LOCALIZATION READER
2003–2004

SELECTED ARTICLES FROM LOCALISATION FOCUS AND MultiLINGUAL COMPUTING & TECHNOLOGY
Localization Reader
2003–2004

Published jointly by:
Localisation Research Centre (LRC), Department of Computer Science and
Information Systems (CSIS), University of Limerick, Ireland, www.localisation.ie, for the International Third-Level Localisation Teaching, Training
and Research Network (LttN), http://www.localisation.ie/LttNWeb/.

and

MultiLingual Computing
319 North First Avenue
Sandpoint, Idaho 83864 USA
www.multilingual.com
publisher of MultiLingual Computing & Technology

All articles published in this reader appeared first in
Localisation Focus and Multilingual Computing & Technology
during the period September 2002 through September 2003.

A PDF version of this reader is available at no charge to instructors and
students from the LttN Web site, http://www.localisation.ie/LttNWeb

The compilation of this reader was financed by the Localisation Research
Centre (LRC) and the European Localisation Exchange Centre (ELECT)
with the support of the European Union’s eContent programme.

© Copyright 2003 Localisation Research Centre
and Multilingual Computing.
The Localisation Research Centre (LRC) is proud to publish, together with Multilingual Computing, a selection of articles published over the past twelve months in both its own quarterly magazine, Localisation Focus, and in Multilingual Computing & Technology.

The idea for such a compilation first came up at a meeting of the third-level Localisation Teaching, Training and Research Network (LttN), operating under the umbrella of the LRC, in 2002.

The reader is aimed at students in third-level institutions studying localisation-related courses who otherwise would not have access to up-to-date reading material. It is a long overdue resource for lecturers who so far have had difficulties in assembling the latest news from the localisation community for their students.

This effort would not have been possible without the very generous and enthusiastic support from the staff of Multilingual Computing, especially Donna Parrish and Laurel Wagers. Thanks also to Patrice Fanning in the LRC who reviewed the articles.

We intend this reader to be the first in an annual series of readers, assembling the best articles published over the past year in the two leading publications in the field.

Enjoy!

Reinhard Schäler
Director
Localisation Research Centre (LRC)
reinhard.schaler@ul.ie
This reader consists of articles published in the magazines Localisation Focus and MultiLingual Computing & Technology from September 2002 through September 2003. It is intended for use by teachers of localization and translation and their students.

Localisation Focus is published quarterly by the Localisation Research Centre at the University of Limerick, Ireland. Reinhard Schäler, director of the LRC, is its editor.

MultiLingual Computing & Technology is published eight times a year (plus an annual index and resource directory) by MultiLingual Computing, Inc., in Sandpoint, Idaho USA. Donna Parrish is the publisher; Laurel Wagers is managing editor.

This reader is a joint publication of the Localisation Research Centre and MultiLingual Computing, produced by MultiLingual Press. Articles were compiled in a common general format, but not re-edited, so the different editorial styles of the two magazines will be apparent. While the Irish spellings are retained within the Localisation Focus articles, for example, this volume was produced in the United States, so the title is Localization Reader with a z. And to simplify formatting and double-sided printing, all articles begin on odd-numbered (right-hand) pages.

Readers should be aware that information in the authors’ biographical notes was current as of the original publication of the articles and may no longer be accurate.

The compilers and publishers welcome comments from instructors and students that will help in developing the 2004-2005 reader.

Laurel Wagers
Managing Editor
MultiLingual Computing & Technology
laurel@multilingual.com

INTRODUCTORY NOTE
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Author(s)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Cultural Dimension in Software Localisation</td>
<td>Reinhard Schäler</td>
<td>5</td>
</tr>
<tr>
<td>Impact of Translation Web Services</td>
<td>Jaap Van der Meer</td>
<td>9</td>
</tr>
<tr>
<td>Documentation Localisation Costs</td>
<td>Damian Scattergood</td>
<td>13</td>
</tr>
<tr>
<td>Comparing Tools Used in Software Localization</td>
<td>Thomas Wassmer</td>
<td>17</td>
</tr>
<tr>
<td>The Bible as a Resource for Translation Software</td>
<td>Jeff Allen</td>
<td>23</td>
</tr>
<tr>
<td>Localisation in South Asia</td>
<td>Pat Hall, Tapan Parikh, Shaileyl Minocha &amp; Venkatesh Hariharan</td>
<td>29</td>
</tr>
<tr>
<td>L10n and I18n: Dealing With the Difficulties</td>
<td>Barry Caplan</td>
<td>33</td>
</tr>
<tr>
<td>Pragmatics of English-Spanish Localisation</td>
<td>Montserrat Bermúdez Bausela</td>
<td>37</td>
</tr>
<tr>
<td>Chinese Script Simplification</td>
<td>Thomas Emerson</td>
<td>41</td>
</tr>
<tr>
<td>.NET Internationalization: RegionInfo and TextInfo Classes</td>
<td>Bill Hall</td>
<td>47</td>
</tr>
<tr>
<td>Creating Internationalization Quality Assurance Teams</td>
<td>Gabriella Spatolisano</td>
<td>51</td>
</tr>
<tr>
<td>Translation Is Not Enough</td>
<td>Svin Yeo</td>
<td>55</td>
</tr>
<tr>
<td>TMX Implementation in Major Translation Tools</td>
<td>Jost Zetzsche</td>
<td>59</td>
</tr>
<tr>
<td>An Introduction to Using XLIFF</td>
<td>Yves Savourel</td>
<td>63</td>
</tr>
<tr>
<td>The .NET Classes StringInfo and TextElementEnumerator</td>
<td>Bill Hall</td>
<td>71</td>
</tr>
<tr>
<td>Complying With European Language Requirements</td>
<td>Aidan Collins</td>
<td>77</td>
</tr>
</tbody>
</table>
# Table of Contents

The Use of XML in Localization  
Hans-Günther Höser  

Comparative Review of Four Localization Tools  
Thomas Waßmer  

Localisation Outsourcing: Where to Draw the Line?  
John Papaioannou  

Do TM Systems Compare Apples and Pears?  
Philippe Mercier  

A Little Paranoia Can Go a Long Way (I)  
Roman Civin  

Managing Translation for Communities in NYC  
Debbie Folaron & Igor Vesler  

New Organizations Serve the Localization Industry  
Nancy A. Locke  

Games Localization: Production and Testing  
Helen Trainor  

ELECT Online Offers Information  
Michael Bourke  

A Little Paranoia Can Go a Long Way (II)  
Roman Civin  

Can Productivity Gains Continue?  
Tex Texin  

Subscription information for Localisation Focus and MultiLingual Computing & Technology  


Ten reasons why localisation and cultural diversity are mutually exclusive.

1. **Localisation Is Top Down**

Many software developers in Silicon Valley still wish the rest of the world would be just like them: share the same values, speak the same language, have the same taste and preferences, use the same character set. Over the past 15 years, they could only be convinced to internationalise their code when managers — whose salary was closely linked to the sales figures for their international products — wrote this into their objectives. In fairness, they have been extremely successful. The overwhelming majority of publishers in the digital world now make more money from the sales of their localised products than they make from the sales of the original product. More than 60% of the overall revenues of Microsoft now originate from international sales. In 2001, and in Ireland alone, Microsoft had revenues of US$1.9 billion from its international sales.

Localisation provides access to the digital world “made in the USA” to non-English speakers. This is good news — but obviously a one-way street. The flow of products, of content, is in one direction only. The contribution to maintaining linguistic and cultural diversity is — if you can still call it a contribution — extremely limited.

2. **Localisation Is ‘Make-believe’**

Some localisers define localisation as the effort to “adapt products so that they look and feel as if they were developed in the country, the language and the culture that they are localised for”.

A trendy Web site in France will have a black background, while bright colours and a geometrical layout give a site a German feel. Dutch surfers are keen on video downloads, and Scandinavians seem to have a soft spot for images of nature, according Ben Vickers in the Wall Street Journal (26 March 2001), reporting on the EU-funded seed project Multilingual Digital Culture, or MUDICU (www.mudicu.org). Interestingly but hardly surprising — MUDICU is coordinated by Helene Abrand, an internet consultant working for Real Media France, the French subsidiary of the US company Real Media Inc.

