

Key Challenges in Software Internationalisation

James M. Hogan Chris Ho-Stuart Binh Pham
Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
GPO Box 2434, Brisbane, QLD, 4001, AUSTRALIA
{j.hogan,c.hostuart,b.pham@qut.edu.au}

Abstract

The trend toward globalisation of products and services has brought a strong economic imperative to the development of general methods for the localisation of software to different cultural environments. While ad hoc localisation may satisfy immediate commercial objectives, its extension to multiple locales is not cost-effective and an integrated strategy is needed. In this more sustainable approach of software internationalisation, the requirements of disparate markets are addressed during analysis and system design, with the architecture developed so that localisation to a particular environment is straightforward, and involves minimal re-engineering.

Given the limited size of the Australasian market, detailed attention to the technology of internationalisation is of critical importance to the future of software development in the region, as is the availability of graduates adequately prepared for this environment. Thus motivated, this paper examines the state of play in a number of aspects of application level software internationalisation, with our focus the core technical challenges of the next few years, and the need to transfer such skills to graduates and practitioners alike.

Keywords: software internationalisation, software localisation, internationalisation

1 Introduction

Historically, commercial software has been developed for the English speaking – especially the American – community, with only a limited attempt to cater for other environments. However, globalisation of business and technology have heightened demand for software products *localised* to a particular language and cultural environment – demand which has been exacerbated by the spectacular growth of the world wide web. Minimally, such localisation must include translation of user interface strings to the target language, and adaptation of display formats to comply with local conventions for items such as date, time and currency.

Many of these issues are readily handled through standard programming language support, but others are surprisingly subtle, and even string translation has hidden complexity – such as that which arises through the absence of an identifiable equivalent term in the target language. When the process is extended to include cultural variations in the use of symbols and colour, and adaptation to local business processes such as the taxation regime, it is clear that suc-

cessful localisation depends upon a careful and time-consuming blend of technical and native professional expertise.

Yet if the cost of localisation for a particular environment appears high, the expense becomes unsustainable if incurred for each new language and locale, and some amortization over multiple adaptations is essential if the process is to remain commercially viable. This more systematic approach, termed here *software internationalisation*, requires that localisation issues be addressed earlier in the software development lifecycle, with architectural and subsequent design provisions to minimise re-engineering.

The future of the Australasian software industry depends in no small measure upon its response to these challenges, and the degree to which it supports small enterprises and young developers with information and tools. Yet, identification of an appropriate body of knowledge is more difficult than it may seem. Major international vendors have addressed these issues at both the operating system and key application level, but much of this work remains sensitive, and not always successful – even experienced vendors may have difficulty optimising their usage of internationalisation services and integrating the process with their existing workflows. More disturbingly, there has until recently been relatively little effort directed toward the establishment of *open*, industry-standard methodologies and documented practice – seemingly an essential pre-requisite if small and medium size enterprises (SMEs) are to compete in the global market.

Happily, there have been some developments in this area, with vendors increasingly willing to provide partial detail of their approaches (see for example (Gray 2003)) and ongoing efforts from the Localization Industry Standards Association (LISA) and related organisations such as OASIS to develop XML-based standards for the management and exchange of locale-sensitive information¹. The provision of open XML standards and the ready availability and adaptability of open source XML processing tools is one of the more exciting developments for the global-minded SME in recent years.

Yet the challenges which remain are also technical rather than merely educational. It is perhaps self-evident that solutions will be centred around the fundamental principles of good software engineering – modularity and re-usability – but this does not diminish the need for a clearer understanding of their application in this context. Such principles naturally inform key technological decisions – for exam-

Copyright ©2004, Australian Computer Society, Inc. This paper appeared at the *Australasian Workshop on Software Internationalisation (AWSI 2004)*, Dunedin, N.Z. ACSW Frontiers 2004, Conferences in Research and Practice in Information Technology, Vol. 32. J. Hogan, P. Montague, M. Purvis and C. Stekete, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

¹Some of these standards – notably TMX and XLIFF – are considered in section 2. Given the range of standards and organisations involved, we shall reference only the LISA standards page: <http://www.lisa.org/standards/>, which contains links to each of the principal initiatives, and Yves Savourel's compendious guidebook (Savourel 2002).

ple the division of localisation tasks between source code modification, link phase binding and run time resource loading. Yet these decisions are equally subject to economic factors, the drive not merely for quality, but for cost-effective and sustainable quality. And given the dependence of successful localisation upon the approval of the native speaker, it is this latter factor which is in many respects the more influential, determining the prerequisite skill set and background of translation staff, and even the degree of tool support acceptable to the vendor.

