In: Proceedings of the Language Engineering Convention, 16-18 October 1995, London, pp. 71-78.

GLOSSASOFT : METHODS AND GUIDELINES FOR SOFTWARE INTERNATIONALISATION AND LOCALISATION

Ray Hudson, The Open University, UK and Aarno Lehtola,
VTT Information Technology, Finland

1. Introduction

The Glossasoft project provides practical help to software companies to enable them to exploit the business opportunities provided by a wider, international market. To take advantage of these opportunities, the needs of an international customer base must be taken into account in the development of new products and in the adaptation of existing products. This involves designing and implementing products so that language and other cultural differences are taken into account in the user interface, documentation and packaging of the product.

When software is developed, its design is inevitably influenced by the culture and native language of its developers. To adapt software successfully for international markets, the culturally and linguistically-dependent parts of the software must be isolated, a process referred to as *internationalisation*. These parts include text manipulation and display, character-encoding methods, collation sequences, hyphenation and morphological rules, formats used for numbers and dates, as well as subtler cultural conventions such as the use of icons, symbols and colour. The local market requirements for these items are encapsulated in the term locale.

So, to achieve successful internationalisation, product design should be linguistically and culturally-independent, making products ready for cost-effective adaptation to the locales of the target markets - the process of *localisation*. Localisation is the opposite of internationalisation, taking an internationalised product and adding features to match it to the language and culture of the target market. It does not just consist of translating menus, commands and messages into the required language, but includes adaptations for the culture of the target country, and particularly its business practices and culture. We refer to the process of decision-making, planning, internationalisation and localisation as the *globalisation* process.

2. Globalisation

To be successful, any globalisation effort needs to start with an investigation of customer needs in the target markets. Various techniques can be used, from simple structured interviews of large samples of users to lengthy discussions with focus groups. Customer needs, the implications for the design or redesign of the product to satisfy these needs, together with the data on market sizes, trends and potential, drive the process of estimating the costs and likely revenues obtained from offering a localised product version for a market or group of markets. The most important factor in the decision to localise concerns the financial investment involved. Cost and revenue schedules, which build up the cash flow picture for the project, can be used in various investment analysis methods to facilitate decision-making. Cash flow data can also be modified for a series of different internationalisation and localisation strategies in order to choose the best option.

The number of potential markets will determine whether it is desirable to localise on a one-off basis per market or to develop or re-engineer an internationalised base version from which multiple localised versions can be derived. The benefits of choosing the latter approach are a faster time to global market, reduced cost per localised version, simplified support and maintenance and a better quality product.

The approach we recommend is to develop an international base version of the software. This version can be described as being ‘enabled’ for localisation as it results in more efficient and effective localisation. The basic idea in the design of this base version is to isolate and separate the elements of the product that require changing into discrete components. The appropriate components for particular target markets are then ‘plugged in’ during the localisation process. An example of this might be a message catalogue that is opened via a language control variable to supply the appropriate local language versions of textual parts of the user interface.

For new software, the customer's needs for localisation can be integrated into the development process at the requirements analysis stage, where they will influence the overall design of the software. In very general terms, the internationalisation process for existing software involves investigating the software to locate assumptions about language and culture. Reverse engineering, restructuring and re-engineering techniques and tools can then be used to identify and separate out elements that require localisation.

2.1. Satisfying language needs

The most fundamental level of language-related requirements for localisation is at the data level, where the product must be able to input, process and output text (and speech in special cases) for the local language. Depending on the nature of the application and the languages to be supported, localisation at this level can involve significant effort. The next localisation level is the language used for the messages in the user interface. Examples of items that need to be localised are menus, status bars, dialog boxes and special key assignments. This text localisation can lead to resizing problems as translation often results in longer message texts. Other elements of a software product, which are referred to here as product support, may also need to be localised. Some examples are:

- On-line documentation: help systems, tutorials, examples to illustrate how the product works, installation procedures and 'read_me' texts.
- Hard-copy documentation: user manuals, reference manuals, training manuals, quick-reference guides, special application keyboard templates, marketing and sales literature, installation guides and 'getting started' manuals.

