Decision Support Systems for Sustainable Development

A Resource Book of Methods and Applications

EDITED BY
GREGORY E. KERSTEN
ZBIGNIEW MIKOLAJUK
ANTHONY GAR-ON YEH

International Development Research Centre (IDRC)
and
Kluwer Academic Publishers

11357

DECISION SUPPORT SYSTEMS FOR SUSTAINABLE DEVELOPMENT

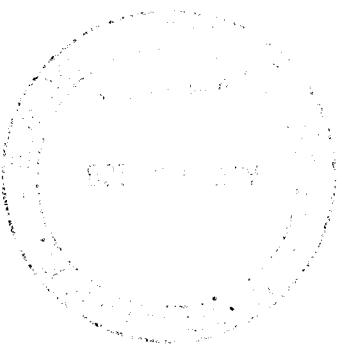
A Resource Book of Methods and Applications

DECISION SUPPORT SYSTEMS FOR SUSTAINABLE DEVELOPMENT

A Resource Book of Methods and Applications

Gregory (Grzegorz) E. Kersten
Zbigniew Mikolajuk
Anthony Gar-On Yeh

Editors



Kluwer Academic Publishers
Boston/Dordrecht/London

ARCHIV
MIKOLA
No. 113698

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available from the Library of Congress.

Canadian Cataloguing in Publication Data

Main entry under title :
Decision support for sustainable development : a resource book of methods and applications

Includes bibliographical references.
ISBN 0-88936-906-2

1. Sustainable development — Developing countries.
2. Decision making — Developing countries.
3. Information, storage and retrieval systems — Sustainable development.
I. Kersten, Gregory E.
II. Mikolajuk, Zbigniew.
III. Yeh, Anthony G.O., 1952- .
IV. International Development Research Centre (Canada).

HC79.E5D43 1999 338.9'009172'4 C99-980239-9

© International Development Research Centre 2000
PO Box 8500, Ottawa, Ontario, Canada K1G 3H9
http://www.idrc.ca/books/

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, mechanical, photocopying, recording, or otherwise, without prior permission from Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, MA 02061, USA.

Published in hardcover by
Kluwer Academic Publishers
101 Philip Drive, Assimippi Park, Norwell, MA 02061, USA
http://www.wkap.nl

Printed on acid-free paper in the United States of America.

For
Margaret Kersten,
Anna Mikolajuk and
Brenda Yeh
with thanks for
their inspiration
and encouragement.
Thank you!

Contents

15 SOFTWARE INTERNATIONALIZATION ARCHITECTURES FOR DECISION SUPPORT SYSTEMS

Patrick A.V. Hall

1. Introduction

As can be seen in the many other papers in this book, decision making using DSS occurs in all countries and is undertaken by people who are not necessarily fluent in any language other than their mother tongue. Thus the decision support software must be made available in this mother tongue. But the situation is more complex than that, for some decisions may need to be taken using information gathered from across a complete region from data sources in different languages and cultures, and the decision making process may also involve many people from similarly diverse languages and cultures. The processes of collaborative decision making are covered in Chapter 12 on business negotiations, while in this chapter we will focus on how to make software available for a range of languages and cultures.

Globalization of commodity software products like word processors, spreadsheets, and their underlying operating systems, is becoming widespread. The suppliers of these systems, like Microsoft, Lotus, Claris, Apple, the main platform suppliers, and many others, have recognized since the start of the 1990s that more than half their revenues must come from outside the US and the English speaking markets. The US market is saturating.

This has meant that the manufacturers of software have developed methods for translating their products from their original target market, typically the US, to new markets. There are a wide range of issues to be addressed – these are described in Section 2.

The methods of attack have been based on relatively simple methods, on characterizing the essential features of a market within a "locale", using a suitable character coding standard, and using resource files to factor out the locale dependent data like

The methods of attack have been based on relatively simple methods, on characterizing the essential features of a market within a "locale", using a suitable character coding standard, and using resource files to factor out the locale dependent data like messages so that they can be easily replaced. These current approaches are described in Section 3.

