, but conveys the equivalent of 'subpage' in English. Clearly, a burden of interpretation is imposed on native Greek users by this choice of term.

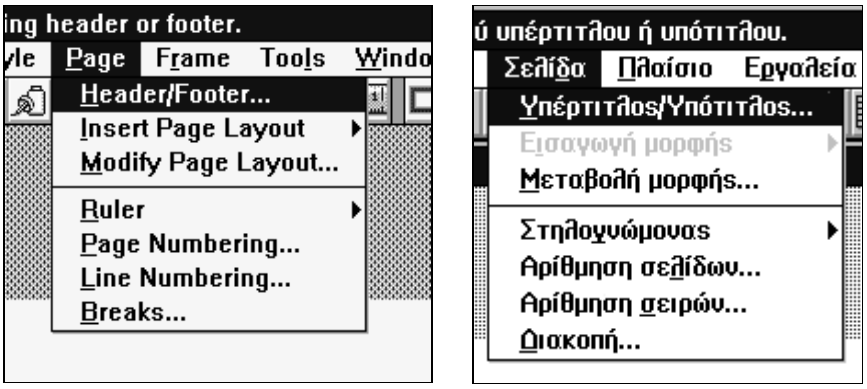
One should not assume from this example that Microsoft have simply ignored the subtlety of Greek language in their menu design. There are difficulties inherent in the translation process. This point can be underpinned by comparing equivalent terms from the AmiPro wordprocessor produced by Lotus.




(a) (b)

**Figure 7 : Microsoft Word 'View' Menu (English & Greek)**

Figure 8(a) illustrates the English version of the *Page* menu which includes the terms 'header' and 'footer'. The equivalent menu from the localised Greek version of AmiPro is given in Figure 8(b).




(a) (b)

**Figure 8 : AmiPro 'Page' Menu (English & Greek)**

Note that the Greek equivalent of 'header' is not Κεφαλιδα but Υπερτιτλος, while 'footer' is represented not by υποσελιδο but Υποτιτλος. Here, the Greek terms literally mean 'supertitle' and 'subtitle', respectively. There can be little doubt that many Greek speakers will experience confusion over this choice of terminology, particularly since the header and footer bear no relevant connection to the title of a document.[11]

In the face of such problems, one may adopt a charitable view and regard such terminological turmoil as the inevitable consequence of misfit in system conversion from English to native language, rather akin to inherent difficulties in translation from one cultural and linguistic context to another. Perhaps the aim of complete localisation is misguided or destined for problems.

With respect to more extensive information exchange between user and computer system, a more realistic approach may seek to provide partial duplication of an English-language original. This holds the prospect of facilitating comprehension where the perceived need is greatest whilst minimising the effort of implementing the native language component.

For existing applications there remains the task of 'attaching' such supplements to the original system software. Work in progress[12] promises a means of replacing or supplementing existing help files in MS-Windows applications with local-language help, thereby retaining context-sensitivity via the 'factory-defined' links to the application.

### *4.3 Second language explication*

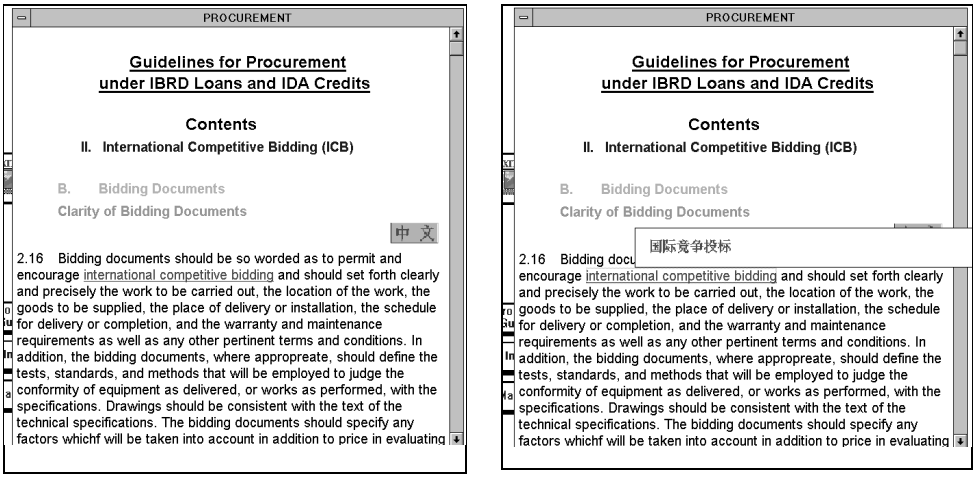
The second-language explication strategy is a more subtle approach to the application of local language supplements. In common with the partial duplication approach, this alternative focuses on 'key' areas of interaction where greatest need for native language support is anticipated.