Was it really Columbus who discovered America? Was the telephone invented by Alexander Graham Bell or by rival pioneer and inventor Marconi? What is the height of Mont Blanc — 4,808 meters as the French believe, 4,807 meters as the Dutch believe, or 4,810 meters as the Italian believe? What about such issues as:

- Date, time and number formats
- Colour schemes
- Pictures and images
- Hand signals
- Symbols
- Sounds
- Historical data
- Product names and acronyms

The result of the cultural adaptation effort is that a programme or a web page sends out all the right signals to you, the user — something like a chameleon that tricks its enemies. But you know, and most of all you feel, that something is not quite right — because no matter how much you change the colour of a web site, the hand signals, the symbols and the sounds used, the content will remain the same — and people in Finland will still not laugh at a joke from Texas.
3. LOCALISATION IS A US-driven Industry

With very few exceptions, the most notable one probably being the case of the German software developer SAP, most large-scale localisation projects are those where US material is localised for other, non-US markets. Around 90% of the overall global localisation effort is — and this is a conservative estimate — invested in the localisation of US-developed digital material. In other words, localisation is currently used almost exclusively by large US corporations as a vehicle to increase their profits. This is only logical and easy to understand. Localisation is relatively cheap compared to the development costs of the original product, yet it provides access to markets much larger than the original US domestic market.

Clearly, market expansion and higher revenue are the primary drivers for the localisation effort of the large IT publishers, not language, not culture, and certainly not diversity.

4. LOCALISATION WIDENS THE GAP BETWEEN RICH AND POOR

The argument has been made that access to the power of the internet and the web will narrow the gap between the have and the have-nots. It will lead to a better world, with more equality and prosperity for everybody. If that were true, localisation would support this development because it facilitates access and use of the web for non-English speakers, who otherwise could not participate in the Information Society. Sadly, the opposite is true.

In a world where half of the population has never made a telephone call — a world where only 3% out of 80% of the total population has a telephone connection — access to the internet is restricted to those who have the money to pay for it, to those who are already among the better off. While localisation could help to overcome the language and cultural barriers that can prevent access to the web, it cannot solve the fundamental financial and development barriers that prevent access for the vast majority of people.

The internet and the web are actually helping to widen the gap between rich and poor in our world. Localisation as we know it does not counteract this development; it supports it.

5. LOCALISATION FAVOURS ‘NORTHERN’ LANGUAGES

Software and web pages are not localised into languages that are spoken by the majority of people on our globe. They are mostly localised into what could be called the northern languages, those spoken by the affluent people of the northern hemisphere.

Today approximately 80% of web pages are in English, according to Julian Perkin in the Financial Times (7 February 2001). Most of the remaining web pages are available only in major northern languages like Japanese, German, French, Spanish or Portuguese.

This contrasts sharply with the 5.4% — little more than 1 in 20 — of the world’s population who speak English as their mother tongue (according to IDC, the Boston IT market researchers). Not surprisingly, the majority of these are Americans. A further 7% of the world’s population are proficient English speakers, so the language is a good means of communication for just around one eighth of the world’s inhabitants.

There are fractionally more native Spanish speakers in the world, and the Chinese languages dwarf English as a native language, spoken by 20.7% of the world population. Three-quarters of these, or 15% of the world population, speak Mandarin alone — nearly three times as many people as have English as a first language.
6. LOCALISATION CONTRIBUTES TO CREATION OF THE MCWORLD

In 1995, the political scientist Benjamin Barber first used the term McWorld to describe “the cosmopolitan, international, consumerist, multinationalised, advertising-based culture of cable TV, popular magazines, and Hollywood films”. This culture, according to Barber, aims at universal accessibility, in which billions watch the same World Cup finals. It is a culture in which MTV (translated), dramatisations of the lives of imaginary American millionaires, CNN and films like Titanic dominate and homogenise local cultures, producing a thin but powerful layer of consumerist, advertiser-driven, entertainment-based, and perhaps in the last analysis, American-influenced culture with great popular appeal, backed by enormous financial and technological resources.

He did not mention localisation explicitly, but, without a doubt, localisation facilitates access to this culture and facilitates the penetration of this culture into other local cultures, changing and adapting them in the process. In a “simple twist of fate”, almost magically, localisation is no longer just a process of linguistic and cultural adaptation but becomes itself an active player that adapts and changes (localises) other languages and cultures.

There is no doubt that the plethora of Americanisms, especially in the terminology dealing with IT in many “northern” languages has been introduced to these languages through localised products.

Similar patterns can be observed in consumer behaviour, business attitudes and social change. The American concept of “stock holder value” runs directly against the traditional attitude of social responsibility that has characterised the way large German corporations run their businesses. Yet it took less than a decade during the eBoom emerging from California’s Silicon Valley towards the end of the 1990s to force German businesses to dramatically change the way they run their businesses.

7. LOWEST COMMON DENOMINATOR AND GLOBAL SYMBOLISM

Whereas culture needs diversity and thrives on difference, localisation, internationalisation, and globalisation often aim for the lowest common denominator and a global symbolism. While diversity should be interesting and stimulating, one of the golden rules of localisation is that of reusability. Reuse as much material and change as little material as possible.

Designers of global products, of websites aimed at the global market, use globally acceptable standards, symbols, and conventions. They refer to what they believe are globally applicable appeals to needs and desires, and they create global icons, references and stereotypes.

These stereotypes strongly influence the design, development and presentation of web content. By contrast, culture is about the spice of life, culture is about difference, culture is about contrasts. A prevailing lowest common denominator of tastes, preferences and opinions, nurtured by the global web, marks the end of diversity and contrast and, ultimately, the end of culture.

8. GLOBAL ICONS SUPPRESS LOCAL CONTENT & CREATE ‘ACCIDENTAL WEB’

Because of the power, persuasiveness and the resources behind them, global icons, global symbols and global content are pushing aside and suppressing local content. When you travel to Spain, do you want to find out from a web-based, localised US travel guide where to eat in Santiago, Madrid or Seville? Or when you travel to the Middle East, do you want to read up on the history of the region on a localised US web page? Regrettably, this is what you will most likely be offered when searching the web for this kind of information.

If one searches for background information on any region of the world, the likelihood is that the information presented will not originate from the region but from US, localised web sites that are registered with the major search engines and shown at the top of the list of search results.

Like travel writer Macon Leary in Anne Tyler’s book The Accidental Tourist, who hates both travel and anything out of the ordinary, many global eContent publishers dislike diversity and divergence from standards — for the simple reason that it makes their lives more difficult and (even more crucially) their projects more expensive. They create perfect “accidental” Web sites that are acceptable to every global citizen’s tastes, beliefs and customs. There are no surprises, no deviation from the norm. There is an almost clinical feel of global political over-correctness to them.

Macon Leary needed Muriel, a wacky dog-obedience trainer, to end his insular world and thrust him headlong into a remarkable engagement with life. Local content producers, local cultures need the technical experts to bring their content to the world so that the world can enjoy the different perspectives and approaches offered by them.

9. CONTENT MUST BE LOCAL — LOCALISATION MUST BE BOTTOM UP

To preserve diversity, to preserve culture, digital content must be local and must be presented and accessible in as many languages as possible. In Europe, for example, there is no shortage of digital content. In fact, many market researchers have highlighted Europe’s advantage over many other regions in the world in this area. However, this content must be made easily available to the rest of the digital world. It must be made available to people living in different countries, speaking different languages.

Imagine how interesting and entertaining it would be to be able to listen to the Spanish and the English commentaries of a soccer match between Barcelona and Manchester United. And there is no reason why the sources of information on which we base our interpretation of contemporary events should exclusively be supplied by and filtered through the large news networks like Sky and CNN. A recent example of the important role, and indeed the success, of a relatively unknown digital content provider (in this case a local satellite news channel) is the Al Jazeera satellite news channel, which is supplying up-to-date and often exclusive reports and pictures from the Arabic world (see www.aljazeera.net).
There are reasons why major European television stations regularly report on torrential rains and severe flooding in the Mississippi Delta causing great damage to local houses and sweeping away half a dozen cars, while minimum coverage is given to human catastrophes, death, hunger and disease in Africa and Asia.

These reasons are directly linked to availability and cost.

Diverse, locally produced, eContents can only be offered if this content finds its way into the digital world and is then localised to make it accessible to speakers of other languages. It is “bottom-up localisation” (and not the top-down localisation that we have become so used to) that will contribute to preserve the diversity of languages and cultures.

**10. WHY LOCALISATION EFFORTS MUST FUNDAMENTALLY CHANGE**

Most of the reasons I’ve covered so far why localisation efforts must fundamentally change and be redirected concerned mainly political and cultural areas. As I said earlier, in the world of big IT business, a world dominated by young, hard-working, no-nonsense professionals whose decisions are mostly determined by stock market analysts’ reports, it is difficult — if not impossible — to interest anybody in diversity, never mind culture. So let’s talk about revenues, new markets and cost models.

It has been established that most large digital publishers make most of their revenue from their international sales. These publishers would never have got where they are now, they would never have achieved their current revenue figures, they would never have been able to achieve that level of market penetration — without their highly successful localisation strategies.

The success stories of companies like Microsoft, Lotus/IBM, Oracle and Symantec, which by now have long moved from being simple software developers to becoming electronic publishing houses, should encourage publishers in other regions in the world.