In this paper, our treatment is necessarily slanted toward those issues which we believe are amenable to a technical solution or simplification, our objective being to identify some core challenges and to relate them to mainstream research programmes in the hope of fostering solutions. Central among these challenges are problems of text translation and management (section 2), and the process issues of architecture and language support (section 3.1), and quality assurance (section 3.2).

At a more mundane level, standards are particularly important in the present context, given the nature of the task and the need for co-operation across borders. The most recent developments in XML-based approaches are considered within section 2, but some of the more fundamental work is outlined in section 4, noting the considerable success of the standards movement to date, and its key challenge for the future – the problem of cultural adaptation. The treatment of cultural adaptation is here somewhat superficial, serving primarily to indicate the scope of the challenge if the process is to be automated, and offering an interim solution based upon extensions to the locale standards framework.

We conclude with some guarded speculation on the future of software internationalisation in the region, and the challenges for the academic and professional computer science community if the issues are to be addressed successfully.

2 Translation and Tools

Issues of text translation lie at the heart of software localisation, and act as a touchstone for the health of the discipline. Developments in translation and tools are necessarily linked to improvements in programming language and operating systems support, which in turn serve to limit the scope of the research problems remaining.

While tool support is an integral part of modern translation, it must be emphasised from the outset that while significant advances have been made, machine translation remains a challenging task. Moreover, the use of machine translation in software internationalisation is likely to prove problematic even when the dreams of the language technology researchers have been realised, due to the unusually high frequency in software translation work of novel word forms and combinations for which no ready equivalent exists in the target language. Such a problem appears frequently in the selection of appropriately intuitive key or user interface button labels, and the value of the experienced translator over the novice may lie in the quality of this kind of selection (Dohler 1997).

Thus, since localisation at present requires significant intervention from human translators, the goal of the efficient software vendor remains one of optimising the use of such services wherever possible.

In an earlier phase of globalisation of the software market, text translation typically proceeded through

an ad hoc attempt to localise existing software designed solely for the English speaking – and especially American – world. Such software is by definition poorly architected from an internationalisation perspective, requiring substantial redesign of the product source code to isolate language and locale sensitive material. Moreover, localisation was initially fragmented, driven by the commercial need for penetration in a particular region, and in consequence requiring a limited range of language expertise. Subsequent localisation was often characterised by duplication of effort, and suboptimal usage of translation services. Tool support for such a process was similarly ad hoc, perhaps focused more upon re-engineering the product architecture than upon support of the process of translation.

While numerous vendors remain at what might be described – by analogy with the Capability Maturity Model (The CMMI Product Team 2002) – as Level 1 internationalisation firms, the practice of software internationalisation has evolved substantially, to the point that many vendors have addressed the most fundamental architectural issues, and modern programming languages and operating systems now provide direct support for the process.

Successful vendors have developed the following strategies for minimising the impact of internationalisation issues in their products, and these have become de facto industry standards:

- The extraction or “externalisation” of user interface strings into resource bundles and message tables – thus limiting the task of the translation service to that of direct translation of a database of known strings;
- Subsequent careful control and identification of the original strings, their translated target language equivalents and the ability to incorporate the *context* associated with each translation performed;
- Almost universal outsourcing of translation tasks to specialist software industry translation services, often based in key internationalisation centres such as Dublin and Beijing; and
- “Dumbing down” or simplification of string content and associated context in preparation for submission to the translation services or machine translators.

Each of these core activities requires careful and integrated management of locale-sensitive material and its associated meta-data. Not surprisingly, given its support for localisation through locales and unicode, significant efforts have been made to exploit the technical glue of XML-based formats and processing. These efforts have culminated in standards for the exchange and storage of localisable material (XLIFF - the XML Localization Interchange File Format), translation memories (TMX - Translation Memory Exchange) and terminology databases (TBX - Term Base Exchange). While such standards offer immediate advantages to localisation savvy developers, particularly among the SME community, perhaps their more important role will be to act as scaffolding within which more significant technical progress may be made.