In some cases the needs of the computer user may be adequately met through literal translation of existing English information. In other cases, e.g., problematic points in user interaction, the best support through native language material may lie not merely with a *translation* of English but in a first-language *explication* of the original English language material.

---

[11] These examples are drawn from Lepouras G. & Weir, G., 'Its not Greek to me: a cross-language comparison of Word Processor Interfaces', (in preparation).
[12] Lepouras, G., et al, *Re-attached help for Windows* (in preparation).

This selective technique has been applied in the GWIC system (op.cit.). In this case idioms or technical expressions that may only have ready significance for a native English speaker are not simply translated but explicated (more fully explained) in Chinese. Figure 9(a) shows an example in which the phrase 'international competitive bidding' is highlighted in red to indicate that an explanation in Chinese is available. This explication is shown as a pop-up in Figure 9(b).




(a) (b)

**Figure 9 : Example of Chinese explication**

Obviously, systems which are essentially concerned with delivering information place more stringent demands upon the English comprehension skills of the second-language user. For this reason the explication strategy may be extended in order to obviate any remaining risk of language failure on the part of the user. One simple yet powerful extension is the provision of brief summaries for particularly salient information. An example of this technique (also from GWIC) is shown in Figure 10. Here, a whole page of English text is supplemented with a pop-up native language summary that re-iterates the most important points in the output.


**Figure 10 : Native language summary of salient points**

In our view, so long as the target user population has reasonable proficiency in English, the combination of flexibility and economy of effort renders the selective 'explication' approach to second-language support more appropriate for localising existing English-based systems.

One difficulty inherent in the explication strategy lies in finding adequate bases for choosing the contexts in which to offer native language explication. Two possible sources for such criteria come to mind:

- applied linguistics;
- human-computer interaction.

### 4.3.1. Linguistic approach

Many non-native users of English will experience difficulties in interaction with English-based information systems. Aside from any intuitive plausibility in this claim, this is in keeping with views expressed by the EFL[13] profession. For example, Alderson notes the common experience of EFL teachers that 'most students fail to learn to read adequately in the foreign language' (Alderson, 1984, p.1).

Since the language employed in any interactive software application is a subset of the 'natural' English language one might incline to the view that the linguistic requirements for computer applications are an example of 'English for Specific Purposes' (ESP).[14] In fact, this is barely plausible unless one construes the complete set of possible computer applications as a single language domain. Such applications range far and wide in their purpose and share a common mode of interaction rather than a common set of linguistic objectives. Despite this, the comprehension requirements for interactive systems that rely on text are plainly affected by the user's language proficiency.

Some (though not all) interactive difficulties will arise through failures in comprehension and such instances are a prime candidate for mother-language support. If we gain insight into the likely contexts for such language-based errors we should be better equipped to give support (either in the form of native language duplication or as native language explication).

The major source for such insight must be empirical analyses of typical software interaction, i.e., performance evaluation on non-native speakers of English using English-based software systems.[15] In the absence of such experimental information we can turn to existing work on second-language use and comprehension.

Boulton describes several causes of cross-language misunderstanding and identifies four common characteristics (Boulton, 1973):

- malapropism;
- differences in definition;
- differences of association;
- misunderstanding of the context;

[13] English as a Foreign Language.
[14] For background on ESP pedagogy see Kennedy & Bolitho, 1984.
[15] We are currently conducting a survey of Greek users.

Each of these scenarios describes a basis for miscomprehension. In the first case, the second-language user confuses one term with another that is morphologically similar but semantically dissimilar. For example, the English word 'routing' might be confused with 'routine' which looks similar but means something quite different. Malapropisms are likely to be common among second-language readers since such users characteristically attend to 'surface' features of the lesser known language.

Misunderstanding may also arise through differences in the definition of words. This may occur in cases where one term has frequent use but diverse applications. For example, the word 'server' as applied to X windows applications may readily be misconstrued by someone who is acquainted solely with distributed file systems.

Differences in association relate to the connotation of words. A native language speaker may be aware of connotative aspects that are lost on the second-language user, though the latter may be familiar with the foreign word or expression in question. Finally, the context in which terms are encountered may influence the interpretation of second-language expressions.

Beyond these potential comprehension failures a number of general factors are likely to cause difficulties for non-native users of English. These include 'special use' expressions, (e.g., jargon, slang, or idioms) which are likely to prove unfamiliar or elusive for second-language users. Likewise, acronyms depend upon the readers familiarity with their expansion. Such abbreviations are a compressed form of language but for non-native speakers of English acronyms are a compressed *foreign* language.

Once more, words that do not translate directly into the user's mother tongue are potential stumbling blocks. Thus, the phrases 'zoom in' and 'zoom out' have no direct equivalents in Greek. On the other hand, the terms 'landscape' and portrait' do have direct translations but the Greek equivalents do not convey the original meaning. Informal evidence indicates that Greek users find difficulty in understanding the Greek versions of these terms.