Europe has the knowledge, the experience and the expertise to localise its vast amounts of electronic content and make it available to the rest of the world, creating new business opportunities and markets.

What is needed is the political foresight and the industrial initiative to make this happen.

**Conclusions**

I hope to have shown that localisation certainly has the potential to preserve linguistic and cultural diversity. To fulfil this potential, however, its efforts have to be redirected. Initiatives and actions such as those undertaken within the European Union eContent Programme (www.hltcentral.org and www.electonline.org) as well as the sharing of the lessons learnt during the course of successful and unsuccessful localisation projects (especially those for languages and cultures ignored by mainstream localisation efforts) could, in my opinion, help to use the large pool of expertise and experience available within the localisation community to preserve cultural diversity and to prevent a slow but certain demise of cultures currently not present in the digital world.

Earlier on, I quoted David Brooks, who said any language that is not captured in this electronic world will soon become obsolete. I believe that any culture not captured in this electronic world will eventually become obsolete.

In addition, dominance and exclusivity cannot ultimately benefit anybody, not even to the big players, because no culture can live if it attempts to be exclusive (Ghandi).

I believe we should not remain at the level of mere recipients of the localised content of others. We should become actors, localise our own content and make it accessible to everybody in the digital world. We should all become localisers.

---

**References**


Reinhard Schäler is director of the Localisation Research Institute in Limerick, Ireland. He can be reached at reinhard@ul.ie
In November of 2001 Bill Looby of IBM gave a presentation at the LRC conference in Limerick about web services and how this technology could apply to the translation market. A lot of talk and some real developments have happened since then. Teelingua announced its T-Remote Memory product, which uses a web services definition to retrieve translation matches from multiple distributed translation memory systems. Berlitz uses a web services definition in its Open Translation Connector to link content management systems with BerlitzIT. Lexelnet is about to launch its translation marketplace using web services definitions for the providers of the services. Globalsight announced its .Net edition, which allows localisation portals to link easily with its globalisation technology. Those who have not yet implemented web services one way or the other in their translation technology will most certainly be talking about it.

Web services create the best opportunity for the cottage translation market to cross the chasm and become a professional services industry. Web services help us to overcome the dilemma that has suppressed the translation market ever since we started talking about localisation as an industry (around 1990). The mergers and acquisitions of recent years and the launch of new market offerings have only reinforced this dilemma. The supply chain has become more complex, with more vendors and more tools — and more incompatibility. Any possible gain from the use of translation memory is lost again because of the increase in overheads of managing multiple translation databases. Web services connect tools and suppliers automatically without the need for direct communication. Articles or sections from websites will find their way to the translator automatically. No phone conversations, no faxes, no FTP, no emails (except perhaps automatically generated emails). Translation memory matches will be retrieved fully automatically from anywhere in the network. And for real-time translation, the web services architecture will automatically route texts to a machine translation engine. No pre- and post-processing, no translation folders and no so-called translation engineering. Web services allow us to build the intelligence into the infrastructure in standard definitions that communicate openly with any tool and any supplier that comply with the standard.

Web services will undermine current business models, especially those of the so-called MLVs. Ironically speaking, it was in their interest to keep the dilemma alive. If the infrastructure were to become intelligent enough to automatically deliver a quote or final approved translation, the role of the translation brokerage firm would become less relevant (unless perhaps they build and rent the web services infrastructure themselves). In this article, I will describe the role of web services from the perspective of the dynamic market we live in.

**Market Relevance**

As information becomes a corporate asset for many companies, translation and effective communication in all required business languages also becomes vitally important. While translation in many companies is still a relatively isolated activity, more and more corporate decision-makers now understand that it must be integrated into corporate processes. Translation and effective multilingual communication become part of corporate information strategies. They play a crucial role in managing customer relationships, human resources and supply chain. Market surveys indicate that people stay four times longer on sites when the information is provided in their native language. The extent to which intranets and extranets are translated has a direct effect on the success of companies.

However, the current market offering does not provide for adequate integration in corporate purchasing platforms, in content management or customer relationship management. The translation market is still a cottage industry. Services are offered by hundreds of thousands of freelancers working from...
their homes and thousands of agencies. They all have their specialties, like subject knowledge and language pairs. And all have their own terms and peculiarities when it comes to delivery and invoicing.

**Market Size**

According to IDC, the size of the localisation, translation and interpretation (LTI) services market and cross-lingual application (CLA) services was $2.981 billion in 2001, predicted to grow to $6 billion by 2006. (IDC, Worldwide Globalization, Internationalization, and Localization Services Market, Forecast and Analysis Update, 2001-2006). The diagram at right provides a breakdown of the major sub-industries, their 2001 revenues and their projected growth up to 2006.

Website localisation is a specialised service that has emerged in recent years (since 1999). It is basically a packaging of translation services with technical services that ensure the proper functioning of the translated sites. A number of specialised service companies have entered this marketplace.

Computer-aided translation as a sub-sector consists of the sales of software and services that support the automation of translation processes, such as translation memory software, machine translation and dictionary management software. Although these software products have been around for quite a long time (since 1985), the market thus far has developed slowly. IDC expects this sub-sector to grow by 35% a year. This clearly illustrates the corporate need for process automation and integration.

Software localisation is a specialised service that emerged with the proliferation of desktop computers at the end of the 1980s. Since that time a new generation of specialised service companies has emerged offering packaged translation services and technical services (like software testing and quality assurance) that ensure localised software will function properly.

Translation is at the forefront of the LTI services market. It is the basic service provided by hundreds of thousands of freelancers and thousands of agencies. In fact, most of the software localisation and website localisation companies use the services of the translation sub-sector on a subcontractual basis rather than using in-house translators. The market is known for its multiple layers of subcontracting. The management overheads of these multiple transactions represent a considerable share of the size of the translation market. While a freelance translator may in a particular case receive $0.08 for each word he translates, the ultimate customer may be paying $0.24 — three times more.

To adequately represent the focus of translation vendor web services, it is appropriate to split the IDC category of the translation sector into an actual translation service part and an overheads of transaction management part. IDC projects an average annual growth for the translation sub-sector of 6.5%. With the emergence of web services standards, it is likely that overhead expenses will even out over the next couple of years. Workflow automation, e-procurement, supply chain automation and the arrival of new marketplaces will help to bring transaction costs down.

We assume that at our base point revenues of the translation sector are equally divided between actual translation service costs and overheads. By 2004 we will already see the result of transaction automation and a reduction of overheads. The proliferation of translation web services standards will help to bring transaction costs further down over time.

**Cascaded Supply Chain**

The translation market has grown into a multilayer industry with many interdependencies between the various players. First, there were the individual professionals or freelance translators. They are listed in the registers of national translator associations, but they receive their work usually through personal contacts and yellow pages. Since the 1950s and 1960s many agencies have been established that broker the services of freelance translators. They receive their work through yellow page advertising and now more and more through the internet. Since the 1980s specialised software localisation companies have been established that select suitable translation resources from the freelance and agency supply base. Since the 1990s a wave of consolidation started at the top of the translation market.

<table>
<thead>
<tr>
<th>ACTOR</th>
<th>TASKS PERFORMED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Procurement, Vendor Management, Quality Assurance,</td>
</tr>
<tr>
<td></td>
<td>Translation Memory Management, Project Management</td>
</tr>
<tr>
<td>MLV</td>
<td>Procurement, Vendor Management, Quality Assurance,</td>
</tr>
<tr>
<td></td>
<td>Translation Memory Management, Project Management</td>
</tr>
<tr>
<td>SLV</td>
<td>Procurement, Vendor Management, Quality Assurance,</td>
</tr>
<tr>
<td></td>
<td>Translation Memory Management, Project Management</td>
</tr>
<tr>
<td>Agency</td>
<td>Procurement, Vendor Management, Quality Assurance,</td>
</tr>
<tr>
<td></td>
<td>Translation Memory Management, Project Management</td>
</tr>
<tr>
<td>Translator</td>
<td>Translation, Translation Memory Management, Quality</td>
</tr>
<tr>
<td></td>
<td>Assurance, Project Management</td>
</tr>
</tbody>
</table>
and a handful of global multilingual vendors have emerged. They select and buy the translation services from agencies and from the smaller specialised localisation vendors.

A cascaded supply chain has evolved that is made up of many actors who often play similar roles in the production of the services. At every new link of the supply chain, project management tasks, quality assurance, procurement, file handling and other tasks will be performed. But as no industry standard process definition exists, it is inevitable that a tremendous amount of overlap is hidden in these trails of subcontracting and outsourcing.

The table below illustrates a typical outsourcing model and the tasks being performed on every level.