Nevertheless, translation of software remains expensive, and not all markets are large enough to warrant translation. Thus, for example, the Same language of Lapland in Northern Finland, Sweden, and Norway, will not have the interfaces of office products translated, though the ability to store and manipulate data in the Same language will be enabled. Same is closely related to Finnish.

New approaches are needed, and I have been involved with some of these within the EU-funded Glossasoft project. These new approaches aim to exploit software architectures to factor out the locale dependent aspects behind Application Programmer Interfaces, and linguistics to enable this and avoid internationalization imposing unnatural interfaces upon the software. These approaches are described in Section 4.

Finally, in Section 5, we look forward to the kind of work that will be necessary to enable Decision Support Systems to succeed across countries and regions of the developing and developed world. All current approaches are aimed at single languages, though the same principles would work for a single system accessed through multiple languages, such as a decision support system spanning several countries or locales. There are two levels of linguistic issue: diverse interfaces in multiple languages accessing a single common repository of data which is in some manner language neutral; and systems where the shared data is in multiple languages intermingled. But behind all this there are deeper cultural issues about how decisions are made: social relations, and the perceptions of space and time, are critical. This area is only partially understood, and the issues and some indication of how to handle these technically will be discussed.

2. The localization problem

In moving any piece of software from one part of the world to another, it is important that the different needs of the users at these different locations are taken into account. These differences are:

1. the language of the user interface,
2. the language and number representations and measurement units of the data stored within the system,
3. the language used for product support
4. "cultural" factors like representation of currency, dates and colours
5. local practices such as legal requirements
6. deeper cultural issues like those characterised by Hofstede (1991).

These factors are also equally important when a single system is used through multiple languages, as when sharing information between different countries. Even

when the language is the same, some of these factors will be important because the user community is different.

2.1 Language

The user interface of software should feel "natural" to the user of that software. All aspects of the computer and its communication with the user should be in the language of that person, or some second language in which the person is very comfortable.

For example, in European countries, nationalities differ markedly in their ability to understand one or more foreign languages. In some countries as few as 20% of the population are able to understand a language other than their own, while in others like Luxembourg the proportion can be as high as 90%. In South Asia computer systems are delivered almost exclusively in English as a consequence of their colonial heritage, but English is only spoken by the 5% of the population who are the elite. However, the real ability to understand subtleties and shades of meaning of the second language, and the meaning of specific terminology used in software applications, may be substantially lower. English or some other dominant "world" language such as Arabic, Russian, Spanish, Chinese, or French just cannot be used.

The aspects of languages that are important are:

Writing systems: different languages use different scripts, which may be written in different directions. While most scripts will be alphabetic, using a small set of symbols (say less that 100) with which to write their language, by no means all are. The important languages of the Eastern Asian countries are ideographic and use a repertoire of many thousands of symbols. Many languages are written from left to right in lines which progress from top to bottom of the page, but the Arabic script is written right to left top to bottom, and some ideographic languages are written top to bottom right to left. Some scripts are composed of characters that can take a number of forms, like the capital letters of the Roman script and the letter variants in Arabic, or have compound characters as in Devanagari. Diacritics may be important, effectively extending the alphabet. There are different conventions regarding line breaks and hyphenation.

Word structure: the individual units of language used to express the meaning and typically separated in some manner in the orthography (eg by blank space or a break in the cursive flow) may be richly structured. For example, in the Finno-Ugric languages of Finalnd, Hungary and central Asia, and in some South-Asian language like Bengali, a single word may take many thousands of forms as it plays different roles within a sentence – we say that it is richly inflected. Other languages allow the free construction of compound words, like German. The general rules regarding the formation of words is known as morphology.

Collation and sorting: different languages using the same (or largely the same) alphabet and script system may arrange words in different orders in dictionaries. Of particular significance here are the accents, and "compound" letters like the Spanish "ll" which sorts between "l" and "m". Some Devanagari letters seen as compounds in Hindi and treated as distinct letter in Nepali.