There can be profound differences in style for such items between different cultures so localising all these items can constitute a large part of the effort when localising a package, and can significantly increase the cost and time required. Selective localisation of these items should be considered. For example, if English is a common second language in the target market, would some of the documentation be acceptable in English? For the data and user interface levels, if users need that level, everything in the product corresponding to that level must be localised. At the product support level, it may not be necessary to localise everything - leaving some technical documentation untranslated may be acceptable, depending on the application and the target market.

2.2. Satisfying cultural needs

Cultural items include the use of symbols, icons, graphics, the style of the text in the user interface and product support, the use of humour, the use of colour and sound, and general business infrastructures and practices. As an example of cultural differences, in Western countries red is often used to signal danger, while in China it signals joy. Culturally dependent icons, such as mail boxes or waste baskets may be misinterpreted in locales where these items do not follow the same standards.

2.3. Satisfying the needs of local practices and conventions

The main areas include the presentation of numeric data, numeric and other separators, the presentation of monetary data and measures such as weight and length, the formats used for dates, times, page sizes, names and addresses, holidays, telephone numbers and so on. These needs are easily determined from existing reference material or from customer feedback. Application-specific local practices in the target market are often specified by legal and professional standards bodies, and may involve unrelated areas such as communication interfaces, laws and regulations such as statutory holidays, and financial accounting rules such as monetary rounding, revenue and tax computation.

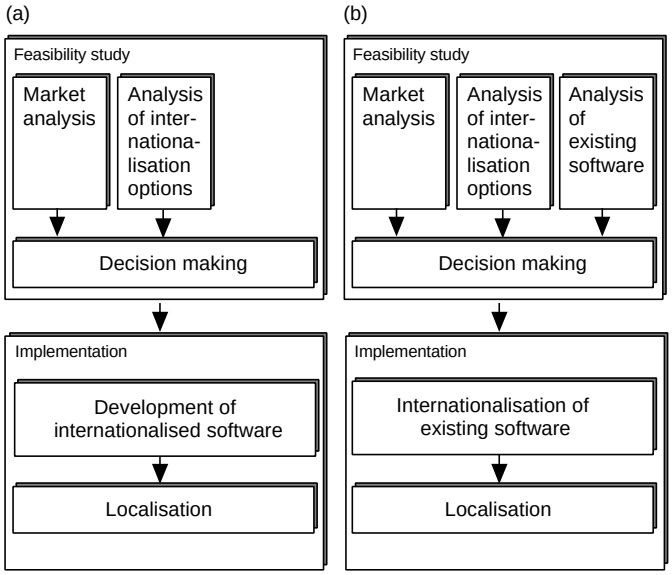
2.4. The Glossasoft globalisation method

Figure 1 outlines our suggested globalisation method. Each phase requires different expertise. Market analysis and decision-making requires a knowledge of business issues. Analysis of internationalisation options, analysis of existing software and the development of internationalised software requires expertise in software engineering. Localisation requires skills in translation and linguistics. The feasibility study is designed to answer the following questions:

- Is it profitable to localise the product for particular markets?
- In the case where an existing product is localised, should it be done as a single task or after an internationalisation stage?
- In the case of new software development, which aspects of the product should be internationalised and to what extent?

The implementation phases are discussed in Sections 3 and 4.

Figure 1. *Idealised phases of internationalisation and localisation while (a) developing new, internationalised software, and (b) internationalising existing software*



3. Internationalisation

Next we discuss the implementation of software internationalisation. The first subsection describes a Framework for Global Software, which provides guidelines for software developers on how to develop and implement internationalised software. An outline Internationalised Software Design Architecture follows. Internationalisation of online and printed documentation is discussed at the end of the section.

3.1. Framework for Global Software

Our Framework for Global Software (FGS) proposes a service architecture with *international application programming interfaces* (IAPI) that assist software development for global markets (Figure 2). In FGS, application functionalities have been clustered into six groups of interrelated services. Table 1 illustrates these groups. The functionalities covered are listed for each group, and those that involve locale specific behaviour are written in bold face. In FGS a new concept, LocaleContext, addresses the problems of parameterising locale specific behaviour and realising locale conscious data structures. A LocaleContext is an object which can be defined from scratch or by overriding some properties of an already existing LocaleContext. LocaleContext is used in various locale specific service calls. Hyphenation, spelling, and grammar checking of multilingual documents with varying script systems are examples of some of these calls.

