Working Paper Series
ISSN 1170-487X

Software for the Rest of the World

by Alvin Yeo and Robert H. Barbour

Working Paper 96/2

February 1996

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Software for the Rest of the World

Alvin Yeo and Robert H Barbour

Department of Computer Science, University of Waikato, Hamilton New Zealand

Email: r.barbour@waikato.ac.nz

Abstract

A survey is made of facets of the process of providing software across national and cultural boundaries. Internationalisation (i18n), localisation (l 10n) and globalisation (g11n) are identified as three descriptors for recent Information Technology developments in this field. Current practice and advice for successfully providing software in places other than countries and cultures of origin is reported. Suggestions for further work are made in the light of the survey.

1. Introduction

Information Technology (IT) is being adopted by users all over the world. Software users are not limited to the English speaking West but include people from all language groups and cultures. Kay (1994) reports "international sales make up more that half of the revenues for the top 100 US software companies". Reduced demand in Western nations has provoked software developers to widen markets and amortise development costs by attempting to meet the needs of all users from diverse cultures at the scale of major language groups. Software developers have also attempted to customise their products to meet the needs of specific communities and cultures. In this paper we describe the current strategies and approaches used in the design of software intended for use in global markets. A check-list of cultural attributes that need to be addressed during the software development process is provided. Possible future approaches to the globalisation of software will also be discussed.

2. Key concepts and Definitions

Software that is designed to accommodate the needs of many cultures meets design criteria through internationalisation or the process of making a system or application software independent of or transparent to natural language. If a system can support any language, then it is fully internationalised: if it supports only a limited subset languages, then it is partially internationalised (Unicode and Internationalisation Glossary, 1995). Preparing software in this way is usually carried out in the country of origin of the software (Russo and Boor, 1993). The modifications consisting of the removal or replacement of culture-specific features in the original software (Russo and Boor, 1993). Internationalisation is also known as i18n: there being 18 letters between the "i" and "n". It is also related to the process known as globalisation in which 'the global corporation will seek sensibly to force suitably standardised products and practises on the entire globe (Levitt, in Russo and Boor, 1983). The processes of globalisation falls outside of the scope of this paper.

Contrasting Levitt's view, others promote the idea of diversity in product marketing. Software *localisation* refers to the modification of software to allow its use in a *locale* other than the software's origin. A locale is a geographical region defined by a country, language or set of cultural conventions (adapted from Hall, 1994). Locales are not only characterised by country alone because in some countries, three different languages are spoken in a particular country, e.g. French, German, and Italian are spoken in Switzerland (del Galdo, 1990); Malay, English, Mandarin, Tamil are used in Malaysia. Software localisation is,

therefore, not limited to "translating applications menus, dialogue boxes, alert boxes and content areas into a language or regional dialect" as defined in the Human Computer Interface Guidelines (Apple Computer, 1992). Localisation is also known as l10n; there being 10 letters between "l" and "n" in the word "localisation".
Both the software localisation and internationalisation processes should reflect consideration of the language issue and other user interface design issues such as the provision of icons, metaphors, navigational techniques and mental models, input and output mechanisms..
Culture is defined as behaviour typical of a group or class (Webster, 1995). Culture is conceptualised as a "system of meaning that underlies routine and behaviour in everyday working life" (Bødker, 1991). "Culture includes race and ethnicity as well as other variables and is manifested in customary behaviours, assumptions and values, patterns of thinking and communicative style" (Borgman, 1992). Thus, using the phrase "supporting a locale" means supporting that locale's language, cultural conventions and interface design preferences; an internationalised software package supports many locales simultaneously while localised software supports only one particular locale. An example of partially internationalised is Microsoft Windows in which the user can employ the Internationalisation section in the Control Panel to select a particular locale.

3. **Internationalisation and Localisation**

Most of the early software was developed in the United States (US) by Americans for Americans. There was little demand for software outside the US. As the number of computers increased around the world, demand for software also grew. Economic growth, especially in Asia, probably contributed most significantly to this demand. (See Section 5). With this growth of the overseas markets for software, software developers began to adapt their original software to other locales. The first attempts at localisation would have involved direct translation of the software into the language of the target community with modifications to format or display of numbers, date, time as well as character sets. This localisation process demanded time and effort. Furthermore, the early software developers did not bear in mind "problems of translation or effects of differences in culture and language on usability of software interfaces" del Galdo (1990).
The efforts required to localise software were magnified given that particular software needed to be localised to more than one language. Software companies found it was easier internationalise software - initially design the software such that the software provides the necessary support for a number of languages and cultural conventions (Chris Miller, 1994). Internationalisation thus brought about huge savings, increased revenues and profits, and shorter time to market, as little or no modification was required to localise an internationalised software (Chris Miller, 1994). The internationalisation process is, in a sense, a means to an end; the end being localisation.

4. **Approaches of Internationalisation and Localisation**

The current approach to preparing a product for world-wide use is to first internationalise and then localise the product (Russo and Boor, 1993) (Belge, 1995). There are a number of approaches to internationalisation.
Chris Miller (1994) reported these methods of internationalisation:

"1. Compile Time Internationalisation
Programmers change the files that contain the source code and algorithms
2. Link-Time Internationalisation
Programmers extract all the text strings along with algorithms that are dependent on language or culture, from the program code.
3. Run-time Internationalisation
End users select locale. The software contains text files for more than one locale."

Chris Miller (1994) also quotes another strategy by Bill Tuthill's four levels of internationalisation from the Solaris Internationalisation Guide: Global Product Design (Prentice-Hall, 1992):

> Level 1: software with texts and code sets that are adaptable internationally or are considered "8-bit clean" and that support the Latin-1 code set.
> Level 2: Software with formatting and collation methods that are locale sensitive. Includes sorting order for alphabets and formats for date, time and so on.
> Level 3: Software that allows easy translation of user-visible text, usually by placing such text into a separate file form the executable code,
> Level 4: Software that supports East Asian languages, which frequently require multi byte code sets.