Three of the key activities are considered in more detail below:

Externalisation of UI Strings: Extraction of user interface strings and other language sensitive material to resource bundles is now an industry standard,

with only those vendors content with a single, monolingual market failing this initial test. Yet, again the apparently straightforward translation task is complicated by the embedding of markup codes (such as those used in RTF-based Windows help files) and hyperlinks within the supplied files. As noted above, the experienced translator has an important role to play in coining appropriate novel word combinations and even neologisms in the target language corresponding to the intention of the original. This practice is often hampered by the limited provision of system specifications to the translation service, and by the pressures of time to market – in which a “mediocre translation . . . delivered on time is much more valuable than a perfect one . . . which is three days late” (Dohler 1997).

This process may become more standardised and readily managed through the advent of the XLIFF, and the ready availability of XML formatting and transformation technologies to support transitions between stored repositories, resource files and the localised application. Of particular value here, for interaction with human translators and for research into intelligent string management, is the provision for extensive context and annotation fields as part of the standard, motivating more extensive and usable tool support. Yet one must insert a caveat – recent industry surveys suggest that the uptake of the approach has been relatively slow.

Maintenance of a String Database: The benefits of this strategy are self-evident, as translation costs are geared to the number of words in the original document – with prices ranging between US\$0.30 and US\$1.00 per word (Ierner 1999)². However, optimism must be tempered by the realisation that the usefulness of the individual string is governed by the independence of its translation from surrounding context.

This area represents the most important opportunity for research, through the investigation and development of intelligent and multi-lingually aware systems for the management of source string databases. Of principal concern must be the robust and *context and grammatically sensitive* identification of similarities between novel source language strings, and those for which a translation is registered in the database. Appropriate technology of this kind has the potential to realise enormous savings in translation service costs, and in the reduction of time to market for new systems. As suggested above, the TMX standard, in partnership with the XLIFF format, provides an appropriate framework within which this work may be successfully hosted.

Preparation of text for translation: This pre-processing step is particularly important in the context of machine translation, as the reduction of ambiguities greatly simplifies the task and enhances the reliability of the underlying grammatical model. The approach may also assist translation service staff, removing potential ambiguity and reducing the word count.

One major machine translation vendor summarises these issues nicely in their recommendations to potential users:

“Try to eliminate or alter sentences and phrases that have multiple interpretations. Remove unneeded words from a sentence or split a long sentence into two shorter ones” (Lan n.d.).

²Indeed, the authors are aware that one leading vendor – in spite of careful attention to limiting the need for new translation – paid almost US\$10 million to translation services during their most recent upgrade, involving more than two dozen target languages.

The language technology community have investigated similar issues through the development of a computationally sound simplified English, such as the “Controlled English” of Mollá and Schwitter (Mollá & Schwitter 2001). While acceptable Controlled English may be too restrictive for all applications – indeed the construction of the equivalent CE form requires translation of plain text into logical forms – the approach is nevertheless a important pointer to the future of assisted and ultimately automated translation.

While this analysis is necessarily speculative, it is likely that an important pre-requisite for the widespread adoption of such an approach will be the availability of target language specific text generation frameworks, which will act as a complementary service to any text simplification engine on the source side – or perhaps as a client drawing upon a repository of a more formal semantic specification such as that discussed above. This kind of technology has been adopted in a number of industry applications, with one of the better known systems being the CSIRO’s Isolde project (Paris 2002).

Discussion: In summary, the problem of text translation remains at the core of internationalisation efforts, and there appears little prospect of a fully automated solution to the problem in the immediate future. While assisted translation applications are already available, the most important problem at this point lies in string maintenance and the avoidance of unnecessary and costly translation. To date, this effort appears to have been limited to careful indexing strategies and to our knowledge there are no applications in existence which perform intelligent trawling of an existing database for useful substrings – existing translations which may be exploited as part of the task even if there are differences in the original context. The introduction of the TMX standard and its ready interaction with the XLIFF format provides some preliminary steps in this direction, offering a framework in which such applications may operate successfully while limiting the degree of data re-engineering.