Although this supply chain seems to be an obvious candidate for definition and automation, very little has been achieved until now on this level in the globalisation market. Several of the largest suppliers in the market have developed workflow automation products as well as supply chain and customer portals for online project monitoring and collaboration. However, these packaged products have not found very favourable customer response because customers in general fear to be locked into a proprietary service environment.

**Customers’ Buying Criteria**

Customers of multilingual communication services require a rare combination of the highest level of language quality and topic specialisation with the lowest level of overheads and the fastest turnaround. They want to be able to change their specialist providers if better quality, better turnaround or lower cost can be obtained somewhere else. Therefore, they do not want to be locked into a one-to-one marketplace or portal from one of their service providers.

The customers’ buying criteria can be listed as follows:

- Quality of devoted specialist translators;
- Non-proprietary portal for procurement, vendor management, project management and monitoring, collaboration, quality assurance;
- Lower cost;
- Faster turnaround;

The portal or functionality must be integrated into other enterprise applications, like a procurement platform, intranet, extranet to allow direct access to all authorised users within an enterprise;

The service must offer interoperability for all translation memory, machine translation, content management and other enterprise applications.

To meet all of these customers’ buying criteria the globalisation market requires a fresh approach, which comes down to a separation of the delivery of the high quality services by the devoted professionals on the one hand and the facility of a highly automated and standardised infrastructure for transaction management on the other hand.

**MARKET PLAYERS**

The localisation, translation and interpretation (LTI) and cross-lingual application (CLA) services market is populated by multiple layers of service providers and only a few pure technology providers. The table at top right provides an overview of the service providers in categories of size from large to small.

A few service providers have entered technology development, especially at the top of the supply chain, but most have withdrawn from it. Customers would not buy their technology for fear of being locked into one service provider’s technology.

The diagram above illustrates the rapidly growing importance of cross-lingual applications services. The percentage of cross-lingual applications services of the total size of the localisation, translation and interpretation services market is rising from 1.5% in 2001 to 2% in 2004 and to 3.3% in 2006.

The human translation segment is growing at only 6.5% annually, thus indicating the acute need for leveraging software in the form of transaction management and translation technology. The industry’s growth is currently suppressed by the complexity and cost of managing the translation supplier base. This reality is stimulating the growth of translation automation and transaction and workflow automation.
**Hybrid Solution**

Web services potentially have a much greater impact on the translation market than any other development before, including the consolidation and creation of large MLVs since 1995. The route to scalability and efficiency does not seem to go through mergers and acquisitions as it does perhaps in many other industries. What the translation market needs is a hybrid solution, consisting of an open infrastructure on the one hand and a multitude of highly specialised human resources on the other hand. The best quality can be achieved by those who are nearest to the work being produced: the translators, the reviewers and the writers. The most efficient service management system is an automated open industry infrastructure.

Web services have the potential to create such an intelligent and open infrastructure that automatically links with high-quality resources and interoperable tools and databases.

As some people say: “This industry goes around in cycles. …”

Jaap van der Meer is a partner in Cross Language n.v., a consultancy and system integration company dedicated to translation and language technologies. He can be reached at jaap.vandermeer@crosslang.com
Cost control in the current economic climate is critically important to maintaining an effective global presence for your products and services while remaining solvent. In this article, localisation experts KPS outline 10 steps to effective cost control to be considered.

Working with a number of software publishers, KPS has seen a typical internal/external cost split of 60/40, and the external 40 is in turn split by the translation vendor 30/70, where 30 is the management and engineering cost and 70 is the cost of translating the words. So out of each $1 only 28 cents is spent on translating the words! Remember that the margins for straightforward localisation tasks based on a price per word (of culturally adapting content from one language into another using the correct terminology and style) is low (5-10%), so seeking to negotiate lower prices with localisation companies is not feasible without an unacceptable loss in quality. Instead, the trend for these companies is to look for revenue from per hour activities (DTP, engineering) and other services (such as globalisation assessments and project management). In addition, your company may be under internal pressure to reduce head count or to operate under a head count freeze.

You should seek to address lower costs of localisation and control head count while increasing or maintaining quality and decreasing time to market. An underlying tenet of these solutions is the use of automation, appropriately deployed with common sense. (Note: For the sake of simplicity, we assume that the source language is English. However, these techniques can be adopted for any source-target language(s) combination.)

One: Educate your colleagues to treat localisation as a business process like any other activity critical to the bottom line of your company. Any file can be translated, but not by a standard process. Start with basic acceptance criteria for localisation. 1) Is the product internationalised, 2) can the source files or externalised strings be safely and easily translated by a commercial and easily available process and 3) can the localised strings and sizing coordinates be recycled and reused as the source files change between updates. No one-off throwaway localisations please!

Two: Volume — typically documentation projects represent the biggest localisation expenditure. Whether it is online...
help or documentation, the per word cost of localising these, combined with any related engineering costs can represent 80% of your budget, providing a serious barrier to entry. Start by promoting the idea of asking two related questions: 1) should there be documentation written about this? 2) Should it be localised? The answers to these questions are not the same, and it’s wrong, perhaps even expensive, to believe that everything written in English must be localised.

Once you are satisfied that documentation should be written about a product (for example, when the product’s user interface cannot be made more intuitive), adopt a selective translation approach. Structure your documentation’s information into discrete content baskets based on a small number of audience types. For example, three categories of user could be adopted: installer-implementer; advanced super user; and self-service user.

- The first type may not need to be translated since it deals with installation, implementation, consulting and development information that is best left in English, dealing with APIs, code samples, development tools, techniques and so on.
- The second type could deal with infrequently performed tasks such as changing business parameters. It has a smaller audience.
- The third is typical task information — for example, “how-to” procedures. It has a wide audience. This needs to be translated if the product UI is not self-explanatory.

Using XML facilitates this structure and management decision. Define your data using a DTD (Document Type Definition) that allows audience attributes to be applied to elements, and then use XSLT to transform the selective elements to a localisable file format like XLIFF. Since this data definition is not only logical but flexible, it can be varied according to market requirements and extended as market conditions vary.

<table>
<thead>
<tr>
<th>Audience</th>
<th>Localised</th>
<th>% of total volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installer/ implementer</td>
<td>Never</td>
<td>30%</td>
</tr>
<tr>
<td>Advanced super user</td>
<td>Sometimes/ as market requires</td>
<td>30%</td>
</tr>
<tr>
<td>Self-service users</td>
<td>Yes</td>
<td>50%</td>
</tr>
</tbody>
</table>

**Three:** Sorting — alphabetically sorted files are painfully costly. This is because other languages contain different characters and numbers of characters from English. Once translated, the files need to be resorted, extra characters added, redundant ones deleted and so on.

Have you ever tried sorting HTML alphabetically? It’s a costly and time-consuming exercise! Instead, externalise sortable content into a database and use the NLS collating sequence to sort the file dynamically, or use XML and XSLT to do the same.

**Four:** Reuse of previously translated content is critical. Develop solutions that not only allow fast reuse but also secure that reuse by maintaining context. Place identifiers on data at a level of granularity that is appropriate, for example on paragraphs.

```
<ID_ABC123455-->To save the file, press CTL+S
<ID_ABC123456-->To run the printing process, press CTL+P
```

**Five:** Grammar check your work before translation. This sounds obvious, but it isn’t uncommon that content is sent for localisation before it has been spell checked. Using a simple technique like running the grammar checker in your word processor may not only catch spelling errors but also detect problematic sentence constructions: for example, the use of the passive voice which can only otherwise be detected by using expensive controlled authoring techniques. Even such a simple step means that some editorial changes can be eliminated before translation starts, and the English content is better too.

**Six:** Graphics and images files should be separated from text to allow the text to be added to the translation memory. Graphics work is expensive. Treat callouts, captions and explanatory text sensibly. Use numbered callouts with corresponding numbers on graphics and add the text into the main document. By separating the text from the graphic, the same graphics file should be reusable in all language versions without the need to change it. Only the language of the text will change.

**Seven:** If you have accessibility requirements, handle this sensibly. Rewriting HTML documentation is not required. For tables, you can add summary captions that say “this table is for formatting only” and for graphics the ALT tag can say, “this picture is described in the text that follows”. These simple phrases can be easily localised once and then automatically inserted into the source code without fear of loss of context or the need to write and localise expensive descriptions. You can tender product or service for government and US or European Union contracts and meet statutory accessibility requirements using these simple techniques.

```
<IMG SRC="genericgraphic.gif" ALT="This image is described in the text that follows">
```
Eight: Use open standards so that your files can be easily translated by any tool or localisation vendor. XLIFF is an excellent medium for this purpose. It also allows easy customisation and the need to develop proprietary filters.