Abramson (1994) believes that "the best design approach to internationalise software is to maintain strict language independence throughout the code".

Russo and Boor (1993) provided these guidelines:

"1. A product team needs to think about each culture the product will be sold in from the beginning of the product development cycle.
2. Are the product requirements different? Do the features need to be modified? How are the users different? The team needs to identify the differences, understand the design implications and design for them before the original product is released.
3. Establish A Relationship Within The Target Market
   A close working relationship or partnership with natives from the target cultures is needed.
   During the various stage of the product development cycle, information must be reviewed and feedback must be provided from the representatives of the target cultures.
   Early in the development cycle when features are defined, review for errors of omission and commission must be carried out by the target cultures.
   If additional features are identified, the original product development team can make appropriate adjustments at the product's architecture to allow for the change during the localisation process.
4. International Usability Testing
   Usability testing should be performed at the same time as domestic usability tests; before the product is released."

Belge (1995) provides the following steps to "more thoroughly serve international markets":

"1. Identify all areas of all products that change when going from one country to another. (A list of these areas are provided in Section 7).
2. Gain a better understanding of users.
   How users in the US and abroad think about, and deal with international issues, At this stage, one should investigate actual user work habits, and propose a conceptual, or user model that best describes how users think about their work.
3. Leverage technology to facilitate design suitability and consistency.
   Take advantage of the best and most appropriate technologies to test out your conceptual and design ideas.
4. Decide a common strategy that can accommodate the many ways in which international users work.
   Once ideas have been tested create guidelines and design specifications. When working on very large projects this will ensure that everyone is going in the same direction.
5. Unify the user interface design of the products by using this strategy".

The most direct approach localising approach to a product is translating the text in the interface, as well as that embedded to the language, to that of the target community. The translation models used to translate software in Aston and Dolden (1994) are similar to these techniques mentioned by Chris Miller (1994). For instance, the translation of "text external from the executable file (extracted either manually or with software tools" and "text external to the source code files but embedded in the executable file at compilation" is similar to Chris Miller's (1994) methods 1 and 2 above. If the product to be localised is an internationalised product, and if the internationalisation has been carried out effectively, localising the product will require little effort. The localisation process may only require the selection of desired locale.

5. Historical marketing of Software

5.1 Money Matters

It was only in the early 1990's that there has been indication of growth in research into localisation of software (Russo and Boor, 1993),(del Galdo, 1990),(Nielsen, 1990). One reason that may have catalysed this development has been the economic boom in the Asia as well as the opening up of markets in China and Vietnam, thus allowing a greater *economy of scale*. With greater economies of scale, more firms may find it more financially feasible to localise software. A large component of computer corporation's revenue have been derived from outside US , for example in IBM's 1994 Annual Report, 67% of IBM's revenue were derived from outside US. Furthermore, more and more countries are adapting Information Technology . The software market has become more culturally diverse with more countries becoming more technologically adept. The natives of many countries are able to develop software or localise software for their own consumption. Such as for example, the development of input and display of Chinese, Japanese and Korean characters.

5.2 Ethnocentrism

Another reason why software was not localised was because of the *ethnocentrism* of the software developers in the US. The Oxford Dictionary defines ethnocentrism as the egotistical opinion that their own race is the most important. This ethnocentric behaviour could be explained by this abstract from Gannon and Associates (1994):

> "Consequently, US economic success and military might often induces egotistical reactions to international events among its citizens. These narcissistic feelings are frequently evoked by news media's persistent portrayal of and fascination with the world disasters and mishaps. To the American media, "no news is good news," and thus the outside world is often reported as volatile, violent, and miserable. The positive features of foreign countries are rarely highlighted by US media, because the average American is constantly bombarded with news of famines, wars, violence, and political turmoil from the outside world. He or she is frequently not aware of the pleasant and fascinating features of other nations. This ethnocentrism is so extreme that many Americans believe that the United states is the safest and most prosperous country in the world, even when the facts do not support such a conclusion."

It is possible that such ethnocentric behaviour may have prevented the development of localised software. Other examples of this ethnocentric behaviour: rationalisation "they can understand English" was given for not translating a product's documentation and on-line help (Sprung, 1990), an American international marketing manager when asked whether translation was at all necessary was quoted "because isn't 1992 when everyone in Europe is going to speak English?" (Sprung, 1990).

6. Why localise software?

Increasing profits should not be the only goal for companies internationalising and localising their software. This section describes other reasons to globally marketing software.

From a social standpoint, localisation of software can serve as a medium to bring nations together by increasing the potential for communication between the countries involved. For example, software written in English and translated to Chinese may require the co-operation of both countries. Software, such as CAL, require large investments of time and

programming talent. With localisation of such software, there would be increased world-wide availability of state-of-the-art software as software companies increased the distribution of software, thus meeting the needs of a wider range of people. Many projects fail due to lack of acceptance by target cultures. With localisation, such projects would also stand a better chance of success. Furthermore, by localising software, communities less adept in software technology are exposed to state of the art software; thus, localisation of software can also be an effective mechanism for the transfer of technology.

People who can interact with computers in their own language learn and progress faster than those forced to use foreign software (Griffiths, 1994). For example, Bulgarian children only painted and drew but did not write when using software with an English language interface. However, in software that was localised, the children attempted to write creatively and explore all potential of the software (Griffiths, 1994).