Figure 2. The structure of FGS.

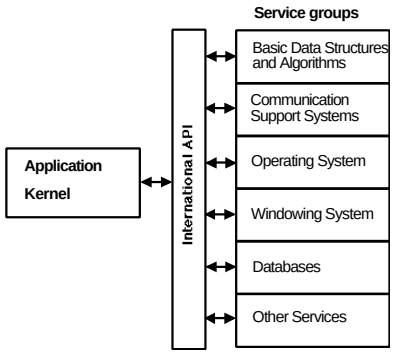
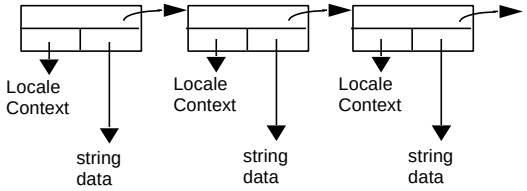


Table 1. Service groups in FGS.

Basic Data Structures and Algorithms	locale context **characters, strings, streams** **string search, sorting, comparison, case conversion** **spelling, hyphenation** hash tables, lists, containers data compression, data encryption
Communication Support Systems	**messaging interfaces (electronic mail, Edifact)** protocol interfaces (sockets, X/Open Transport)
Operating Systems	processes and **interprocess communication** system configuration, memory management, resources (device drivers etc.) **libraries** **file system** **object management support** **basic I/O** **clock and calendar** **command interface**
Windowing System	**windows, views** event handling **controls (including text editing)** **cursors, mouse handling, beep/flash** interclient communication (DDE, drag and drop, clipboard services) **graphics, icons, colours** **fonts, keyboard drivers** **voice and sound**
Databases	**highly structured databases (e.g. relational databases)** **hypermedia databases**
Other Services	**various servers (help engines, dictionaries)**
	document management facilities (SGML, ODA, OpenDoc; indexing facilities)
	security
	debugging support

Figure 3. An internationalised string.



As an example of internationalised data structures and their usage, Figure 3 illustrates an internationalised string which consists of segments that can be in different locales. Internationalised string processing services are needed to process such strings. Windowing system services would need to include internationalised controls. An internationalised menu may contain items in different script systems. An internationalised text edit control needs to cope with different script systems in different parts of the document. Writing direction, font, and keyboard map would be selected depending on the LocaleContext.

The general format of an internationalised control constructor function in the C language could be:

id CreateIControl (Frame &frameGeom, Position &position,
id fatherId, Style &styleStructure, LocaleContext lc);

Related accessory functions would be enhanced to handle internationalised data structures. Their usage could have side effects on the controls due to locale changes. A simple function for setting window titles, **SetITitle(id controlId, IStringId titleId)**, could be defined to effectively change the major locale of the window according to the internationalised input string parameter. This would be reflected in the menus and status bars of the window. These examples demonstrate the semantic richness that may be involved in internationalising different services. Moreover, it gives an idea of how powerful the usage of LocaleContext could be. A LocaleContext can control:

- **Language related things:** alphabet, primary writing direction, month and day names and abbreviations, ordinal abbreviations, spell-checking, hyphenation, and other linguistic aids etc.
- **Text processing functionality:** character sets and associated fonts, conversion functions (upper/lowercase, diacritical/accent removal between character sets), collating sequences etc.
- **User interface text and control:** error, help, system texts, flow control, command parsing tables, recognition logic for commands, searches etc.
- **Country:** keyboard control (key sequence-to-function mapping, keyboard drivers), other device control (e.g. print control mapping), time transformation (calendar offset from GMT, zone name, zone abbreviation, daylight savings time), currency formats (currency symbol, negative currency indicator), data representations (decimal separator, thousands separator, list separator, default formats for time, date, phone number, and addresses) etc.