Intelligent string management is a problem which confronts a number of difficulties known to the machine translation community, but these are ameliorated by the more limited scope, and by the ready acceptance of a failure condition should an appropriate confidence prove unattainable for the given query. Moreover, the costs of translation suggest that even modest, incremental reductions in the translation task will be welcomed, and that integration of such a system with appropriate text generation facilities will bring enormous interest from the industry.

Yet such systems are of little use if the advantages they provide are overwhelmed by the cost of additional translations resulting from poor software engineering practice. These issues are considered in the following sections.

3 Software Engineering Practice

3.1 Software Technology and Architecture

While our focus in this and the subsequent section is upon issues of software engineering process, the selection of an appropriate software architecture cannot entirely be divorced from the class of implementation language, and it is useful to consider briefly the level of internationalisation support provided by industry standard languages. Such support has now advanced to the point that many of the key research problems in this area have been solved, with the convenient input

of pictographic characters the one glaring exception³.

The development of integrated programming language support for internationalisation may be traced to the release of the Java programming language in the mid 1990s, and we shall consider its facilities as representative, referring to competing products only as necessary. Java was the first mainstream offering to incorporate 16-bit Unicode standard (The Unicode Consortium 2000) characters as part of the language. Moreover, the associated class libraries provided extensive support for a substantial number of ISO standards for locales, incorporating date, number format, currency and language specifications.

While usage of these classes appears somewhat cumbersome – with the necessity of constructing both `Locale` and `Formatter` objects prior to output of a currency value, for example – the additional line or two of code seems a small price to pay for the inherent utility of the approach. In any case, the tutorial material available with the SDK and the considerable number of professional references on the market each provide appropriate test programmes, with boilerplate code which may be readily integrated into more elaborate applications.

Internationalisation support in Java is not, however, limited to these more obvious considerations (Horstmann, & Cornell 1999). The `Locale` object may be used as a passport to localised behaviour in the following domains:

- Sorting and ordering at various levels through a `Collator` object;
- Text boundaries, through a `BreakIterator` object; and
- Management of resource bundles, through a `ResourceBundle` object.

The advent of similar support within the latest release languages from vendors such as Microsoft (through C# and the availability of associated class libraries to managed code C++) serves only to emphasise the earlier point that most of the important challenges in this area have now been addressed and the fruits of this research are now widely available. While it would appear that there is some scope for further extensions to locale support, perhaps even at the level of tables of lexical equivalence, such resources are extremely unlikely to be integrated within the fabric of the language in the same way as the Unicode characters are in Java. Indeed, a prime example is provided by the International Components for Unicode libraries (International Components for Unicode n.d.), which are a natural extension of the JDK facilities.

Thus, even if such facilities were to be supplied by the language vendor, they cannot in practice be distinguished from third party suppliers offering support libraries for a multitude of purposes. In this sense, we are travelling beyond the scope of the programming language and are leading into the domain of the software architect, in many respects by way of the software component.

Much of the available literature would suggest that progress in internationalisation sensitive software architectures has been limited – and indeed this impression is in part a consequence of the paucity of articles, the *redbook* document from IBM (IBM Globalization Redbook n.d.) being an honourable exception. In this

³As noted earlier, our focus here is upon applications development and this issue is best regarded as an operating systems issue – and indeed the problem is receiving a good deal of attention from both Microsoft and the Linux community.

discussion we will analyse the two extremes of the debate, offering some limited resolution of the conflict.

On the one hand, internationalised software could be based on general purpose processing engines, which can then access and use tables of information specific to a particular location to give appropriate behaviour in that context. This has the advantage that the software can quickly be adapted to new locations, and gives the possibility of a single binary release which works everywhere, since it is only the tables which need to be modified, not the processing engine itself. This style of architecture is proposed by Kokkotos and Spyropoulos (Kokkotos & Spyropoulos 1997).

The central limitation of this approach lies in its capacity for scope creep – yielding a process engine of byzantine complexity, anticipating problems which may never, ever occur and offering grossly inappropriate levels of functionality.