Many people identify strongly with the culture they were brought up in. A particular community may feel "personally rejected and second rate if their language is not accepted into the magic circle of technology" (Griffiths, 1994). These communities may never develop the confidence to adopt IT. As such, significant contributions where IT is used would not materialise if these people cannot use the technology (Griffiths, 1994).

Furthermore, focusing on providing for "indigenous" cultures, preserving some of the diversity of human experience. By doing so "a wider variety of perspectives, potential insights and solutions to the world's problems" are preserved (Griffiths, 1994). This argument is also supported by a study carried out by the American Management Association (AMA) that "heterogeneous work groups create solutions to work and business problems that are more innovative and more effective than those developed by homogeneous groups" (HR Focus, 1993). It is clearly in the interest of humanity to maintain the diversity of human cultures.

7. Factors in internationalising and localising software.

Internationalising and localising software is not just a matter of translating the software. Many issues and factors are included in the partial list below. Software developers interested in developing software outside their locales need to be aware of these issues.

7.1 Date

Date formats differ in sequence from one locale to another for example: day-month-year, month-day-year, year-month-day. The dates also differ in the use of separators (full-stop/period, comma, hyphen, space, slash or no separator). Some dates use either numbers or alphanumeric characters (del Galdo, 1990)(Russo and Boor, 1993). Del Galdo suggests using alphanumeric characters for the months to avoid confusion especially if the application has an audience of more than one nationality. Dates can be written in a number of ways: May 12, 1959 (US), 5 June 1995, 5-Jun-95, 12/5/59 or 12.5.59 (UK), 12/5/59(Denmark), 1959-05-12 (Sweden) (Chris Miller, 1994).

7.2 Calendars

Different countries use different types of calendars such as Gregorian, Chinese, Chinese Lunar, Islamic, Hebrew and Japanese Imperial era. The US and UK and most the Western world use the Gregorian calendar. The Chinese use a Lunar calendar that is synchronised with the solar calendar by adding months, and the Muslims use another version of lunar calendar (Belge, 1995). Some countries use two calendars. Israel uses the Gregorian and Hebrew calendars while the Arab countries, the Gregorian and Islamic calendar (Chris Miller, 1994). Although the Gregorian calendar is used, the Japanese also count years using the Japanese imperial era based on the date the Japanese emperor ascended the throne (Chris Miller, 1994) (Belge, 1995). However, it is impolite to create a calendar that allows the user to reset the year to the beginning as this resetting questions the emperor's immortality. Even changing the date has hidden consequences for the unwary.

7.3 Weekends

Countries differ in the days the "weekend" or "Holy Day" is on. Most of the West have Saturday or Sunday as "weekends" because Sunday is the Christian's day of rest or Sabbath. In some Islamic countries, the "weekend" is Thursday and Friday, not Saturday and Sunday (Belge, 1995) as Friday is Prayer day. Some states in Malaysia have weekends on Saturday and Sunday whereas some other states, Thursday and Friday. In some Middle East countries, the "weekend" days are also not necessarily contiguous (Belge, 1995).

7.4 Day Turnovers

In most countries, date changes at midnight, e.g. US and UK. However, in Islamic countries, date changes at sunset (Belge, 1995).

7.5 Time

Designers should be aware of the 12 or 24 hour notation and valid separators used e.g. colon, period, space and no separators. Time are denoted as 8:32 pm. (US), 20:32 (Canada), 20,32,00 (Switzerland), 20.32 Uhr (Germany), KI 20.32 (Norway)(Chris Miller, 1994). Del Galdo (1990) suggests that time should cater for 12 and 24 hour clock notation and for international time, use of time zones to indicate the variation from the Greenwich Mean Time for example +12 hours (Hamilton, New Zealand). It would be useful to include the country in which the city is located to avoid confusion. For example, there are a number of Hamiltons in the world. Besides the one in New Zealand, there is one in Canada and another in Bermuda (Crystal, 1994) among others. Another feature that should be incorporated is the allowance for daylight savings, especially in the temperate countries which experience longer hours of daylight in summer.

7.6 Telephone Numbers

Different countries have telephone numbers of different lengths, formats and separators (space, hyphen, period, comma, square brackets, parenthesis, and plus sign). The telephone numbers may differ depending on whether the numbers are for national or international use(del Galdo, 1990). Del Galdo suggests a design for arbitrary formats for national numbers. For example, in New Zealand, local telephone numbers are written as 856 2889 or 856-2889 and (7) 856-2889 for international numbers, (7) being the area code for Hamilton. In the United States, the numbers are written as +1-212-626-0500, 1 is the country code and 212 the area code. In the UK, (01222) 584396 or 1222 584396, where (01222) is the area code for Cardiff region in UK. Abbreviations of telephone may also differ e.g. "Ph:" (NZ), "Phone:" (US), "Tel:" (Malaysia). The abbreviation for Facsimile, "Fax", seems consistent for UK, US and Europe.

7.7 Personal and Business Addresses

Address formats differ from country to country. House numbers are generally placed before street names. For example, in America but not in Latin America and Europe, the street name is written first (Chris Miller, 1994). The postcode may also appear either before or after the city, state or province, or suburb. Americans would write their house number before the street address, postcode after the state:

John Doe
456 Del Mar Avenue
Santa Barbara California **93140**
USA.

An example of an address in Europe, with the house number after the street and the postcode before the city:

European Service Center
Avenue Marcel Thiry **204**
1200 Brussels, Belgium.