Currently there are no comprehensive industry standards available for an IAPI. Fortunately, standards with limited coverage are available (e.g. Unicode, NLS of X/Open) and can be used to implement parts of an IAPI. In everyday work, solutions have to be based on the available APIs provided by systems software and licensed third party tools such as database access and GUI libraries. Many internationalised services can be implemented by introducing extra mapping layers on top of the existing services. However, some of the services are beyond this layering approach and need profound functional extensions to become internationalised (e.g. an internationalised text edit control).

The specifications of FGS can be used as a model of how an internationalised application could be made to interface to external services, how locale-specific features could be parameterised, and how locale consciousness could be handled in data structures. The IAPI of FGS provides guidelines and examples on how and where the borderlines could be set, when language and culture-dependent parts are detached from those parts of an application that are universally applicable. FGS can be used as a general guideline in the design of internationalised applications.

3.2. Internationalised Software Design Architecture

We have also defined an internationalised software design architecture (ISDA). An application developed in compliance with ISDA is built on a layer of tools and utilities, including the API for internationalisation. The

developer decides on the amount of hard-coded information in the core of an application, according to the nature of each specific application.

The program core calls functions in external libraries. These libraries enable functions and whole libraries to be replaced or added without the need for changes to the program core. The libraries contain functions that perform locale-sensitive operations, such as device recognition and management, display and printing, input management, language and culture-dependent processes and so on. The operation of these functions can be customised from external configuration or resource (C&R) files. The library functions read these files and the file contents modify the behaviour of the application according to the needs of the current locale.

ISDA provides the centralised management of the C&R files with a utility called the Configuration Management Utility. The Configuration Management Utility may be offered either by the application or by the operating or windowing system. The end-user, system manager and local vendor/localiser can modify C&R files, thus altering the behaviour of the library functions and consequently the appearance and behaviour of the application.

3.3. Internationalising on-line and hard-copy documentation

For on-line and hard-copy documentation, a controlled or simplified language, with restricted syntax and vocabulary, reduces the amount of ambiguity in technical texts and will facilitate computer-aided translation. The following points should be considered to determine whether a controlled language is suitable:

- The volume of materials to be translated
- The number of target languages
- The need for simultaneous translation or quick turnaround
- The feasibility of controlling the creation and editing of source material
- Whether texts should be written directly in the controlled form, or written in a free form and then ‘translated’ into the controlled language
- How to implement the use of controlled languages in the document production process, and how to persuade and train writers to write in a controlled format

Generally, if the text is written in a free form and then completely rewritten in a controlled form, the style is radically changed. If the original text has been written in a particular style, it is difficult to apply the rules afterwards and preserve the original style. A compromise can be acheived by making limited changes to the original while trying to apply some rules of the controlled language.

4. Localisation

The following points need to be considered in the localisation process:

- The strategies to be adopted to implement the localisation process, the different ways of organising the localisation work and the various roles in a localisation group
- The issues to take into account during the implementation of the various localisation tasks
- The tools that can assist the implementation of these tasks

The current industry trend is to use external localisation service providers for translation to avoid the high fixed costs of using in-house translators, and to use translators who are based in the target markets who know the up-to-date usage of particular languages. These approaches still mean that there are management, control and other issues which need to be tackled for these strategies to work effectively. The contents of the localisation kit provided to the outside vendor and the exact contents of the material that the vendor should deliver must be clearly specified when outsourcing some of the localisation tasks. The basic localisation strategies are:

- In-house localisation
- Localisation using outside support by outsourcing and freelancing

To decide which to choose, various trade-offs should be considered:

- To keep localisation staff in-house, there must be enough localisation projects to keep staff busy.
- An experienced translation house might provide both administrative and localisation services for the target market. In this case, the overall management and quality control tasks remains in-house.
- By maintaining in-house staff, consistent quality may be achieved more easily for all localised products.

4.1. Localisation tools

Some of the most important tools are those to control the terminology used in the product. Terminology should be fixed and translated before the software is localised. Inconsistencies in terminology can result in poor quality and higher costs. The selection and translation of terminology should be done as early as possible to save time and money. During the translation process, translators verify the translated terms and translate untranslated ones. Any checks concerning the appropriate use of terminology are done, and subsequent changes are propagated throughout the product.