At the other extreme lies the pure minimalist approach, in which the application is at the outset designed as far as possible to exclude consideration of locale dependencies, with the initial focus upon core functionality. In principle, localisation support is then introduced as a goal of subsequent refinements of the product. Such an approach would appear well suited to the light weight, highly iterative processes now gaining wide industry acceptance for software development; the “agile” process movement.

A disadvantage here is that even a light weight process can become slow and expensive if it requires repeated rework for each new circumstance. The usual agile solution is to *refactor* parts of a product when it becomes clear that a portion of a system is being repeated or revised on multiple occasions. Relevant code is then refactored to ensure more generally applicability. Purely bottom-up solutions of this nature are likely to prove suboptimal for internationalisation, and indeed internationalisation may provide a useful counter example for attacks upon the architectural bona fides of methodologies such as eXtreme Programming⁴.

While it is plausible that the optimal approach lies between such extremes, it is difficult to discuss the general case without degenerating into banality. Plainly, in designing for a global market a developer needs to maintain a particularly clean separation of concerns between the human-computer interface, and the universally applicable data processes that may exist; and this requires due care from the outset.

On the other hand, there remains a clear benefit to simplifying the interface of globalised applications as much as possible, thus providing some natural bound upon the complexity of adaptation.

Ultimately, it is our belief that the best solution to the architectural problem lies not in the specification of the internationalisation structure per se, but rather in the specification of a standard interface for internationalisation components, so that developers may phrase their localisation requirements in terms of appropriate messages through the necessary API. In one sense, we are proposing that the *industry* adopt overall architectures in the style of Kokkotos and Spyropoulos, with developers able to select only those components necessary for their application. Moreover, such an approach offers the promise of a healthy third party software component market, particularly in important business domains.

Such applications may provide a fundamental test of the viability of the component market. Different

⁴It is a common theme among advocates of the Agile approaches that even complex architectures may emerge in this fashion, an issue of some controversy among practitioners.

locations may involve markedly different legislation, regulations and taxation regimes with a substantial impact upon the processing that must be performed. For software to be adapted easily to a new location, it should be easy to alter the business rules by which it operates, and this in turn suggests that even some underlying processing rules should not be hard coded into the application, but be included through some external mechanism. Ideally of course, the developers should be able to purchase off-the-shelf components which embody the regime in which they wish to operate, but even with the best of intentions this will never be the complete solution for a complex application. Some mixture of off-the-shelf and in-house internationalisation components would appear to be unavoidable.

Inevitably the problem of an internationalisation architecture is linked to the integration of internationalisation workflows with modern software engineering processes – a problem which we shall consider from the perspective of quality assurance in the next section.

3.2 Quality Assurance

The principle that quality cannot be tested into software has a clear corollary for internationalisation: one cannot translate internationalisation into software. Internationalisation issues need to be considered from conception through to packaging and delivery, and there are many distinct aspects to be addressed.

The use of maturity models for development processes has proven highly successful in managing software quality, with the Capability Maturity Model family being particularly influential, with regular extensions to incorporate additional aspects of production.

Given the inherent complexity of internationalisation, and its potential impact upon the development process – from requirements analysis right through to the tasks of support and maintenance of the delivered product – we believe that this is a problem which is well suited to a comprehensive maturity model.

One reasonable approach is to adapt the Test Process Improvement Model (Koomen & Pol 1999), which uses a matrix ranking testing performance across twenty key areas, giving a comprehensive picture of the maturity of testing within an organisation. By analogy, one might straightforwardly define a number of key internationalisation areas – string management, translation, acceptance testing in target environments, internationalised support, change management across target environments, matching of initial requirements to different international contexts – and use a similar ratings system to identify areas in need of improvement.

A study of companies which have been successful in developing a product for diverse international markets is important to guide the identification of key areas and maturity levels for such a model, and conversely, the goal of producing a model in this form gives a useful framework for such a study.

An inevitable difficulty with this approach lies in the protection of the commercial interests of the participant company; those companies which demonstrate high maturity in this area may consider this to give them a critical strategic advantage over competitors. On the other hand, one might reasonably argue that an improvement in the profile and reputation of the regional industry will be of benefit to *all* of its developers, and for some companies there may be a happy coincidence of altruism and self-interest.