7.8 Character Sets

"Every language has its own alphabet or script" (Chris Miller, 1994). English uses the 26 Roman characters while languages like Chinese requires practical modern dictionaries covering about 7000 characters. European languages contains some characters with diacritics

such as ë, â and á. Software in any language would require the support of the correct alphabet and other characters such as currency symbols.
Thus, 7-bit ASCII code set which supports 128 characters was invented to support American English, the language of the earliest programs. The 128 characters that ASCII supported include control characters, accents, control codes and the upper and lowercase of the Roman alphabet (Chris Miller, 1994). To include characters for European languages, diacritics and umlauts, the code set was extended to use 8-bit extended ASCII, which supports 256 characters. The Windows character set is based on ISO standard 8859-1 also known as ECMA-94 and Latin 1 (Hall, 1991). Latin 1 includes character sets that support the main languages of Western and Eastern Europe, such as Cyrillic, Greek and Turkish.
Languages with more than 256 characters such as Chinese, Japanese and Korean use double-byte, multi-byte or wide character sets. "A double-byte character uses 16 bits. A multi-byte can mix single and multi-byte characters. A wide character set typically contains 16- and 32-bit characters" (Chris Miller, 1994). Given the multitude of single- and multi byte-characters, a consortium developed a 16-bit character code standard known as Unicode. Unicode, also known as ISO 10646, is being promoted to replace ASCII and the other multi-byte character sets.

Unicode is a fixed-width, 16-bit character set which can support up to 65,536 characters. This support allows the inclusion the written text of almost all known modern languages as well as some ancient languages (Abramson, 1994). Windows NT has embraced Unicode while Apple intends to do so in future releases of its system software (Chris Miller, 1994). Any software developer intending to internationalise software products should ensure that support is provided for the character sets of their target community's languages. .Problems that may arise from use of 2-byte representation include font data storage and handling (Peterson, 1990) as well as a requirement for significant amounts of additional memory to run the eventual applications (Chris Miller, 1994).

7.8.1 Input and Output

Although input of Roman alphabets may be straightforward on the Qwerty keyboard, input problems arise in other languages. For example, there are more than 50,000 Hanzi. There exists more than 100 methods to enter these characters (Huang, 1985). Some methods to input Chinese, Japanese and Korean characters include the use of coding schemes, arrangement of symbols and phonetic schemes which are covered in (Huang, 1985), (Becker, 1985),(Walters, 1990) and (Hsu, 1991). Software developers entering the Chinese market would need to be aware that the input methods of the Windows 3.1 version for Taiwan include Chang Jei, Phonetic, Quick/simplified, Internal code, DA-YI, and Array (Chris Miller, 1994). Output of languages also differ with certain languages such as Arabic and Hanzi being more graphical than other languages such as English. A minimum resolution for screen display is required to ensure readability of the script (Sukaviriya and Moran, 1990). For example, in Walters (1990): "According to Huang and Huang, dot matrix techniques using a 16 x 16 matrix cannot produce (Chinese) characters of acceptable quality but storage of the more acceptance 24 x 24 matrix requires additional memory".

7.9 Collating/Sorting Order sequence

As different countries have different alphabets, so sorting or collating sequence also differ (Zobel-Pocock, 1990)(del Galdo, 1990)(Belge, 1995). Sorting sequences affect address lists and telephone directories. In Japan alone sorting sequences can be according to Ascending Lead Byte, Code Order, Pitch Casing, Pronunciation, and Stroke (Belge, 1995).
In certain cases, different rules govern the way the same sets of characters are sorted (del Galdo, 1990). Del Galdo (1990) describes sorting of letter with umlauts (diacritics) e.g. ä,ö,ü: in the German collating sequence, letters with umlauts are sorted as the same letters without umlauts; the Swede sorts the "ü" with "y" and places the other letters with diacritics at the end of the sequence. Walters (1990) reports that sorting sequence in the same language may be different, e.g. the German telephone directory is sorted slightly differently from a German dictionary.

7.10 Flow

Reading and writing directions differ from language to language. Chinese characters are written top to bottom with each new line of characters appearing to the left of the old line i.e. top to bottom, right to left. Arabic is written from right to left with each new line of script appearing below the previous line i.e. right to left, top to bottom. However, the numbers in Arabic are still written from the left to right (Tayli, 1990). Other languages follow the English language, left to right, top to bottom flow. Software developers must bear in mind the effects of language direction on user interface design, particularly with menus, dialogue boxes and error message windows.

7.11 Punctuation Marks

In Greek, interrogation is expressed not by the American English question mark but by what looks like a semicolon "; " (Chris Miller, 1994). Other punctuation marks include an inverted exclamation and question mark (Hall, 1992). In Hanzi, Greek and French "« »" are used as quotation marks (Chris Miller, 1994).

7.12 Translation

The translation of a user interface is not as easy as the translation of a book (Nielsen, 1990). There are many pitfalls to avoid when translating the user interface as "an interface introduces many additional subtleties such as computer-human interaction which can complicate a translation" (Russo and Boor, 1993). Translation problems are exacerbated when "the translator is unfamiliar with the application content or human factors principles" (Russo and Boor, 1993). An Indonesian exchange student, hired by a US company, translated "software" as "underwear". Sun Microsystems translated the word "menu" as "list of food items" in Cantonese (Russo and Boor, 1993). Examples of translation problem "page" in MS Word is translated to "side" Danish. In MacPaint, "Goodies" (English) to "godter" (Danish), a proper translation, but "godter" refers more to candies than to a variety of functionalities. Del Galdo (1990) also warns about the use of acronyms or abbreviations since the acronyms or abbreviations may have negative associations in other cultures. Software developers need to be aware of the following translation issues.