Spelling: different regions using the "same" language may expect different spellings for a wide range of words – a well-known example of this is the difference between America and UK English. In South Asia the spelling (orthography) of the northern Indo-Aryan languages may be varied with font size. From this arises the equivalence between words that have the same meaning but differ only in their spelling.

These aspects of language may be of importance wherever language appears in an application. The most obvious and visible part of a system is its user interface, where all these issues are important.

When text data is stored and manipulated by the computer, these linguistic issues may also become important, particularly where searches and matches are made. For example, a word processor searching for a whole word must understand what is meant by a word in the language and may need to take into account the morphology of the language.

Some of the documentation supporting a product may be specialized in its usage, and not require translation outside a set of international technical languages. Other material of a training nature may need more than just translation since examples may be cultural specific and may need to be changed – an example here is the use of baseball in examples for the US market being changed to football for other markets.

2.2 Culture

Social anthropologists define culture as "those socially transmitted patterns for behavior characteristic of a particular social group" and "the organized system of knowledge and belief whereby a people structure their experience and perception." (Keesing 1973: page 68). So it includes the non-linguistic human conventions that distinguish one group of people from some other group, their patterns of thinking, feeling, interacting, and ways of recording this in their art and artefacts.

These cultural aspects include the way time and dates and currency are denoted. Here there can be subtle variation, such as the use of "," and "." in the representation of numbers, through to the particular symbols used to denote currency. Typically there may be a variety of ways of writing equivalent information, such as we see in the writing of dates. One particular example is the way western number systems have used thousands and millions as major groupings, while in South Asia lakhs and crores (100 thousand and ten million) are used.

Calendar systems can vary: the western calendar works with a solar year with a Christian orientation for its origin for counting years: it uses various devices to synchronize days with years, through leap years, all worked out by calculation based on scientific theories of the earth's orbit around the sun (see Duncan 1998 for a history of the Gregorian system). By contrast many parts of the Islamic world use the Hijira calendar with is lunar months and yearly cycle of 12 lunar months with no attempt to synchronize with the solar year, while synchronization between days and months is not done by calculation as in the west, but by direct observation of the moon. A person could be 30 years old in the western calendar and 33 years old in the Islamic calendar. The Bikram calendar system of Nepal (historically widely used in South Asia

before the colonial imposition of the Gregorian calendar) uses a solar year but lunar months, and synchronizes these by having an extra month every three years and other devices (Pillai 1911). These calendar systems embody within them the deeper cultural values, so that for the West knowing the calendar for years in advance through the calculations of the Gregorian system is important – for Nepal it has always been sufficient to have the calendar for the next year determined a few months before the start of the next year.

Cultural areas are full of pitfalls for the unwary. For example, an icon depicting a hand held up with palm towards the viewer, commonly used to indicate 'Stop!' or 'Danger!' in the Apple Macintosh and Windows interface, has a completely different meaning in Greece, where it is extremely rude. The product name "Nova" could denote "new" to readers in one locations, but "does not go" in some other location. Colors may also have radically different connotations in different cultures, so for example red is a warning color in Western countries, but in China it illustrates joy, while white is the color of mourning and black is lucky: this could be important in the coloring of maps in GIS systems, since in the west red might well be used to denote danger zones, but lose that connotation elsewhere. Any of the many books on cross-cultural communication will have numerous examples (e.g. Jandt1995).

2.3 Local conventions and practices

Local practices are often specified by legal and professional bodies. These consist of factors such as market-specific computing practices: communication system interfaces, law and regulations, and financial accounting rules. For example, most Western European and America countries express salary in monetary units per year, but in Greece salary is expressed in monetary units per month and the annual salary consists of fourteen monthly payments.

Also included here are differences in tax regimes, and more generally legal constraints, that can influence the way software must work.

3. Current approaches to software globalization

Current approaches to the localization of software focuses on three topics:

1. the choice of character codes
2. use of locales
3. use of resource files.