Even between American and British English there are instances of apparently identical words bearing different meanings[16] or cases of what is acceptable in one language being vulgar in the other. This is a case of 'interference' in which the reader's interpretation of the second-language is influenced (and often corrupted) by the native language.[17] A simple illustration of such interference is given by Figure 11, which shows the brand name of a washing powder.


**Figure 11 : An example of 'interference'**

Most of us could be forgiven for assuming that the product named in this artistic script is called 'USA'. In fact the powder in question is named 'Breeze'. This can be more readily

[16] For example, British designers must be sensitive to the American distinction between a rubber and an eraser.
[17] Yorio claims that 'the reading problems of foreign language learners are due largely to imperfect knowledge of the language, and to native language interference in the reading process' (Yorio, 1971; cited by Alderson, op. cit.).

accepted when one appreciates that Figure 11 does not portray the Latin characters for ‘USA’ but four characters written in Thai, whose pronunciation is a transliteration of the English word ‘Breeze’. The non-Latin nature of the script is apparent when fuller context is given (Figure 12).


**Figure 12 : ‘Breeze’, a Thai soap powder**

Naturally, we colour our interpretation according to our own background and experience. In the context of second-language interpretation such experience can become interference and lead to error. Indeed, the scope for such interference should not be underestimated. Steffensen and Joag-Dev note that

> ...TESL [Teaching English as a Second Language] and foreign language pedagogy has moved away from the idea that comprehension is in some sense present on the page and is recognizing the creative contribution made by the reader. Interference is now understood as extending beyond the affective domain to the denotative values of words, and the propositional content at the sentence and text level (Steffensen & Joag-Dev, 1984, p.61).

We believe that further research into the second language problem will highlight specific difficulties faced by such users in coping with English-based software systems. Beyond this research, we can also turn to other work within Human-Computer Interaction as a further source of guidance on how to locate key points for second language support.

### 4.3.2. HCI approach

Aspects of system operation that most commonly give rise to problems in interaction provide a clear directive for user support. Naturally, this may vary from application to application but most interactive systems explicitly address the need for user support. Thus, as one might target information or advice to the *ordinary* system user (e.g., in the form of on-line help) at those points in user-interaction considered most prone to difficulty, so, with the *extraordinary* system

user who may require second-language support, one would aim the native language supplements at precisely those points.[18]

Here, our guidance on the application of second-language support derives from available methods of determining the usability of the interface. Obviously, any workable method may be called into play, e.g., heuristic, formal or empirical evaluation (cf. Neilsen & Phillips, 1993).

A variety of techniques have been described in the literature, e.g., the Cognitive Walkthrough method (Wharton et al, 1992). This form and task-based methodology aims to detect and correct defects in the user interfaces by focusing on a user's cognitive activities. Alternatively, Neilsen concludes that heuristic evaluation is likely to locate major usability problems more readily than minor problems, so additional measures should be taken to find problems relating to exits and user errors (Nielsen, 1992).

A fast-paced user interface evaluation procedure (the cognitive jogthrough) has been developed by Rowley & Rhoades (Rowley & Rhoades, 1992). This reveals significant user interface problems that can then be studied using other techniques. According to Karat et al,

> the empirical testing condition identified the largest number of problems, and identified a significant number of relatively severe problems that were missed by walkthrough conditions (Karat et al, 1992).

Clearly, estimating the usability of a user interface is a painstaking and complex task. In cases where 'add-on' second-language support is proposed the original application will be in use already so feedback from initial users' experience should be available to assist in identifying areas of difficulty as targets for language support. Once this has been achieved, there exist several techniques that can be used separately or in combination, in order to focus reference material in the native language. Some of the alternatives have been outlined in the present paper.

## 5. Conclusions

This paper has characterised the second language problem as a special case for user support. Although this problem is determined in large part by a user's native language, the issue does not lie solely within the realm of linguistics. Applied language studies promise insights on the characteristic interpretative errors that may arise for a second-language user and this can provide focus for the practical effort of honing a user interface design to suit its intended (foreign) user base. More generally, the user support issues raised by the problem remain firmly located within the realm of Human-Computer Interaction.

The insights from this work have two areas of application. Firstly, sensitivity to the vagaries of second language users can serve to mitigate against significant (language-based) problems in interaction. Thus,

> Text developers can perform an important service by employing writers with a detailed (or native) knowledge of the student's cultural background to produce reading materials and by using ethnic reviewers to screen out potential misunderstandings (Steffensen & Joag-Dev, 1984, p.61).

[18] Significantly, a number of authors have proposed enhancements to on-line help that entail greater linguistic sophistication on the part of the user (e.g. Gwei & Foxley, 1990). This has the natural corollary that any attempted native language support should be equally sophisticated.