Nine: Redesign your localisation process so that your automation is central to reusability, internal checking and validation. Use the power of technology to do as much work as possible internally by using existing resources or fewer staff. Once a file leaves your organisation and is sent to localisation for any task, you will have to pay heavily even for tasks that those companies have themselves automated (such as updating or leveraging files). Retain non-language critical tasks such as project management, leveraging and quality checking inside your own organisation, and use solutions designed to perform these tasks safely and automatically, 24 x 7, with minimum manual intervention.

Ten: Terminology — sign off on the terms that will be used in your translation up front before the actual localisation progresses too far. Most terminology change requests are based on a matter of taste, usually by people who don’t have to pay for the changes. Take control. Extract key terms from the product or service, localise them first and then have your customer sign off that they are acceptable. Do a market test with the terms on sample customers. Most major companies invest heavily in terminology. Even if translation isn’t 100%, the documentation is still usable and the UI perfectly navigable. Get a proper perspective on terminology. Most documentation is read only once. In the days when there were spelling errors on the front page of the Guardian, no one complained that they didn’t get the message. String extraction techniques can improve localisation quality, improving consistency and usability, giving you that competitive edge.

For more information, email info@keyperformancesolutions.com KPS was founded in January 2002 with a vision to be the leading expert of software globalisation and process knowledge worldwide. KPS provides consultancy services to the G11N, I18N, L10N and eContent markets.
The localization of software often involves a variety of specialists, such as programmers, translators, localization engineers, quality assurance specialists and project managers. Therefore, not only a smooth handling of each task but also the co-ordination of teamwork is required. Without specialized tools, this is a very time-consuming process, and quite often the same task is uselessly repeated over and over again. Fortunately, specialized software tools effectively assist freelance translators as well as whole in-house localization units in this multilevel process. My comparative review examines three specialized software localization applications and two more general translation packages, which also cover the internationalization of software.

Specialized localization applications are tailored to deal with resource code files (RC) or binary files such as EXE or DLL, which usually contain only short translatable text strings surrounded by non-translatable code. They, therefore, have to extract these short strings properly, provide a convenient graphical user interface (GUI) for the translation and save the translations correctly back into the unaltered surrounding code. Special attention has to be paid to controls embedded in the translatable text, such as ampersands (&) and shortcuts (Alt+D), which have to be translated to be both unique and easily memorable.

Another requirement of software localization is that translated strings should be about the same size as the original text. This is because the translated text needs to fit into the appropriate spaces of menus and dialog windows. If a size-equivalent translation is not possible, resizing procedures are needed. As only short text strings are translated, there is no need for translation memories (TM). A bilingual or multilingual terminology base or glossary is sufficient.

In contrast, software documentation (Windows Help — HLP, HTML Help — CHM, Web pages — HTML or Adobe Acrobat PDF) contains much more translatable text in much longer strings. These files are usually better handled by TM software, which memorizes already used phrases that are typically segmented by full-stops and enables their recycling. Most TM software includes some fuzzy match algorithm to identify the degree of concordance between new and already translated segments and allows the user to insert and edit such strings within an authoring environment.

The tools described in this article are Alchemy CATALYST 4.02, PASSOLO 3.5, RC-WinTrans 6.01, STAR Transit XV and TRADOS 5.5.
Alchemy CATALYST 4.02

The Irish company Alchemy Software Development Limited develops the successor to Corel CATALYST. In version 4.02, the software supports all Windows programs (DLL, EXE, INI), RC and Help files (RTF, CNT, CPP, HPP), as well as XML/XHTML-based files and Internet applications, including full Microsoft .NET support. All supported file types are handled competently and conveniently besides Windows Help files. Here you do better turning to a real TM solution. CATALYST’s XML/XHTML features proved to be most impressive: three specialized editors allow the user to translate only elements requiring translation while avoiding all formatting tags, code and script segments.

Alchemy CATALYST is by far the most complete software localization tool within this review. It is the only specialized software localization tool on the market that comes close to the number of features usually only seen in complex TM or computer-assisted translation (CAT) applications.

Editions and prices: CATALYST QuickShip Environment is a nice gift to freelance translators working for companies that use CATALYST. Free of charge.

One’s own projects can be established in the Translator Edition ($599), but leveraging of previous translations is not possible.

The Localizer Edition ($3499) includes additional Leverage and Runtime Validation Experts.

With the Developer/Pro Edition ($5999), translation projects can be deployed throughout the enterprise or between translation partners.

PASSOLO 3.5

PASSOLO was founded as a medical analysis system provider in 1990. After first being used internally, PASSOLO became available as an independent localization tool in 1998. PASSOLO supports applications (EXE, DLL or OCX) programmed in Visual C++, Visual Basic, Borland Delphi and Borland C++Builder as well as Pocket Windows. Using free downloadable macros, the tool adds support for Java.PROPERTIES and text files such as Windows INI files. In order to avoid translation artifacts, PASSOLO intentionally lacks support of RC. Several glossaries can be used simultaneously in a fuzzy-match display of all concordances. Pseudo Translation identifies programming errors. The Professional Edition allows for data exchange between PASSOLO and other tools via Add-Ins, currently available for the translation memories TRADOS and STAR Transit. Thanks to the integrated Sax Basic Engine, the creation and editing of macros are an easy task even for non-programmers. Due to an internal handling avoiding the limitations of the Windows 9x API, PASSOLO can save back localized EXE files under Windows 95, 98 and Me.

PASSOLO is available in four editions. The Translator Edition is a free editor for the translation of bundles created with the Team Edition. Standard Edition ($606) is the basic standalone tool. The Professional Edition ($953) is extendable and customizable by means of Add-Ins, OLE automation and VBA compatible development environment.

The Team Edition, which allows creation of translation bundles, starts at $2241.

STAR and TRADOS Add-Ins are available at $650 each.

RC-WinTrans 6.01

Schaudin.com was founded in 1993 as a software design company in the Frankfurt Rhein/Main high-tech area in Germany. The company has focused on translation software for the localization of MS Windows software since 1995. Schaudin.com’s flagship RC-WinTrans is a powerful and well-organized tool for the localization of RC files, executables (EXE), link libraries (DLL and OCX) as well as Java.PROPERTIES files, simple C/C++ text definition files (H), Windows INI files and InstallShield stringtable files (SHL). As the name indicates, RC-WinTrans’ specialty lies within the translation of RC files for which it offers the most powerful parser in this comparative test. Another specialty of RC-WinTrans is the ability to extract text components from ActiveX controls. RC-WinTrans does not support Help files and Web formats. It can afford to lack its own translation memory. By providing a thoroughly developed interactivity with the TRADOS Translator’s Workbench, however, extended features of the TM such as concordance search can be used within RC-WinTrans. RC-WinTrans is the only product in this comparative review which does not use a hardware key (dongle) for licensing protection. As dongles often cre-
Inconveniences for the user, this is another minor advantage of RC-WinTrans.

The Standard Edition is priced at $840. An External Translators’ Group License is $2410. The RC-WinTrans Lite Edition, with limitations in the maximum size of a translation project, is mainly thought of as a free trial version.

**STAR Group Transit XV**

STAR AG was founded as a small translation agency in 1984 in the northern Swiss city of Stein am Rhein near Schaffhausen. STAR’s TM, Transit, is not really a TM in the classical sense. Instead of a central database, it uses a system of reference files that allows users to base each translation on a number of old translations selected for the actual purpose. Transit includes standard filters for Text, Corel WordPerfect, WordPro and AmiPro, Word documents (DOC and RTF), WinHelp files (RTF), RC, Microsoft Excel and PowerPoint, as well as tagged text file formats, including XML, SGML, HTML and QuarkXPress. Optional filters increase this spectrum for DTP file formats such as Adobe FrameMaker and PageMaker, Interleaf and XGate and C/C++, Java/VB or other source code. All input formats are translated within one internal editor. In addition, the same GUI is also used for quality assurance, statistics, manipulation of the reference files and the alignment of source and target files of previous translations. STAR’s terminology solution TermStar can be limited to a single workspace or to almost all common database servers and can manage large multilingual terminology bases of several million elements. STAR Transit is well equipped to guide you through the localization of resource script files and various types of software documentation such as WinHelp and HTML/XML files. It fails to be a general software localization tool, however, because it lacks support for executables and link libraries (EXE, DLL and OCX).

Some editions are Transit XV Enterprise ($4030), which is network enabled; Transit XV Professional ($1430), for a single workspace; and Transit XV Smart ($713), which is feature limited for freelance translators. Transit Satellite Personal Edition is available free.

TermStar versions include TermStar XV Enterprise ($892), which is network enabled; and Term Star XV Professional ($623).