7.12.1 Words that don't exist

It is very common to find that computer-related words or terms do not exist in some languages. After all, the present computer vocabulary was invented when the need arose or when the item or concept was created. Terms like "Diskette", "Disk drive", "Zooming" and "Panning" do not have a direct translation in the Thai language (Sukaviriya and Moran, 1990) and probably in many other languages as well. Thus, new words are created and new phrases are invented to convey the meaning. However, even if new words are invented, the new word may not encapsulate the meaning intended. For example, in Microsoft Word, a direct translation of "ruler" in Thai refers to an object rather than a tool in Word (Sukaviriya and Moran, 1990). New phrases created also tend to be lengthy and sometimes users return to the original word. Nielsen (1990) found the use of "undo" command in English while all other commands were in Kanji because it was really impossible to translate "undo". Lengthy translations would also increase the space required for the commands in menus as well as dialogue boxes thus "cluttering" up the user interface.

Misinterpretation of translations also exists. For example the translation of "eject" in English to "aflever" which means "hand over" in Danish; users misinterpreted "aflever" as hand over the file from the computer to diskette (Nielsen, 1990). A better translation was found, "skub ud" which means "push out" (Nielsen, 1990). Another example is "Quit" in English translated to "slut" in Danish where "slut" means "the end"; naive novice users were scared of the word but not the more confident new users or the experienced users (Nielsen, 1990). The word "slut" was interpreted as too much of a termination, such as shut down of the computer (Nielsen, 1990).

In countries that need to create new words, it is important that there be an organisation that invents these new words. This organisation should publicise the words created to ensure there is a consistency in the terminology used.

7.12.2 Menu accelerators

According to Nielsen (1990), the problem users hate most is the inconsistency in control/function-keys/menu-accelerators. Nielsen illustrated the control-keys problem with this example: As the Danes used both English and where possible Danish software,

"MacDraw 1 retained control keys such as R for right-justify and L for left-justify (instead of H for "højre" and V for "venstre" thereby destroying their mnemonic value)". "Save" and "Exit" in French are "Sauver" and "Sortie" respectively. There is a confusing overlap if "*S" was also used for "Exit" (del Galdo, 1990).

7.12.3 **Names**

The names for everyday objects vary from country to country. Some common examples includes the trash can (US) vs. a dustbin (UK) or a rubbish bin (NZ); truck (US) vs. lorry (UK), a period (US) vs. a full-stop (UK). Some product names do not translate well. The Italian automobile manufacturer, Fiat, had to change the name of their car "Uno" because "Uno" means garbage in Finnish (Russo and Boor, 1993). Similarly, Rolls Royce had to alter the "Silver Mist" in Germany because mist means manure (Russo and Boor, 1993). The brand name of a New Zealand line of clothes, Tahi (one in Maori) needs to be changed if the manufacturers plan to export this line of clothes to Malaysia; "Tahi" in Malay means manure. Terms preferred by male software engineers in past such as kill, trash, abort should be avoided as some women find these terms "brutal" (Marcus, 1993).

7.12.4 **Translation of Text**

When translating text from English to another language, Chris Miller (1994) suggests leaving enough space the for text expansion. Chris Miller (1990) reports that Microsoft Software Development Kit recommends an allowance of 200 percent extra space from 1 to 10 English characters, 100 percent extra space for 11 to 20 English characters and 30 percent for 71 or more English characters. Hall (1991) describes Machine Readable Information (MRI) as "IBM's term for the language-sensitive information that passes between the user and the program". MRI can comprise commands, responses, messages, menus, dialogues, help animation audio, tutorials, clip art, and icon (Hall, 1990). Thus when translating MRI from one language to another, enough space must be provided for MRI expansion (Hall, 1990). The table below gives recommended minimum MRI Expansion space (Hall, 1990).

English text length	**Expansion space recommended**
0 to 10 characters	100 percent to 200 percent
11 to 20	100 percent to 200 percent
21 to 30	80 percent to 100 percent
31 to 40	60 percent to 80 percent
41 to 50	40 percent to 60 percent
51 to 70	31 percent to 40 percent
Over 70	30 percent

An extreme example of MRI expansion : "message pop-up" in German is; "Nachrichtenuberlagerrungsfenster" and in Portuguese "janela de sobrepopsiçao de mensagem, Danish pop-op-meddelelse" (Hall, 1991). For example, "Bildschirmeinstellungen" is the German translation of the Preferences menu item from the Windows menu (Chris Miller, 1991) Certain situations may arise that with the space allowed for dialogue boxes may become too "crowded". To solve this situation, boxes that contain text could be self sizing or movable (Chris Miller, 1994). Chris Miller (1994) warns "terminology consistency is crucial, since there can be hundreds of cross-references between the interface, the documentation, the text files and the filenames". One must be thus aware of the effects of translated items on its linked or related components.

7.12.5 **Documentation Translation**

Sprung (1990) reports that a marketing manager will often decide not to translate his product's user interface (documentation, on-line help) with the rationalisation: "they can

understand English". Companies do not to translate documentation because the translation of such documentation is effort-intensive and costly. However, documentation can serve as a marketing and public-relations tool, a means of showing that the company stands behind its product (Sprung, 1990).

7.13 Paper Size

Different countries also have different paper standards. Eg. American paper standards are legal, letter or ledger, elsewhere in the world are A3, A4 and A5 (Chris Miller, 1994). Japanese paper size are JIS-B4 and JIS-B5(Chris Miller, 1994). All these paper sizes need to be provided for WYSIWYG software (Belge, 1995).

7.14 Units of Measure

Although many countries have converted to the Metric System, some countries like US still use the English imperial units e.g. pounds, inches, gallons, miles, Fahrenheit, instead of kilograms, centimetres, litres, kilometres, Celsius or Centigrade. These units need to be reflected in software such word processors where units such as inches, centimetres, picas are used or in scientific software requiring correct quantitative measurements.