Notwithstanding our opening corollary, however, such a model cannot succeed without due attention to testing, and regression testing is especially important as an application is adapted to each new locale. Moreover, testing staff require highly specialised skills, and ultimately there can be no substitute for third party evaluation by experts who are native to the target culture and language⁵. Formal certification of third party evaluation or testing services would help, and there are some relevant standards that apply, such as ISO 2384 (International Standards Organisation 1997), which describes how translations should be presented.

While all of these matters are properly considered as quality control for localisation, internationalisation takes the process one step further, and aims to be easily and naturally localised for different cultural and language contexts – with perhaps a greater likelihood that changes may be made at different stages of the life cycle and by different people. Particular care needs to be given to consistency in the case of distributed applications working at once in several diverse locales. This is one aspect of internationalisation which has not received a great deal of attention.

In the preceding paragraphs we have examined a number of specific aspects of internationalisation and localisation through the lens of traditional notions of quality control. While the adoption of better quality management is of considerable importance, the internationalisation domain serves to highlight issues of process communication and documentation. Here particular attention must be given to ensuring a clear separation of concerns between the general problem or task addressed by the software, and issues that relate to the choice of locale.

It is a well understood axiom of effective quality management that it is not enough to address the matter from a technical perspective; it is also important to focus on the attitudes and interests of people in the organisation. Personal issues may become particularly important when there is a need for people with diverse background and subtly different cultural assumptions to be working on and evaluating a project.

Automated management of such cultural variations remains an open problem, and some preliminary, standards based approaches are considered as part of the following section.

4 Internationalisation Standards

This section concerns the role of the international standards movement in addressing key problems in software internationalisation, focussing upon its success in character encoding and locale specifications as a model for the remaining challenges – notably the profoundly difficult issues of cultural adaptation.

International standards pertinent to software internationalisation emerged primarily through the explosion of interest in email and Internet services, with developments by numerous Working Groups and Technical Committees over the past decade – the International Organization for Standardization (ISO) being involved in a number of character set and API definitions, with comparable activities at a regional (trans-European or Asian) level.

The most obvious success of the standards movement in this domain lies in agreements covering the encoding of different languages. Such work has developed since the mid-eighties, commencing with the

⁵To some extent this skill combination is available from international students.

sixteen encodings of the ISO 8859 series of 8-bit character sets. This approach is inadequate for the pictographic languages, most notably Chinese, where thousands of characters are required, or the Korean Hangul script, which uses a phonetic alphabet, but arranged in two dimensional form that makes it more convenient to represent each complete syllable as one "character".

The solution was Unicode; a universal character set able to represent every character in every language in common use today. The Unicode Consortium maintains the standard (The Unicode Consortium 2000), in close co-operation with ISO⁶, and the approach has quietly revolutionised the industry, particularly in the period following its integration within SUN's java development kits.

Deployments of this nature within industry standard languages and operating systems are now routine, reflecting the influence of the movement from the arrival of *locale* within POSIX (The Open Group n.d.), through to its successors in the ISO umbrella project *Functionality of the internationalization of applications* (working group JTC1/SC22/WG20; the overall framework is described in (International Standards Organisation 1998)). Such work on standards is at once significant *and* elementary, establishing consistency of approach, and a preferred way of capturing and storing local conventions, which may be accessed by globally aware applications from different vendors.

Such approaches may even extend to the more complex problem of cultural adaptation.

4.1 Cultural Adaptation

Internationalisation is a matter of special importance for interactions between Asian and "Western" nations, with the character set burden supplemented by substantial variations in cultural conventions, some of considerable subtlety. Such *covert* cultural factors (Yeo 1996) are often subdivided into four categories:

- **Mental Disposition:** a culturally specific set of user interface design preferences, concepts of usability and priorities for functionality of the software;
- **Perception:** Perception is here used to encompass the whole psycholinguistic apparatus of interpretation of metaphors and symbolic representations (including colour, choice of icons, graphical art work, and audio signals);
- **Social Interaction Rules:** Social interaction rules are related to perception, but refer more specifically to the conventions governing interpersonal communication, through verbal dialogues, hand gestures, body language and facial expressions; and
- **Context of Use:** This refers to the environment within which the software is used – both the physical work space and the organisation in which it is deployed.