**TRADOS 5.5**

TRADOS was founded in 1984 in Stuttgart, Germany. In 1990, the first version of MultiTerm, one of TRADOS’ main components, was created. TRADOS consists of several seamlessly integrated editors and tools supporting Text, Word documents (DOC and RTF), WinHelp files (RTF), RC, Microsoft PowerPoint presentations, as well as tagged text file formats, including XML, SGML, HTML, FrameMaker, Interleaf, QuickSilver, QuarkXPress, PageMaker and Ventura. Through the T-Window for Executables, it is now also possible to translate EXE, dynamic link libraries (DLL) and ActiveX controls (OCX).

The TRADOS 5 system contains the new central interface WorkSpace, which serves as a launch pad for TRADOS components and includes project management, distribution and reintegration of packages. Likewise new, XTranslate enables project managers to transfer approved translations automatically from previous versions of bilingual documents into new versions of source documents. TRADOS’ TM, the Translator’s Workbench, can be fully integrated into Microsoft Word showing a fuzzy-match functionality with color-coded highlighting. In addition, it can be run cross-linked to the terminology database MultiTerm.

A third new feature of TRADOS 5 is the terminology extraction application ExtraTerm, which automatically extracts a list of candidate terms and their probable translations from bilingual or monolingual documents. The visual alignment system WinAlign allows users to create a translation memory from previous projects. TRADOS TagEditor enables you to directly open Web and SGML or DTP formats. These files can be translated in conjunction with the Translator’s Workbench as easy as untagged file formats. All tags are protected against accidental editing or deletion. Finally, the new T-Window Collection supports Excel spreadsheets and any text from the Clipboard.
### A Comparison of Software Localization Tools

<table>
<thead>
<tr>
<th>Developer/Software</th>
<th>TRADOS Corporate 5</th>
<th>STAR Trans XV</th>
<th>Alchemy CATALYST 4</th>
<th>PASSOLO 3.5</th>
<th>schaudin.com RC/WinTrans 6.01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Translation Memory</td>
<td>Translation Memory</td>
<td>Software Localizer</td>
<td>Software Localizer</td>
<td>Software Localizer</td>
</tr>
<tr>
<td><strong>Translator Assistance Tools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spell Checker</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes via macro</td>
</tr>
<tr>
<td>Thesaurus</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Glossary/Dictionaries</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Terminology Database</td>
<td>Yes</td>
<td>No</td>
<td>No via TRADOS</td>
<td>via TRADOS</td>
<td></td>
</tr>
<tr>
<td>MT Interface</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Term extraction</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Context-sensitive translation</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Supported File Types</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLL, EXE</td>
<td>Yes/T-Window</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes via TRADOS</td>
</tr>
<tr>
<td>RC</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTF (Win 95 Help)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RTF (Text)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TXT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DOC</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FrameMaker and SGML</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Ventura</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>HTML, XML</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Software Leveraging</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fuzzy Matching</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes via TRADOS</td>
</tr>
<tr>
<td>Concordance</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes via TRADOS</td>
</tr>
<tr>
<td>Perfect Matching</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes via TRADOS</td>
</tr>
<tr>
<td>Paragraph based</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Help/Document Leveraging</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fuzzy Matching</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Concordance</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes via TRADOS</td>
</tr>
<tr>
<td>Perfect Matching</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes via TRADOS</td>
</tr>
<tr>
<td>Paragraph based</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Software Engineering and Testing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pseudo Translate Expert</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Validate Expert</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>General Variables</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protection Mode</td>
<td>Dongle</td>
<td>Dongle</td>
<td>Dongle</td>
<td>Dongle</td>
<td>License key</td>
</tr>
<tr>
<td>TMX Import</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Simultaneous database access over LAN</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Price in euros (most featured license)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Price in US dollars (most featured license)</td>
<td>$4,886</td>
<td>$1,430</td>
<td>$5,999</td>
<td>$5,450</td>
<td>$2288</td>
</tr>
<tr>
<td>System requirements</td>
<td>Pentium II/64 MB</td>
<td>Pentium III/64 MB</td>
<td>686X2/16 MB</td>
<td>486X1/16 MB</td>
<td>400 MHz x86</td>
</tr>
<tr>
<td>Editor</td>
<td>external/external</td>
<td>internal</td>
<td>internal</td>
<td>internal</td>
<td>internal</td>
</tr>
<tr>
<td>Source and target in tables</td>
<td>No</td>
<td>Optional</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes via TRADOS</td>
</tr>
<tr>
<td>Display of concordance</td>
<td>Yes</td>
<td>Yes</td>
<td>Choice</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Editable Metatags</td>
<td>Yes</td>
<td>Yes</td>
<td>Choice</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Viewable Metatags</td>
<td>Yes</td>
<td>Yes</td>
<td>Choice</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Target Preview</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes via TRADOS</td>
</tr>
<tr>
<td>Technical support</td>
<td>72 hr/competent</td>
<td>48 hr/not helpful</td>
<td>48 hr/competent</td>
<td>72 hr/competent</td>
<td>48 hr/competent</td>
</tr>
<tr>
<td>Support Cost</td>
<td>Expensive</td>
<td>Expensive</td>
<td>Expensive</td>
<td>Expensive</td>
<td>Expensive</td>
</tr>
<tr>
<td>Printed documentation</td>
<td>Good/tutorial</td>
<td>Good/tutorial</td>
<td>Unsatisfactory</td>
<td>Very brief</td>
<td>Very brief</td>
</tr>
<tr>
<td>Help/PDF documentation</td>
<td>Good/tutorial</td>
<td>Good/tutorial</td>
<td>Unsatisfactory</td>
<td>Very brief</td>
<td>Very brief</td>
</tr>
<tr>
<td>Software classes available to handle</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes via TRADOS</td>
</tr>
<tr>
<td>Package includes</td>
<td>CD-ROM, printed manuals, dongle</td>
<td>CD-ROM, printed manuals, dongle</td>
<td>CD-ROM, printed manuals, dongle</td>
<td>CD-ROM, printed manuals, dongle</td>
<td>CD-ROM, printed manuals, dongle</td>
</tr>
<tr>
<td>Software ergonomics</td>
<td>Handling large TMX database</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Time to learn</td>
<td>Easy to learn</td>
<td>Easy to learn</td>
<td>Complicated</td>
<td>Easy to learn</td>
<td>Easy to learn</td>
</tr>
<tr>
<td>Linking external dictionaries</td>
<td>Only MultiTerm</td>
<td>Only TermStar</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Direct translate from Excel</td>
<td>Yes/T-Window</td>
<td>Yes/T-Window</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Direct translate from PowerPoint</td>
<td>Yes/T-Window</td>
<td>Yes/T-Window</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Link to MT</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Network Version</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TM management</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Alignment/Word types</td>
<td>RTF, DOC, HTML, XML</td>
<td>All supported file types</td>
<td>RTF, XML, DLL, EXE/RC</td>
<td>MEDIUM</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>Time to establish new project</td>
<td>Fast</td>
<td>Fast</td>
<td>Fast</td>
<td>Fast</td>
<td>Fast</td>
</tr>
<tr>
<td>Integration with speech recognition software</td>
<td>Bad</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
</tbody>
</table>

TRADOS definitely offers the broadest range of source file support on the market. However, translation of RC files and Executables cannot compete with specialized software localizers. Because the TRADOS format has become a quasi-standard, all specialized localization software supports it.

Editions include TRADOS 5.5 Freelance ($795), which has several limitations; and several corporate packages. Details are available via the TRADOS Web site.

### General Conclusions

As none of the reviewed software applications proved to be a total lemon nor absolutely superior to their competitors, there is no simple answer to the question of which product would serve you best. In addition, the value of such tools varies greatly with your individual needs and circumstances.

The first thing you have to consider is whether you are positive that you will only localize software and never accept any task of translating another kind of data. If this is the case, you should consider buying one of the specialized tools for software localization: CATALYST, PASSOLO or RC-WinTrans. If software localization is only a small fraction of your translations or the software localization also requires translating larger text corpuses such as Help files, manuals, documentation or Web sites, you are much better off choosing a general translation memory solution such as TRADOS or Transit.

Another variable in the decision process is the preferences of your clients. If, for instance, you anticipate translating a lot for one of your clients who wants you to use a certain software, you would be ill-advised to buy another tool and risk losing that client. It is a matter of fact that besides personal likes and dislikes, once you’ve gotten used to any of the reviewed tools, they all will increase your productivity and consistency. In software localization, your client’s choices can also determine the file format you have to handle: binary files such as EXE, DLL or OCX — or RC files. In the first case, you can use any of the three reviewed products, whereas PASSOLO cannot handle RC files. In contrast, however, PASSOLO is the only of
the three specialized software localizers that can save back translated EXE files on a Windows 9x or Me system.

If your decision is not influenced by the choices of your clients, there are still several other aspects to take into consideration. Did it take you years to master your word processor? Do you have difficulty finding files on your computer? Or do you have a crisis every time you have to install a program?