7.15 Numbers

Separators used in numbers (Arabic numerals) vary from culture to culture. In Europe, a comma "," is used as a decimal point and a period or full-stop "." is used to separate thousands. This conversion is the opposite of the UK and US convention where a full stop "." is used as a decimal indicator and a comma separates thousands. In Europe, the number 1,234,567.89 in US is represented as 1.234.567,89 (del Galdo, 1990). Valid separators include comma, period, space and apostrophe. Thus 3,912.45 can be 3.912,45 or 3 912,45. (Chris Miller, 1994). Positive and negative numbers can be indicate by either using the plus "+" and minus "-" prefix or suffix. Negative numbers can be denoted by enclosing the number in parenthesis (del Galdo, 1990). A billion in the US refers to one thousand million i.e. a one followed by nine zeros whereas in Latin America and Europe a billion is one million million, a one followed by 12 zeros (Chris Miller, 1994).

7.16 Currency

Different countries would have different currency symbol; British pound (£) or Japanese yen (¥), and the location of the symbol would also differ (Belge, 1995). In some countries, currency symbol may appear at the end of the number, for example, 4.567,89 DM in Germany or in Portugal, where the currency symbol is used as the decimal "symbol" 4.567$89Esc (Belge, 1995)(Esc is the abbreviation of the Portuguese escudo (World Fact Book, 1995)).

7.17 Visuals

In this section, images, icons and symbols are classified as visuals. Visuals used in the user interface impart certain information to the users. However, like other attributes described, visuals can be perceived differently by different cultures. An example of a visual that may not be recognisable is the trash can icon that appears in the Macintosh operating system. To British users, the trash can looks like a postal box (Russo and Boor, 1993).

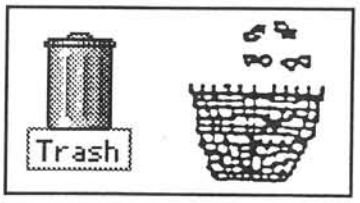

Figure 1 Icons for containers for refuse in the United States and Thailand (Sukaviriya and Moran, 1990).

The American trash can may not even be recognised by the Thais as in Thailand the "trash can" is actually a wicker basket (Sukaviriya and Moran, 1990).

Figure 2 Icons for container for mail in a number of nations (Apple Computer 1992).

The appearance and shape of mailboxes vary from country to country as shown in Figure 2 above (Apple Computer, 1992). Ironically, some urban American dwellers also do not recognise rural mail box icon (Russo and Boor, 1993). In the US and UK, red and blue ribbons are used awarded in recognition for performance or achievements in competitions, but ribbons as symbols of recognition for achievements might not hold true in other countries (del Galdo, 1990). Some visuals are recognisable in another culture but they convey a totally different meaning. In the United States, the owl is a symbol of knowledge but in Central America, the owl is a symbol of witchcraft and black magic (Apple Computer, 1992). A black cat is considered bad luck in US but good luck in the UK (del Galdo, 1990). The use of a cocktail glass icon in a calendar application to represent after work appointments may be appropriate in societies that socialise after work at the local bar or tavern, but the icon is inappropriate in countries where alcohol is not associated with social activities (del Galdo, 1990) such as Islamic countries. An image of the sun shining may conjure up feelings of pleasure and satisfaction to British and New Zealand residents, but to residents of arid countries, the same image may be associated with the threat of pain or death (Griffiths, 1994).

Figure 3 An ambiguous hand gesture (Pease, 1981).

Images of hand gestures such as the ring or OK gesture (Figure 3) may be understandable to most English speaking countries, but in France the same gesture means "zero", "nothing" or "worthless". In some Mediterranean countries the gesture implies a man is a homosexual (Pease, 1981). In the Christian world, symbols like the archetypal cross is associated with prohibition but not in Egypt (Russo and Boor, 1993). Crosses or checks that are commonly perceived as signs or disapproval and acceptance may not have universal meaning (Russo and Boor, 1993). The number 13 is considered unlucky in America as is number 4 in Japan, 7 in Ghana, Kenya and Singapore (Russo and Boor, 1993). To most Cantonese (a Chinese dialect), 8 is considered lucky as 8 is pronounced as "fatt" which means prosperity and good fortune.

Certain visuals may be comprehensible but unacceptable. Care needs to be taken when using religious symbols such as crosses and stars: that's the reason why we have a Red Cross in Western countries but a Red Crescent is adopted in the Islamic countries.

Figure 5 Universally recognised icons for Olympic Games events .

However, there are globally recognisable icons such as those depicting the various Olympic sports shown in Figure 5.

7.18 Colours

Colours hold different meanings for different cultures. For example, red could be associated with happiness or prosperity in China, anger or danger in Japan, life or creativity in India, death in China, aristocracy in France, and danger or stop in the United States (Russo and Boor, 1993). Russo and Moran (1990) also reports the association of green with go. A red "X" was found to be ineffective as a symbol for forbiddance (sic) in Egypt (Russo and Boor, 1993). In the UK, a red and blue ribbon signifies best or first and second in a given class respectively. In the US, the opposite is true, blue for first and red for second (del Galdo, 1990). Workers at a nuclear plant may have different colour associations compared to forest rangers because of the work settings.

7.19 Functionality

Developers must realise that the functionality embodied in software that works for one country may not work for another. Groupware is a useful and effective tool. However, groupware loses some of its value in use in Japan. According to Nakakoji (1994), Japanese businessmen often conduct brainstorming sessions in a social settings after work and they usually conduct negotiations before the meeting. It is "socially unacceptable to challenge your manager's idea in public". During the actual meeting time, employees just agree with any proposal put forward by their manager. Using groupware to record decisions during a meeting would be pointless.