Facilities for these are available in all the popular operating systems.

3.1 Character codes

The traditional 7 or 8 bit code ASCII (more properly ISO 646) for the Roman script permitted limited national variation across European languages, but has proved inade-

quate for more general use. Various extensions to ISO 646 have been proposed, from registration of national variants through to complete new coding systems that would handle all alphabetic languages. These were based on single byte encodings which necessarily limited the total number of codes to 256 and the alphabet size to around 100. In multilingual applications there has to be a shift between coding tables, and the meaning of a single code is context dependent.

Ideographic scripts just cannot be handled in this way, and multiple bytes are necessary. The coding standard that has emerged as the best available is Unicode, which requires two bytes per character, and its ISO extension 10646 which uses four bytes per character. It is claimed that this gives the ability to store all known scripts within the single system, with room for expansion – however inspection of the tables (Unicode Consortium 1992 and 1997) shows that many scripts and languages are as yet not included. In multilingual applications there is now no need to shift between coding tables, and the meaning of a single code is unambiguous.

On the surface of it, the use of two bytes per character by Unicode may seem wasteful, but in most applications text is stored with considerable volumes of management information and the extra byte does not double the storage requirement, perhaps only increasing it by half as much again.

No single character encoding scheme has found favor, though there does appear to be a drift towards Unicode and ISO10646. Microsoft's Windows NT has adopted Unicode, Windows 95 has not done so, though Windows 98 does and the projected Windows 2000 arising from the convergence of NT and 95/98 technologies will do so fully. Other software suppliers are looking to Unicode for support in multilingual applications, and so for example the database supplier Sybase has adopted Unicode as part of its distributed multilingual strategy. It does seem that Unicode is the way of the future.

3.2 Locales

A "locale" is a collection of all the conventions that characterize a single user community:

1. the Script, a reference to the code tables to be used, and any special rendering software
2. the Language, leading to hyphenation rules, morphological rules, and similat
3. Number, time and date system and conventions, the format for input and output and associated software to transforming these to and form the internal representations.
4. Monetary system symbols and rules
5. Messages to be used, for error messages, help, and similar in the language and terminology of the locale

and so on – the exact capabilities depends upon the operating system offering the locale facility. Kano (1995) lists 94 locales for Windows platforms, but many of these are for variants of dominant European languages with there being only 38 different

languages, plus some 16 others defined but not offered. A lot of the world is unsupported through locales though they are supported through Unicode tables.

When a user buys a system the operating system's locale may already have been set, so that the system interacts with them appropriately. Where several different communities are served, the locale may be set at time of sale. For example, the Regional Settings in the Control Panel in Windows 95 gives some user control over the local details. Application programmes may also be able to set their own locale, and can enable locales to change dynamically.

3.3 Resource files and Dynamic linking

To be able to select a locale, and change it dynamically, you need the two basic facilities from the operating system: a level of indirection where the locale sensitive elements can be named but not bound into the application code, and the ability to link in these locale sensitive elements either statically or dynamically.

Most operating systems have suitable facilities. There will be "resource files" or "message catalogues" in which the locale sensitive elements can be placed, such as:

- user interface text for window titles, menus and prompts, error and information messages
- interface and report layout information like position, size, font, colour, intensity, text orientation
- help text
- symbols, graphics and icons
- sound and video
- software to print or display numerical values.

The application is designed to access these resource files using locale parameters to select the appropriate resource for a given locale. Screen layout is designed to reserve the space needed for possible text expansion during localization (Finnish requires double the space of English, while Arabic requires a lot less space than English), where dialogues and other elements may need to be resized.

4. New approaches – software architecture plus language generation

Existing practice can be taken forward by drawing upon current and recent research in software architectures and linguistics. The work reported here arises from the Glossasoft project reported by Hall and Hudson (1997), particularly involving the work of a team in Finland at VTT and a team in Greece at Demokritos.