If you answered yes to all of these questions, you might have problems using Alchemy CATALYST as a specialized software localizer and STAR Transit as a general TM solution. You should instead look at the demo versions of PASSOLO or RC-WinTrans and TRADOS, respectively. On the other hand, if you’re an experienced computer user and are fine messing around with tags in all their glorious forms, you should consider STAR Transit or Alchemy CATALYST, respectively. Both are by far the most powerful applications in their group but do not necessarily glitter by being easy to learn or use.

Finally, I am told that there are people out there for whom price does matter. In the product group of general TM solutions, TRADOS is about three times as expensive as its competitor STAR Transit. Similarly, among the specialized software localizers, Alchemy CATALYST, the most expensive product in this review, is about twice the price of its competitors RC-WinTrans and PASSOLO’s Team Edition. Unfortunately, CATALYST’s Translator’s Edition is no alternative to the much more costly Localizer or Developer editions, since it lacks the ability to allow for leveraging of previous translations. On the other hand, both RC-WinTrans and PASSOLO require TRADOS if a TM is needed or desired (at a minimum cost of about $645 for TRADOS Freelance). In the case of PASSOLO, an additional $650 is necessary for the purchase of the TRADOS Add-In that provides the interaction between the two programs. Furthermore, additional licenses of TRADOS are necessary if you want to deploy the TM to your translators. In such cases, you might actually be much better off acquiring Alchemy CATALYST’s Developer/Pro Edition. Another reason to go for CATALYST is its superior handling of the XML file format. In this aspect, it even exceeds the abilities of the two general TM solutions.

Thomas Waßmer is a multimedia developer and Web designer for Chemistry Education Network in Mainz, Germany, as well as a freelance translator, software reviewer and information specialist. He can be reached at www.infotom.com
The DIPLOMAT speech-to-speech machine translation (MT) project team at the Carnegie Mellon University Center for Machine Translation was working on sparse-data languages in a rapid-deployment environment to develop several prototype bidirectional speech-to-speech MT systems within nine to twelve months. The language-independent translation system needed significant amounts of textual data to train an example-based translation engine for several different language directions. I suggested that rather than reinvent the data-collection wheel for sparse-data languages, we could rapidly train a new system on one of the largest sets of parallel texts ever created: the Bible.

Within hours, we had downloaded the public-domain versions of the Revised Standard Version English Bible and the Bib-la Haitian Creole Bible from the Online Bible Web site, converted the file formats to be used under Unix and force aligned the parallel translations verse by verse. We also conducted sample checking with our Haitian team members to make sure that the alignment process had worked correctly.

The English<>Haitian Creole translation system developed and deployed by that research project was the first to significantly benefit from such research techniques based on the availability of the biblical texts for low-density languages. The Bible was one text resource used for several language directions developed through DIPLOMAT (www.lti.cs.cmu.edu/Research/Diplomat/).

That was about five years ago. I have discovered since then that other researchers were exploring the idea of treating the Bible as a multilingual corpus for computational purposes. Some were considering a multilingual Bible corpus as a “seed” translation lexicon for MT. Others were looking at using the Bible for document retrieval and OCR applications.

The on-line Polyglot Bible project contains the Gospel of Luke in Greek, Latin, Old Spanish, Modern Spanish, Portuguese, French, Italian, Romanian, German, Old English, Middle English, Early Modern English and present-day English. It allows Web users to search by passage or by keyword and then display the parallel text in any of the 30 languages (http://mdavies.for.ilstu.edu/bible/). In addition to compiling the Polyglot Bible, Mark Davies has conducted research on Old Spanish syntax using multi-million word electronic corpora with different versions of the Old Spanish Bible for the purpose of teaching Spanish linguistics (http://mdavies.for.ilstu.edu/personal/publications.htm#11). The ARTFL Project included a set of multilingual Bibles (www.lib.uchicago.edu/efls/ARTFL/public/bibles/index.html). One project included research into corpus annotation and translational equivalence. And neural networks research was also underway several years ago, according to Michel Philippe, with an experiment to develop a translation system using the Gensim/Neuronline G2 software.

Jeff Allen
In February 2002, someone mentioned to me the Web site of the Jesus Film for a few hundred languages, and I wondered about the value of the texts and sound tracks for natural language processing systems. It is obvious that the idea had already existed. The use of these resources had also been successfully implemented four to five years beforehand, and certain limitations had existed concerning use of the Bible as well as the manuscripts and speech soundtracks of the Jesus Film for developing speech-based MT engines.

At the same time, an inquiry from Claudia Gdaniec appeared on an MT discussion list: “Does anyone know whether there are documents with multiple reference translations available anywhere that could be used for MT evaluation purposes? (Any language)”

This request is quite unique because multiple reference translations are several different translated versions into the same language from the same single source text. Such translations are, of course, often difficult to identify and locate because of the rare need to (re)translate a document in several different ways.

This principle seems to go against common sense in the translation industry, where the focus is placed on translating a document once, aligning the source and target versions in a translation memory tool and using the resulting parallel text as reference material for all future revisions of the document and for related documents. A request to create a completely different translation based on an already translated source text is probably due to a poor-quality initial translation that could not even be salvaged as the baseline for a revision. The Bible significantly differs in this respect in that the large number of translation versions are not a reflection of translation quality, but rather an expression of covering the range of varying needs and expectations of the readership.

**Translations, Paraphrases and Localized Versions**

To grasp the value of what the Bible can provide for the translation software industry, it is first important to know how to decide which translation of the Bible to use. This, in turn, requires understanding the reason for having multiple translations of the Bible today for each target language.

A first distinction to make is the difference between a translation and a paraphrase. In Bible translation practice, a translation follows more closely both the wording and the meaning of the original languages: Hebrew and Aramaic for the Old Testament and Greek for the New Testament. A paraphrase, on the other hand, tries to explain and restate in the words of the target language what the biblical texts really mean. This is especially important for hard-to-understand passages. Localized versions known as cultural equivalents try to place the biblical texts in specific local contexts of modern society.

For a text that is meant to be understandable with respect to grammar and style of both the source and target languages, no target language translation of a text is truly “literal.” This is obviously due to the fact that the Bible’s original languages use different word order and grammar to structure their sentences when compared to the target language translations of the Bible that we read today. Even the use of a similar Greek or Hebrew word cannot always result in the same “literal” term in English (or French, Spanish, Quechua, Sango or Croatian) in all contexts.

The closest we can get to having a true “literal translation” is what is referred to as an interlinear version, which simply provides a linguistic gloss in the target language for each word or each word group found in the original language. An example of this is the Interlinear Greek/English New Testament (IGNT), which theologians, pastors, priests and other specialists of the Bible use to conduct research on the etymology, grammatical forms and overall linguistic context of specific words. Yet, an interlinear “literal” translation does not reflect how the Bible — a collection of letters, stories, poems and prophecies — was intended to be read by the source-language audience. Thus, an interlinear literal translation can be used as a tool to identify and study specific points, but it is not a version that presents the text in an understandable and flowing manner in the target language.

Within the group of “translation” versions of the Bible, we can, however, draw a further distinction between a formal equivalent translation and a dynamic equivalent translation. A formal-equivalent (also referred by some scholars as “literal”) translation tends to be more oriented toward
a word-for-word analysis and on maintaining, as closely as possible, the linguistic form of the original Hebrew or Greek texts within the target language translation. English version examples include the New American Standard Bible (NAS/NASB), the 1947 Revised Standard Version (RSV), the 1982 New King James Version (NKJV) and the 1769 Authorized Version/King James Version (AV/KJV).

A dynamic-equivalent (also referred to as “idiomatic”) translation is more thought-for-thought oriented and translates biblical words and phrases into clear and contemporary target language equivalents. Some of the key characteristics of a dynamic equivalent are that 1) contextual consistency has priority over word-for-word translation; 2) dynamic equivalence has priority over literal correspondence; and 3) the linguistic forms accepted by the modern audience have priority over traditional language forms. In general, this type of translation places a priority upon the intended meaning of the source language combined with comprehension in the target language. An English Bible example is the 1985 New International Version (NIV).

All modern target translations of the Bible use the general concept of dynamic equivalence to a certain extent. The differences are not essentially a matter of whether or not the concept is followed but rather of how much one version uses it as compared to other versions.

Each of these two types of translations has advantages as well as potential drawbacks. The formal-equivalent translations are more accurate with respect to terminology and the structure of the original language; but they may sound awkward in the modern colloquial style of the target languages and thus can be hard to read and understand. The dynamic-equivalent translations are often much easier to read and focus on how to translate the intended meaning of the original language into the target language; but in difficult-to-understand sentences, one may lose a bit of the original meaning of the source-language text in order to make the sentence understandable in the target language.