Nielsen (1990) described another example of different cultures' preference of functionality. LYRE, a French hypertext system for teaching poetry, allows the students to see the poem from various "viewpoints". LYRE allows the teacher, but not the students, to add new viewpoints. This design is acceptable to Southern European tradition because if students were allowed to make changes, the teacher's authority would be undermined. However, people in Denmark where Scandinavian attitudes are prevalent would view the current design of LYRE as unsociably unacceptable as "it limits the students' potential for independent discovery" (Nielsen, 1990). Lotus research revealed that methods of amortisation in accounting varied between Germany and America. Thus, the financial functions in Lotus 1-2-3 had to incorporate these differences (Sprung, 1990).

7.20 Sound

Sound affects different nationalities differently. Most error messages are accompanied by a "beep" sound to draw the users' attention to the problem. In Japan, it is impolite to "Beep" as this calls attention to a possible error on the part of the user. Other users may hear the beep and become aware of the person's error, causing embarrassment to all involved (Belge, 1995).

Also, a pig may "oink" but to Hungarians a pig goes "gruff". Thus, if children were required to pair off animals with the sound the animals made, in a software originating from the UK, the children in Hungary may have problems trying to identify the correct "sound".

7.21 Metaphors

A typewriter metaphor would be quite useless for a word processor if you don't know about typewriters. " The typical Japanese has had little exposure to the typewriter because, before computers, documents were formatted using writing pads that typically used a 20-by-20 grid for each character. To judge the length of a document in Japanese, we count characters

instead of words. Japanese users have had considerable trouble understanding the concept so cursor movement embodied in a typical work processor application. Nakakoiji (1994) states in Usage Styles that it is "Advisable to get to know the culture if one is to get a feel ' for what metaphors will be acceptable.

8. **Critique and Further Work**

Internationalisation and localisation have gained more attention today when compared with the late 80s. This trend is evident given more papers being published since 1990. The book *Designing User Interfaces for International* edited by Jakob Nielsen, without a doubt, contains the earliest relevant discussion of internationalisation and localisation. More papers have been published since then, the creation and promotion of Unicode, and the setting up of mailing lists like INSOFT-L and SIGCHI's (Special Interest Group in Computer Human Interaction's) intercultural.CHI and INSOFT-L to encourage active discussions on these topics. Also, more books on the internationalisation and localisation have also appeared. Examples of such books include *Programming for the Whole World: A Guide to Internationalisation* by Sandra O'Donnel (1994), *Global Software: Developing Applications for the International Market* by David Taylor (1993) and *Software Internationalisation and Localisation: An Introduction* by Emmanuel Uren, Robert Howard and Tiziana Preinotti (1993). Furthermore, large software companies including Apple, Hewlett Packard, Sun Microsystems and Unix International also have books internationalising and localising their companies' systems.
Despite the recommendations of the more recent papers like Russo and Boor (1993),Chris Miller (1994), Belge (1995) most of the software existing now still emphasise character sets, time, date, collation formats. For example, Windows NT is "internationalised" and has a choice of language, keyboard, date, time, currency and number formats. Windows NT also supports Unicode. These factors such as character sets, time date, collation format can be termed as the "basics" or covert attributes which are straight forward and intuitive. On the other hand, with the overt attributes such as visuals, colours, sound; problems with these attributes can only be isolated through usability tests with target users. Covering the "basics" is not enough. Russo and Boor (1993) quoted Taylor: "properly localised software applications, just like properly localised automobiles, toasters, beverages and magazines, reflect the values, ethics, morals and language (or languages) of the nation in question". Thus, there is a need to look into the covert attributes to find out how these factors affect the user interface and the software as a whole, and how these factors can be integrated into the interface. For example, if the software supports ten locales, there should be at most ten interfaces with each locale's own cultural conventions.
There is also a need to develop techniques for incorporating the list of factors into the internationalised software development process (Russo and Boor, 1993). Another critical question is how to reduce the effort involved in localising software. A possible solution would be to separate the user interface from the functionality of the application software. With a separate user interface, changes to the user interface is less likely to affect the application code and vice versa. Thus, every locale can have its own user interface. User interface software tools can also speed up the development of the different user interfaces by converting overt and covert attributes to meet the requirements of a different locales.

8.1 **Globalising Software Products**

It can be argued that the whole endeavour of i18n is misinformed. A better solution for non-western nations seeking to exploit the potential of Information Technology (IT) would be to develop software for their own needs within their own the cultural context. It is clear for each culture to recapitulate the IT experiences of western nations represents a huge investment in reinvention of an industry with a wealth of experience in building software. Furthermore, the underlying issue of cross-cultural communication within geographical areas would not be addressed by the separate development of multiple software products. Somewhere between IT Apartheid (each country develops only indigenous software) and Globalisation (only software localisable to any culture is produced) is a balance that takes

into account the desire of people to both take advantage of IT and have those advantages provided in a culturally appropriate way.

9. **Conclusion**

Two processes are involved in developing internationalised software, internationalisation and localisation. There are a number of factors that need to be addressed before any software is internationalised or localised. These factors include identifying overt factors such as date, time collation, number and currency formats specific to the target cultures. Covert factors such as visuals, colours and metaphors also need taken into consideration. Even though there is an increase in publicity of internationalisation and localisation processes, research on these topics is still sparse. The technique of separating the functionality of software from the application interface has been demonstrated (Barbour, 1996) and is advocated because it appears to speed up the development of some localised software.

Successful i18n processes may not resolve issues of cultural independence or enhance the preservation of cultural variety that enriches human experience. I18n suits multi-national companies and ensures the hegemony of western nations over information technology. Cultures with little global influence will have to work very hard to develop information technology in the face of the current lead of western technologies. A first step in that process is the development of a consensus, in the global community, that software for multicultural (i18n) contexts reflects in design the separation of functionality from interface. This step would facilitate the growth of software industries in all cultures which wish to particpate in data exchange. The continued involvement of culturally specific providers would ensure that the issues of identified in i18n documents at United Nations level are accommodated for in localised software. Clearly, further and ongoing research is required to monitor and evaluate new algorithmic techniques and interface designs for specific cultures. Urgent work is also required in monitoring and evaluating the progress the IT industry makes towards enhancing international and intercultural communication supported by software.