4.1 An internationalization architecture

The idea of locales backed up by resource files and dynamic linking is readily generalized to an application programmer interface (API) and this is clearly the next step. OS standards like Posix have made steps in this direction, but what is appropriate must be determined by global user need. All the culturally and linguistically sensitive software components need to be separated from the core of the application. This leads to the kind of architecture shown in Figure 1. The locale sensitive elements are shown at the top and right of the figure, separated by the heavy line from the internationalized locale independent core of the application. Many of the local sensitive elements are associated with the human computer interface, but some may be associated with the application. Whenever the application manages data that relates to the external environment, this is potentially locale sensitive.

The locale dependent components at the top and right should be designed to be plug-compatible so that they can be easily replaced with others performing the same function but for different locales. The heavy line denotes an Application Programmer Interface (API).

Contrast this approach with that currently supplied by operating systems using locales and resource files. Some level of standardization is necessary to identify the locale data structure and its components, but no standardization is used for the formats and protocols for resource files. The interface in current practice is very low level, whereas the interfaces of Figure 1 work at the level of linguistic rules and the meaning of messages.

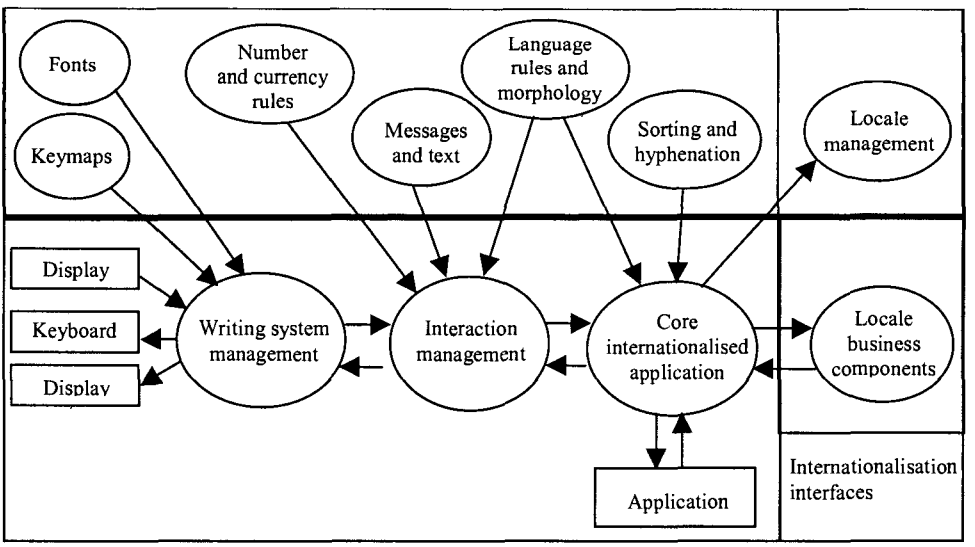


Figure 1. Internationalized interaction architecture.

Current methods would simply give messages numbers, and at the most leave a few slots in these messages for variable data like file names. What applications need to work with are the meanings intended to be conveyed to the user, with this being presented in the language determined by the locale.

Transforming meaning into the words of some language is known as language generation. The architecture of Fig. 1 makes language generation, or a limited form of it, integral to the architecture. This also fits in well with some current approaches to HCI, which suggests that the interactions with the user should be mediated by a knowledge model of the system, a model which is intended to capture the user's view of the software. This user conceptual model is itself locale sensitive and should be changed during localisation.

4.2 Language generation

The handling of error messages gives particular problems when they need to be composed from several parts. When the parts come together they may need to be modified in their grammatical form so that a grammatically acceptable text results. More generally word order may need to be changed and small atomic sentences combined, in order to produce a natural looking solution. In the Glossasoft (Hall and Hudson 1997) project two approaches were taken.

The first approach was the use of message templates, messages with slots into which you can substitute actual values depending on the context in which each message is generated. After substitution there will need to be some grammatical tidying up.