Translation workbench products allow the reuse of already translated words and phrases that have been stored in a translation memory. Their use lowers the cost and time involved in localisation. Translation workbenches are becoming cheaper, and companies can use them for translating software efficiently even in-house. These tools make the translation cycle shorter due to the facilities they offer, and lead to better control and elimination of the communication problems between staff carrying out localisation. Some of these products facilitate project control and allow the import and export of translation memories and terminology lists. A translation workbench comprises a set of tools that support localisation. A translation workbench aims to provide users with the following benefits:

- Cheaper and faster translation
- Better translation by making the best use of language resources and language technology tools
- Better control of the localisation project and reduced communication during the translation process

To set up a translation memory or database, matching pairs of text segments in the source and target languages are aligned. Alignment should be validated to ensure the quality of the translation memory. During this stage, segment meanings should be analysed. If segment meanings do not correspond properly, the necessary segments should be split or joined to ensure good translations. The final translation and the source text are imported into the translation memory, with the relative information about product, document and so on being included. The documents and text segments to be translated, together with the appropriate terminological bases, should be sent to the translation service supplier. The result should be verified, post-edited and approved. The final translation is recorded in the appropriate translation base for future use.

5. Conclusions

The issues involved in software internationalisation and localisation are diverse and they cover the areas of business, software engineering, linguistics, and translation. We have explained the problems involved and given possible solutions, presenting basic alternatives and likely pitfalls. One of the classic ways of increasing revenues at relatively low risk is by extending products to new markets. Software globalisation falls into this category. However, if companies want to take this route, they must offer high-quality local products and keep the costs involved in this process to a minimum. Product life cycles are getting shorter, so faster times-to-international market are vital if advantage is to be taken of these opportunities.

The best way to do this is to take international customer needs into account in the initial product design. If this is not done, we have shown how an existing product can be adapted to satisfy these needs. We have described methods, tools, approaches and guidelines for doing this. From an international base version with no assumptions about the target language and culture of the end users, the product can be localised efficiently and effectively for multiple markets.

Much has been achieved in the area of guidelines and tools for internationalisation and localisation, but anecdotal evidence from a number of companies points to the fact that one of the key issues is getting developers to conform to them. This implies that different organisational structures and controls are needed.

References:

Glossasoft Consortium, Software Design for Globalisation: A Practitioner's Guide, Springer, to be published in Spring 1995

Unicode Consortium, Caldwell, J.T. (1991). *Unicode.* Characters and Computers, Main V.H. and Liu Y. (Editors), IOS Press, 1991.

X/OPEN Company (1993). *Internationalisation Guide Version 2.* X/OPEN Company Ltd., Reading, U.K.

Acknowledgements:

The authors wish to thank Professor Pat Hall, Dr. Costas Spyropoulos, Dr. Stavros Kokkotos, Francis Magann, Nikos Stamatakis, Rafik Belhadj, Timo Honkela, Sakari Kalliomäki, Krista Lagus and all who contributed to the Glossasoft project. We would also like to thank the Glossasoft Interest Group members: DCW Hellas (Greece), Digital (UK), IAC (USA), IBM Hellas (Greece), IDSIA (Switzerland), IXI (UK), KT-Datacentre (Finland), MEMOYEC (Greece), Mendez Translations (Belgium), Microsoft WPG (Ireland) and Microsoft Hellas (Greece), Oracle (UK and Finland), Rank Xerox (UK), SENA (Greece), SRI International (UK), Western Systems (Finland), and VTKK (Finland).

Contacts:

Ray Hudson, Faculty of Maths and Computing, The Open University, Walton Hall, Milton Keynes MK7 6AA, UK
Tel: +44 1908 653096, Fax: +44 1908 652140, Email: r.hudson@open.ac.uk

Aarno Lehtola,
VTT Information Technology,
PB 1201, FIN-02044 VTT, FINLAND
Tel: +358 0 456 6032, Fax: +358 0 456 6027, Email: aarno.lehtola@vtt.fi

The Consortium partners can be contacted by electronic mail on the Internet at:
glossasoft.cons@iit.nrcps.ariadne-t.gr