References

1. Abramson D. 1994: Globalization of Windows. *BYTE.* Vol 19. No 6. p177-184.
2. Apple Computer. 1992: *Human Computer Interface Guidelines.* Addison-Wesley.
3. Aston M, Dolden B. 1990: Logiciel Sans Frontiers. *Computers in Education.* Vol 22. No 1/2.
4 Barbour R.H. 1996: Te Ripanga: the spreadsheet. *Proceedings of the 6th Biennial New Zealand Computers in Education Society.* Hamilton pp66-76.
5. Becker J D. 1985: Typing Chinese, Japanese and Korean. *IEEE.* January 1985. p27-34.
6. Belge M. 1995: The Next Step in Software Internationalisation. *interactions.* ACM Publishers. Vol 2. No 1. p21-25.
7. Bødker K, Pedersen J S. Workplace Cultures. 1991: *Looking at Artifacts, Symbols and Practices.* In Greenbaum J, Kung M (ed.). Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum Associates. p121-136.
8. Borgman C L. 1992: Cultural Diversity in Interface Design. *SIGCHI Bulletin.* Vol. 24. No. 4. p31.
9. Chris Miller L. 1994: Transborders Tips and Traps. *BYTE.* Vol 19. No 6. p93-102.
10. Farid M. 1990: Software Engineering: Globalization and Localization. *IEEE (CDROM version).* p491-495. **
11. del Gardo, E. 1990: *Internationalization and Translation: Some Guidelines for the Design on Human Computer Interfaces.* In Jakob Nielsen (Ed.) Designing User Interfaces for International Use. Elsevier, New York. p1-10.
12. Gannon M J and Associates. 1994: *Understanding Global Cultures: Metaphorical Journeys through 17 countries.* Sage Publications.
13. Griffiths D , Heppell S, Millwood R, Mladenova G. 1994: Translating Software: What It Means and What It Costs for Small Cultures and Large Cultures. *Computers in Education.* Vol. 22. No. 1/2. P9-17.
14. Hall W S. 1991: Adapt Your program for Worldwide Use With Windows Internationalization Support. *Microsoft Systems Journal.* Nov-Dec 1991.p29-58.
15. Hall W S. 1994: Internationalisatioon in Windows NT, Part 1: Programming with Unicode. Microsoft Systems Journal" June, p57-71.
16 Hoecklin, L. 1995: *Managing Cultural Difference.. Strategies for Competitive Advantage.* Addison-Wesley Publishing Company, Wokingham, England.
17. HR Focus. 1993: More Companies Are Drawing Strength from Diversity. *HR Focus.* Vol. 70. No 6. p2.
18. Hsu S C. 1991: *A Flexible Chinese Character Input Scheme.* UIST '91 (Hilton Head, South Carolina 11-13 November). p195-200.
19. Huang J K. 1985: The Input and Output of Chinese and Japanese Characters. *IEEE.* January 1985. P18-24.
20. Kay R. 1994: Software Goes Global. *BYTE.* Vol. 19. No. 6. p90-91.
21. Marcus A. Human Communication Issues in Advanced User Interfaces. *Communications of the ACM.* April 1993. Vol. 4. No. 4. p101-109.
22. Microsoft Annual Report, 1994.
23. Nakakoji K. 1994; Crossing the Cultural Boundary. *BYTE.* Vol. 19. No. 6. p107-109.
24. Nielsen J. 1990:*Usability Testing of International Interfaces.* In Jakob Nielsen (Ed.) Designing User interfaces for International Use. Elsevier, New York. p39-44.

25. Nielsen J (Ed.). 1990: Designing User Interfaces for International Use. Elsevier, New York.

26. _ OECD. 1989: The Internationalisation of Software and Computer Services. OECD.

27. Pease A. 1981:*Body Language: How to Read Others' Thoughts by Their Gestures.* Camel Publishing Company, Avalon Beach, Australia.

28. Pugh J, LaLonde W. 1992: Internationalizing your application. *Journal of Object oriented Programming.* Vol. 4 No. 8. January 1992. p59-62.*

29. Russo P, Boor S. 1993: *How Fluent is your Interface? Designing for International Users.* INTERCHI '93 Conference on Human Factors in Computing Systems: INTERACT '93 and CHI'93. (Amsterdam, 24-29 April). ACM Press. p342-347.

30. Sprung R C. 1990: *Two faces of America: Polyglot and Tongue-tied.* In Jakob Nielsen (Ed.) Designing User interfaces for International Use. Elsevier, New York. p71-102.

31. Sukaviriya P, Moran L. 1990: *User Interface for Asia.* In Nielsen J (Ed.). Designing User Interfaces for International Use. Elsevier, New York. p189-218.

32. Tayli M, Al-Salamah A I. 1990: Building Bilingual Microcomputer Systems. *Communications of the ACM.* Vol. 33. No. 5. May 1990.

33. Walters R F. 1990: Design of a Bitmapped Multilingual Workstation. *IEEE.* February 1990.

34. Weitz, 1995: Posting in newsgroup comp.std.internat.

35. Webster HyperText Interface. 1995: http://c.gp.cs.cmu.edu:5103/prog/webster

36. World Fact Book. 1995: http://www.odci.gov/cia/publications/95fact/.

37. Zobel-Pocock R A.1990:*International User Interfaces.* In Jakob Nielsen (Ed.) Designing User interfaces for International Use. Elsevier, New York. p219-227.