For example, instead of using four messages to announce that one of the four disk drives of the system is damaged, you could use the message template:

```
Disk <number of disk> is damaged
```

If the user is unfortunate enough to have two damaged disks, the application could also use this message template to announce that:

```
Disks 1, 2 are damaged
```

but would need to access a morphological generation routine to make the appropriate adaptations. Using such extended message templates is especially important for languages with many inflectional word forms, for example synthetic languages such as Finnish, as it avoids the need to maintain all the different word forms inside the message catalogues. Using extended message templates also improves the organization of message catalogues.

At VTT a trial was undertaken on the software product OsiCon, a risk analysis support tool, and the approach was demonstrated successfully for English and Finnish.

However, you still have to maintain different extended message templates for the different languages supported. So far we have been looking at "canned text" systems. Canned text systems, in which the repertoire of messages is fixed and immutable, are typical of the situation we are usually faced with in localisation and internationalization. All possible messages have been anticipated and stored. This is even the case with the message template approach discussed above.

tion. All possible messages have been anticipated and stored. This is even the case with the message template approach discussed above.

The alternative approach is to use a general language generation approach (eg. Allen 1987). Language generation is capable of creating a range of messages that could not have been anticipated, messages which are contingent upon user actions and an evolving knowledge base within the system.

Central to this whole process is a knowledge model of the application and its supporting software systems, about which messages are to be generated in the context of specific user actions and difficulties.

Some event occurs in the system which triggers the whole process. This could be a request for help, or some erroneous user action which requires a message to be given to the user. The system must in general know something about the user, because what it wants to say to the user will depend upon what the user knows. As a function of what the system assumes that the user knows, the system will select information to present to the user. The outcome of the selection would be a number of abstract linguistic elements, or "speech acts".

This message is still very abstract, and particular word choices (or phrase choices) must next be made. This choice will affect the focus of the message to be generated, which can influence the "tone" of the message. For example, there is a choice between active and passive voices "You typed the wrong character." and the "The wrong character was typed." dropping the agent in the second choice to remove the accusatory tone that would be unacceptable in many locales.

This leads to some knowledge structure, which indicates what text is to be output, and now it remains to generate the actual text. We can generate the text directly from the data structures, using a collection of mutually recursive routines which embody within them the syntax of the target language, or we can use a table-driven or rule driven approach.

It is only in this final stage that the specific features of the natural language being generated become important. Preceding these, the knowledge model itself may be largely culture independent, though it may make some cultural assumptions such as a specific conceptual model of the computer.

At Demokritos a small example for Hewlett Packard's VUE interface system was tried out, with several different levels of user expertise being selectable. This showed that the approach was very promising, but also very computer intensive. Clearly the template approach is to be preferred if possible.

5. Application to Decision Support Systems

Decision Support Systems for Sustainable Development may be very diverse, as can be seen in the various papers of this book. These will involve information from population distribution, agriculture and health, through to finance. They will be underpinned by technologies covering graphical information systems requiring the display of maps, database management systems with large repositories of information, and knowledge-based systems with methods of inferring facts from those actually stored, as well as the distribution of both processing and data. These systems will

need to be used by people covering a wide range of technical experience and expertise, drawn from a wide cultural and linguistic group.

Decision support systems have the general architectural form shown in Figure 2. At this point we are focusing on the supply of information for decision making available in a number of servers across national boundaries and thus across locales, to a number of terminals or clients where decisions are made. To this architecture will need to be added the internationalization architectures discussed above. This intersection architecture will influence the general DSS architecture in the following ways:

- The application software in the terminals in countries 1 to N will be localized for the user communities involved – there may be several. Where the user interface involves the display of maps in geographical information systems, map presentation conventions will need localization.
- Place names will be important in such systems, and constitute a special case – typically these would be recorded in the native language, but approved transliterations into other scripts and phonetic systems would need to be used, as in the use of Pin Yin for Chinese.