These factors have a enormous impact upon the design of the user interface for software products, with

⁶The character set of the Unicode standard 3.0 is identical to the standard ISO/IEC 10646-1:2000, and the two organisations maintain close links to ensure ongoing compatibility. The Unicode standard is the more comprehensive reference, as it addresses also a number of semantic properties for characters and character sets, like the direction of a script, orderings, and mappings between related characters.

some common colour schemes and iconic representations proving intuitive within one culture but profoundly unsuitable within another. While there has been some research on the impact of culture on user interface design (for example, Ito and Nakakoji (Ito & Nakakoji 1996), Herman (Herman 1996)), this work has found relatively limited application in software design, with training and set-up costs offered as the limiting factor.

While we would argue that interdisciplinary research in this area involving computer scientists, perceptual psychologists and cultural anthropologists is highly desirable, there is little doubt that such an enterprise will bear commercial fruit only in the longer term. From our perspective, adaptation to incorporate covert cultural factors must be viewed as a software engineering design and quality assurance problem – and its solution facilitated through the widespread adoption of standardised conventions. Such an approach should of course be seen as a minimal effort toward cultural localisation, with additional activities a potential market advantage.

Some developments have already taken place in this direction. The Asian Forum of Standardization for Information Technology (AFSIT) offers a *Data Book of Cultural Convention in Asian Countries* (CIC 1997) addressing both overt and more covert conventions – notably forms of address, colour significance, and taboo items. Such a compendium is not a standard, but it may offer the basis for one. Similar leadership has been provided in the web domain by the W3C through the development of version 4.0 of the HyperText Markup Language, HTML 4.0 (W3C 1999), which takes preliminary steps in this direction through support for direction of flow of text.

Ultimately, the transformation from lore book to configurable cultural locale is dependent upon a well-structured, extensible definition of such a locale. While it is difficult to imagine that culturally specific iconography could ever be automated, there is some hope that issues of colour, taboo and preference might be supported.

At present, however, the standards do not address the subtler aspects of cultural convention, and despite the honourable exceptions provided by some of the Unicode material, the documents fall well short of comprehensibility necessary for immediate use as part of the developer's professional library. To the extent that such guidance is not provided by the standards organisations, the responsibility will fall on the shoulders of the academic community, and this is an additional aspect to be considered in curriculum design.

5 Conclusions

In this paper we have examined a number of the more pressing problems of software internationalisation, with a particular focus upon the technical challenges and research opportunities presented. Given the commercial importance of the issues, however, it is critical that these opportunities be pursued in co-operation with the industry, cognizant always of the objective of economically sustainable adaptation of successful software to multiple environments. Within the context of the Australasian industry, the provision of solutions within the open source movement is of particular importance, given the high cost of translation support tools, and the advent of the new XML-based standards a healthy sign for the future.

In the various aspects of internationalisation reviewed in this paper, we can see ample scope for the researcher to pursue both intellectually interesting

and commercially significant problems. While there is a solid foundation of work in translation automation and support, the field is far from exhausted and we have identified in this paper the issues of string management and text generation from simplified sources as the most pressing research problems in the translation domain. With respect to languages and architectures, the core problems lie in the documentation of best architectural practice, and the integration of internationalisation workflows within software engineering processes – with the latter to incorporate some solution to the ongoing complaints of translation staff that they are given too little information and too little time to perform their task adequately. Such issues lie at the heart of good quality assurance, and formalisation of these responsibilities is an important step in reducing internationalisation related defects.

In longer term, the standards movement offers some hope for a solution to the most difficult of internationalisation issues, the management of covert cultural variation and the identification of a general purpose internationalisation architecture. But it must be remembered that standards are not a panacea, and that such relief as they provide may often be a long time coming. Developers should take advantage of what standardisation exists, and then focus on making a success with the rest; standardisation has a history of emerging from success, rather than the converse.

For the future of the regional industry, however, the most important focus must be the education of internationalisation and localisation aware developers. It is now apparent that academic interest in internationalisation is growing, with a number of universities across the world providing individual subjects and courses in the area. What is less clear is the extent to which internationalisation has transcended the boutique offering to become an integral part of the curriculum, part of the knowledge base and associated skill set without which an information technology graduate would not be complete.