Secondly, empirical investigation has its place in revealing the loci of language-based difficulties and can usefully feed back into the design of more amenable localised systems.

Clearly, more is required of localised software systems than simple translation. An adequate localisation will accommodate the range of support needs that arise from the second-language problem. We contend that this is best addressed through finely-tuned selective use of native language support rather than by the blanket approach of full localisation.

## 6. References

Aaronson, A., Caroll, M.J., 'The answer is in the question : a protocol study of intelligent help'. *Behaviour and Information Technology*, vol. 6, No. 4, 393-402, 1987.

Alderson, J.C., 'Reading in a foreign language: a reading problem or a language problem?', in Alderson, J.C. & Urquhart, A.H. (eds), *Reading in a Foreign Language*, Longman, London, 1984.

Archer, N.P., Chan, M.W.L., Huang, S.J. and Liu, R.T., 'A Chinese-English microcomputer system'. *Communications of the ACM*, vol. 31, No 8, 1988.

Aktypis, A., Georgiadis, P., Kolomvos, N., Lepouras, G., 'A bilingual enhancement of X Window System', *Proceedings of the 4th Panhellenic Conference on Informatics*, University of Athens, 1993.

Boulton, M., *The Anatomy of Language: Saying what we mean*, Routledge Kegan Paul, London, 1959.

Campagnoni, F.R. & Ehrlich, K., 'Information retrieval using a hypertext-based help system', *ACM Transactions on Information* Systems, 7, 3, 271-291, 1989

*The GUI Guide*, Microsoft Press, Washington, 1993.

Gwei, G.M & Foxley, E., 'Towards a consultative on-line help system', *International Journal of Man-Machine Studies*, 32, 363-383, 1990.

Houghton, R. C., 'Online help systems: a conspectus', *Communications of the ACM*, 27, 2, 126-133, 1984.

Karat, C-M, Campbell, R. & Fiegel, T., 'Comparison of empirical testing and walkthrough methods of user interface evaluation.', *Proceedings of CHI'92*, ACM Press, 397-404, 1992.

Kataoka, Y., Morisaki, M., Kuribayashi, H., Ohara, H., 'A model for input and output of Multilingual Text in a Windowing environment', *ACM Transactions of Information Systems*, vol. 10, No 4, pp 438-451, 1993.

Kennedy, C. & Bolitho, R., *English for Specific Purposes*, MacMillan, London, 1984.

Myers, B.A., 'Challenges of the HCI Design and Implementation', *Interactions*, vol. 1, 1994.

Nielsen, J., 'Finding usability problems through heuristic evaluation', *Proceedings of CHI'92*, ACM Press, 373-380, 1992.

Neilsen, J. & Phillips, V.L., 'Estimating the relative usability of two interfaces: heuristic, formal and empirical methods compared.', *Proceedings of InterCHI93*, ACM Press, 214-221, 1993

Rowley, D.E. & Rhoades, D.G., 'The cognitive jogthrough: a fast-paced user interface evaluation procedure', *Proceedings of CHI'92*, ACM Press, 389-395, 1992.

Russon, P. & Boor, S., 'How fluent is your interface? Designing for international users', *Proceedings of InterCHI'93*, ACM Press, 342-347, 1993.

So, B. & Travis, L., *A step toward an intelligent UNIX help system: knowledge representation of UNIX utilities*, Technical Report, Computer Sciences Department, University of Wisconson-Madison, 1994

Sukaviriya Piyawadee "Noi", *An Integrated Taxonomy of On-Line Help Based on User Interface View*, Technical Report, Graphics, Visualization and Usability Center, Georgia Tech, October 1991.

Tayli, M. & Al-Salamah, A.I., 'Building bilingual microcomputer systems', *Communications of the ACM*, vol. 33, No 5, 1990.

Uren, E., Howard, R., Tiziana, P., *Software Internationalization and Localization*, Van Nostrand Reinhold, New York, 1993.

Weir, G.R.S. & Ni, X.Q., 'Second-language user support', Research Report No. HCI-03-93, Department of Computer Science, University of Strathclyde, Glasgow, UK, 1993. (This document is available by *ftp* from ftp.cs.strath.ac.uk in the directory 'research-reports'.)

Wharton, C., Bradford, J., Jeffries, R. & Franzke, M., 'Applying cognitive walkthroughs to more complex user interfaces: experiences, issues and recommendations.', *Proceedings of CHI'92*, ACM Press, 381-388, 1992.

Wilensky, R., Arens, Y., and Chin, D., 'Talking to UNIX in English: An Overview of UC'. *Communications of the ACM*, 27, 574-593, June 1984.

*The Windows Interface. An Application Design Guide*, Microsoft Press, Washington, 1991.

Yorio, C.A., 'Some sources of reading problems for foreign language learners', *Language Learning*, 21, 107-115, 1971.