If at all possible information resources will need to be kept in locale-neutral form, or in some standardized form that enables generation and presentation in local form. Numerical data is the simplest, and in many cases may involve no more than display routines, though units for systems of measurement need to be taken into account. Time and dates could be more difficult, but are tractable. Text may give real problems, unless it can be stereotyped in some way, perhaps using some narrow domain knowledge model with some simple form of language generation, or by using a controlled language and a simple translation system like translation memory.

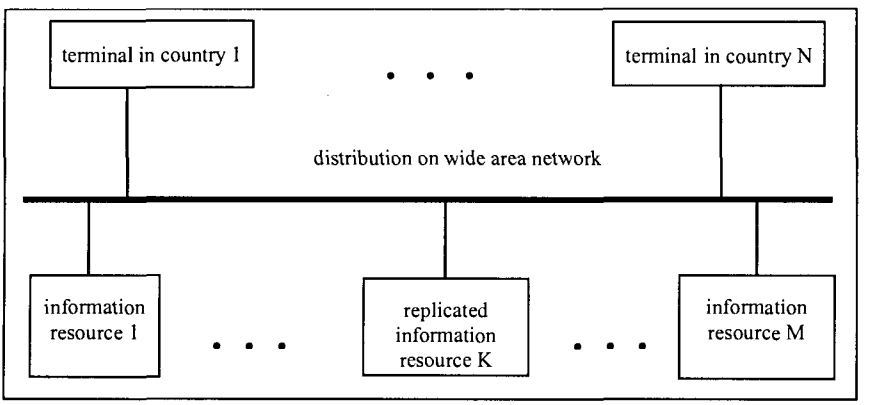


Figure 2. Outline architecture for Decision Support Systems.

Thus implicit in the approach must be the storage of data in some abstract form, with this mapped to the local terminal/client computers into a form that is acceptable

to and understandable by the local users. This approach should also be applied to the local viewing software which should be internationalized to a core product which is locale-neutral, with local mappings to the locale of the user.

In developing a multi-lingual DSSs we will need to carefully characterize all the information that is being stored and shared in terms of its locale assumptions, and then develop locale-neutral forms. Where text is concerned we will need to work to understand it, removing as much of the locale sensitive components as possible. This should be done across all the range of DSSs for sustainable development, so that the localization processes can be used across the range of systems, saving costs and increasing the range of languages and locales that can be covered.

The process of decision making, and what constitutes appropriate information to use in decision making, can vary greatly between cultures. The articles in this book give illustrations of this variety, though there is an undercurrent of acceptance of or aspiration to western modernist rational decision making. Social values may take precedence over efficiency, and family may take precedence over nation. Different ways of making decisions will affect the way the local client systems are configured, and may deny any ability to make decisions between individuals who are not co-located.

To illustrate this let us contrast the USA with Japan. Hofstede (1991) characterizes cultures in four dimensions:

- power distance – should decisions be imposed or should people be consulted? (p27)
- individualism versus collectivism – should the interests of the individual prevail over the interests of the group? (p50)
- masculinity versus femininity – is toughness and assertiveness valued over tenderness, nurturing and modesty? (p83, 92)
- uncertainty avoidance – are uncertain or unknown situations felt threatening? (p113)

Hofstede made a large survey of IBM employees around the world, and scored countries in the range of (roughly) 0 to 100. The USA scores were respectively (40, 91, 62, 46), concluding that the culture is mildly consultative, highly individualistic, and mildly caring and accepts uncertainty. By contrast the Japan scores were respectively (54, 46, 91, 92), concluding that the culture is consultative, collectivist, highly assertive and tough, and dislikes uncertainty. This kind of characterization is very crude, but indicative. More detailed commentary suggests that while in the US decisions may be made by individuals for the advancement of the individual or some externalized corporation and accepting uncertainty, in Japan the decision would be made collectively and for the collective good to reduce uncertainty. Heaton 1998 reports how decision taking in computer aided design requires a move from single person workstations to a collective shared design surface in which all participants can see each other and sense the arrival of consensus. Shatzberg, Keeney and Gupta (1997) report on the failure of email in Japan due to this collectivism and the need for people to see each other in order to be able to reach consensus, deferring to the most senior person if appropriate.