While the academic computer scientist is properly focused toward the education of the next generation, university staff have a wider responsibility to engage in debate across the wider profession. Software internationalisation is an area of profound importance for the Australasian software industry, and one in which our activities may lead to wider acceptance of this imperative. Awareness of the importance of these issues is growing, but there remains a considerable amount of work to do if the imperative of globalisation is to reach the smallest of vendors. It is this partnership to develop the capabilities of the industry which is the most urgent of the challenges we face.

References

- CIC (1997), 'Data book of cultural convention in asian countries'.
- Dohler, P. (1997), 'Facets of software localization', *Translation Journal* 1(1). <http://accurapid.com/journal/01index.html>.
- Gray, A. (2003), "Making Sim Ship Work", *The Globalization Insider*, XII, (1.3). http://www.lisa.org/archive_domain/newsletters/2003/1.3/gray.html Accessed: October 2003
- Herman, L. (1996), Towards effective usability evaluation in Asia., in J. Grundy & M. Apperly, eds, 'Proceedings of OZCHI 96', IEEE Computer Society, Los Alamitos, CA, pp. 135–136.
- Horstmann, C., & Cornell, G. (1999), *Core Java 2, Vol II, Advanced Features*, Prentice Hall, Upper Saddle River NJ.
- IBM e-business Globalization Solution Design Guide, <http://www.redbooks.ibm.com/redbooks/SG246851.html> Accessed: October 2003
- International Components for Unicode, <http://oss.software.ibm.com/icu/> Accessed: October 2003
- International Standards Organisation (1997), 'Documentation – Presentation of translations'. ISO/IEC 2384.
- International Standards Organisation (1998), Information technology – Framework for internationalisation, Technical Report ISO/IEC TR 11017:1998, ISO.
- Ito, M. & Nakakoji, K. (1996), Impact of culture on user interface design., in E. M. del Galdo & J. Nielson, eds, 'International User Interfaces', John Wiley & Sons, New York, pp. 105–126.
- Kokkotos, S. & Spyropoulos, C. D. (1997), An architecture for designing internationalized software, in 'Proceedings of the 8th International Workshop on Software Technology and Engineering Practice (STEP)'.
- Koomen, T. and Pol, M. (1999), *Test Process Improvement: A Practical Step-by-Step Guide to Structured Testing*. Addison-Wesley, Reading, MA.
- Lan (n.d.), 'Language partners international, improving results with machine translation software', <http://www.languagepartners.com/reference-center/whitepapers/mtwp/mtartic.htm> Accessed: June 30 2002.
- Lerner, M. (1999), 'Building worldwide web sites', IBM DeveloperWorks. <http://www-106.ibm.com/developerworks/library/web-localization.html>. Accessed: October 2002
- Mollá, D. & Schwitter, R. (2001), From plain English to controlled English, in 'Processing of the 2001 Australasian Natural Language Processing Workshop.', Sydney.
- Paris, C. e. (2002), 'The Isolde project'. <http://www.cmis.csiro.au/iit/Projects/Isolde/>. Accessed: October 2002.
- Savourel, Y. (2002), *XML Internationalization and Localization* SAMS Publishing.
- The CMMI Product Team (2002), CMMISM for systems engineering/software engineering/integrated product and process development/supplier sourcing, version 1.1, continuous representation, Technical Report CMU/SEI-2002-TR-011, SEI. (CMMI-SE/SW/IPPD/SS, V1.1, Continuous).
- The Open Group (n.d.), 'Locale'. <http://www.opengroup.org/onlinepubs/007908799/xbd/locale.html>. Accessed: October 2003.
- The Unicode Consortium (2000), *The Unicode Standard, Version 3.0*, Addison-Wesley, Reading, MA.
URL: <http://www.unicode.org/unicode/> Accessed: October 2003.

W3C (1999), 'HTML 4.01 specification'.
<http://www.w3.org/TR/1999/REC-html401-19991224>. Accessed: October 2003.

Yeo, A. (1996), 'World-wide CHI: Cultural user interfaces, a silver lining in cultural diversity.', *SIGCHI* **28**(3), 4-7.