The decision making tools described elsewhere in this volume may be easy to localize in the sense of Sections 2 to 5. However localizing them in this deeper sense of the cultural process of decision making is much harder.

6. Conclusions

We have seen that software for decision making can and should be localized so that it works in the language of the decision makers and respects local cultural conventions. Localization of language and simple cultural information like dates can easily be accomplished using today's technologies, but the sharing of information between diverse groups may well require more advanced methods of language generation.

However, the actual process of decision making may not be capable of computer support to the same degree. Decision making is deeply rooted in culture, and may not be capable of distribution mediated by technology. But that does not mean that localisation is not worth doing, so that information can be shared and made accessible. Accurate information is after all a prerequisite for sound decisions.

7. References

Allen, J. (1987). *Natural Language Understanding*, Benjamin Cummings Publishing Company, Inc: California

Apple Computer Inc. (1992). *Guide to Macintosh Software Localisation*, Addison-Wesley, 1992, ISBN 0-201-60856-1

Apple Computer Inc. (1992). *Human Interface Guidelines: The Apple Desktop Interface.* Addison Wesley, ISBN 0-201-17753-6

Danlos, L. (1987). *The linguistic basis of text generation.* Translated from the French (1985) by Dominique Debize and Colin Henderson. Cambridge: Cambridge University Press.

Duncan, D. E. (1998). *The Calendar.* London: Fourth Estate

Hall, P. and R. Hudson (1997). *Software without Frontiers*, Wiley 1997

Heaton, L. (1998). "Preserving Communication Context', in Ess and Sudweeks (Eds.) *Cultural Attitudes Towards Technology and Communication*, Proceedings of the CATaC 98, London, 163-186.

Hofstede, G. (1991) *Culture and Organisations. Intercultural Cooperation and its Importance for Survival. Software of the Mind.* McGraw-Hill.

INSTA (Inter-Nordic group on Information Technology Standardisation) (1992). *Nordic Cultural Requirements on Information Technology*, INSTA Technical Report STRI TS3 1992.

Jandt, F. E. (1995). *Intercultural Communication. An Introduction.* Sage.

Jones, S., C. Kennelly, et al. (1992) *The Digital Guide to Developing International User Information*, Digital Press, 1992.

Kano, N. (1995). *Developing International Software for Windows 95 and Windows NT.* Microsoft Press.

Keesing, R. M. (1975). *Cultural Anthropology. A Contemporary Perspective.* Holt Rinehart.

Kennelly, C. H. (1991), *The Digital Guide to Developing International Software*, Digital Press, 1991.

Luong, T. V, J. S. Lok, D. J. Taylor, K. Driscoll (1995). *INTERNATIONALISATION Developing Software for Global Markets*, Wiley 1995

Madell, T., C. Parsons and J. Abegg (1994). *Developing and Localizing International Software*, Hewlett-Packard Professional Books, 1994.

Microsoft (1993), *The GUI Guide*, Microsoft Press, 1993.

Pillai, D., and B. L. Swamikannu (1911). *Indian Chronology, A Practical Guide*, Madras.

Shatzberg, L, R. Keeney and V. K. Gupta (1998). 'Cultural and Managerial Comparisons: an Analysis of the Use of Email and WWW in Japan and the United States'. *Proceedings of the 1997 Information Resources Management Association International Conference*, Vancouver, Idea Group Publishing, 296-300

Taylor, D. (1992), *Global Software: Developing Applications for the International Market*, Springer-Verlag, 1992.

Unicode Consortium (1991/2). *The Unicode Standard. Worldwide Character Encoding.* Version 1.0, Volumes 1 and 2. Addison-Wesley 1990 and 1991.

Unicode Consortium (1997). http://www.unicode.org/.

Uren, Emmanuel; Robert Howard, Tiziana Peritonni (1993), *Software Internationalisation and Localisation: An Introduction*, VNR Computer Library, 1993, ISBN 0-442-01498-8