**Paraphrase Versions**

The second main type of version is the paraphrase Bible (also referred to as free translations), similar to the dynamic-equivalent versions, which are more concerned with clarity than with exact wording. In addition, they are often not direct translations from the original Greek and Hebrew texts but can result from consulting one or more existing translations. These versions favor readability and comprehension, including simplified and restricted use of vocabulary, for native speakers as well as non-native speakers of the target language. With a focus on modern colloquial expressions in the modern language, they can give the impression that the Bible was written in the twentieth century. For example, the word in Psalm 119:105 that is translated as lamp in the KJV and NAS versions is translated as flashlight in the 1967 Living Bible (TLB/LB). Flashlights obviously did not exist a few thousand years ago, but the use of the modern-day flashlight corresponds closely to the meaning expressed in the text.

An advantage of such versions is the readability of the text and the ability for today’s readers to understand some concepts that existed centuries ago with equivalent interpretations in modern-day society. The known drawback of such versions is that they can compromise on the original meaning of the original source text. Some English Bible examples are the Good News Bible (GNB) and TLB.

**Cultural Equivalents**

The cultural equivalent is an example of taking localization to the extreme. Although I have not read it myself and I know of no one who has such a version, I have been informed about the Cotton Patch Bible, which places the biblical narrative in the context of the South in the United States. This version apparently translates the Jordan River as the Mississippi.

The value of such versions is limited to helping people understand the original context through their modern-day locale, but such versions certainly stray from the intended meaning of the original texts. If king is translated as president or prime minister, if Mount Sinai becomes Mount McKinley, and if the route to Damascus becomes Route 66, then the biblical text can easily be misinterpreted by existing connotations applied by each individual reader based on his or her real experience with these modern-day and locale-specific equivalents.

**Perfect vs. Appropriate Translations**

The reason for explaining Bible translation theory and practice is to show that the existence of multiple translations of the same source text can indeed be intentional and has a certain value for different types of readers. Despite the utopian search for and the often advertised concept of the “perfect translation” for marketing purposes, which can be misleading for anyone who is not at all familiar with translation practice, it is better to speak about a translation which is appropriate or not appropriate depending on the criteria of the intended audience and their expectations. The needs and expectations of the readership are definitely guiding the translation to the extreme. Although I have not read it myself and I know of no one who has such a version, I have been informed about the Cotton Patch Bible, which places the biblical narrative in the context of the South in the United States. This version apparently translates the Jordan River as the Mississippi.

The cultural equivalent is an example of taking localization to the extreme. Although I have not read it myself and I know of no one who has such a version, I have been informed about the Cotton Patch Bible, which places the biblical narrative in the context of the South in the United States. This version apparently translates the Jordan River as the Mississippi.

The value of such versions is limited to helping people understand the original context through their modern-day locale, but such versions certainly stray from the intended meaning of the original texts. If king is translated as president or prime minister, if Mount Sinai becomes Mount McKinley, and if the route to Damascus becomes Route 66, then the biblical text can easily be misinterpreted by existing connotations applied by each individual reader based on his or her real experience with these modern-day and locale-specific equivalents.

**Paraphrase Versions**

The second main type of version is the paraphrase Bible (also referred to as free translations), similar to the dynamic-equivalent versions, which are more concerned with clarity than with exact wording. In addition, they are often not direct translations from the original Greek and Hebrew texts but can result from consulting one or more existing translations. These versions favor readability and comprehension, including simplified and restricted use of vocabulary, for native speakers as well as non-native speakers of the target language. With a focus on modern colloquial expressions in the modern language, they can give the impression that the Bible was written in the twentieth century. For example, the word in Psalm 119:105 that is translated as lamp in the KJV and NAS versions is translated as flashlight in the 1967 Living Bible (TLB/LB). Flashlights obviously did not exist a few thousand years ago, but the use of the modern-day flashlight corresponds closely to the meaning expressed in the text.

An advantage of such versions is the readability of the text and the ability for today’s readers to understand some concepts that existed centuries ago with equivalent interpretations in modern-day society. The known drawback of such versions is that they can compromise on the original meaning of the original source text. Some English Bible examples are the Good News Bible (GNB) and TLB.

**Cultural Equivalents**

The cultural equivalent is an example of taking localization to the extreme. Although I have not read it myself and I know of no one who has such a version, I have been informed about the Cotton Patch Bible, which places the biblical narrative in the context of the South in the United States. This version apparently translates the Jordan River as the Mississippi.

The value of such versions is limited to helping people understand the original context through their modern-day locale, but such versions certainly stray from the intended meaning of the original texts. If king is translated as president or prime minister, if Mount Sinai becomes Mount McKinley, and if the route to Damascus becomes Route 66, then the biblical text can easily be misinterpreted by existing connotations applied by each individual reader based on his or her real experience with these modern-day and locale-specific equivalents.

**Perfect vs. Appropriate Translations**

The reason for explaining Bible translation theory and practice is to show that the existence of multiple translations of the same source text can indeed be intentional and has a certain value for different types of readers. Despite the utopian search for and the often advertised concept of the “perfect translation” for marketing purposes, which can be misleading for anyone who is not at all familiar with translation practice, it is better to speak about a translation which is appropriate or not appropriate depending on the criteria of the intended audience and their expectations. The needs and expectations of the readership are definitely guiding the translation process. Based on such criteria, the finished translation product can be judged to be highly appropriate, appropriate, less than appropriate and so on. Likewise, each of the current Bible versions is appropriate according to its stated purpose.

Interlinear literal versions are valid for studying terminology and grammatical issues in the original languages. Formal-equivalent versions give a target-language translation, but try to keep the overall grammatical sentence structure and the best rendition of terminology from the source language. Dynamic-equivalent versions focus on providing a thought-for-thought translation at the sentence level. Paraphrases give more modern-day equivalents for harder-to-understand sentences. Those involved in the translation process for each of these versions have attempted to preserve the meaning of the original texts as best as possible, while balancing this with how to describe the words and ideas in the target language and according to a specific translation approach. The result is that each version is a rich source of
RELIABLE TRANSLATIONS

During my research over the years concerning these many different Bible translations, I have discovered that all of the direct translation versions and even the paraphrase versions have been compiled and edited by teams of native speakers, language specialists, linguists and often Bible scholars. Hardly any new version is undertaken by a single individual. This is understandable since the text itself represents roughly 800,000 words within approximately 60,000 sentences. A full-time translator who completes an average of 3,500 words per day would take at least 230 working days to finish this volume of work.

We must remember, however, that these translations are done from Ancient Hebrew and Ancient Greek — neither of them a contemporary mother-tongue spoken language — so it is not the case of a bilingual person who decides one day to make a living doing translation work, which does happen in the translation industry. Many well-known Ancient Hebrew and Greek scholars who participate in new translation projects for major international languages such as English, French and German hold doctorates in theology or divinity (four years for a bachelor's degree in theology/Bible, then four years for a master's in theology/divinity and also four years for a doctorate in theology/divinity) and then many years of teaching these languages in Bible colleges or seminaries. Just the preparation time of learning and mastering these languages is 10 to 20 years. As stated in the Preface of the NIV, “The New International Version is a completely new translation of the Holy Bible made by over a hundred scholars working directly from the best available Hebrew, Aramaic and Greek texts.” That translation project began in 1965, and the first edition appeared in 1973, including one general editor and four associate editors. It is not surprising that such a translation for a new version of the Bible in English by such a large group of recognized specialists took nearly 10 years to complete.

As for translations undertaken for modern less-prevalent languages, the number of languages is significant: 700-plus in Indonesia, 800-plus in Papua New Guinea, more than 500 in Nigeria, 200-plus in Australia or more than 150 in the Philippines.

Three main groups involved in such Bible translation work are Wycliffe Bible Translators, New Tribes Mission and SIM. Translation projects for these languages are conducted and led by specialists who have followed two to three years of post-graduate intensive study with linguistics courses such as the non-written language specialized courses offered by the Summer Institute for Linguistics International (SIL) and sets of other advanced courses in Bible translation, literacy education, cross-cultural communication and so on. The average translation project with two to four full-time linguists takes about 15 years to complete.

These Bible translations are definitely not short-term projects that produce low-quality and poorly reviewed translations due to high-pressure turnaround schedules. They are, rather, long-term projects with needs analyses — sociolinguistic-based language surveys covering issues pertaining to dialect inter-intelligibility, language survival and other factors. The overall goal is to provide the best translation possible for any given language based on the resources that can be allocated. Founded in 1936, Wycliffe as an international organization had by 2000 finished the translation of the complete New Testament for more than 500 different languages, plus partial translations of the Old and New Testaments for an additional 1,000-plus languages.

In addition, the many Bible societies such as the American Bible Society and the International Bible Society, which are more or less responsible for ensuring the quality, diffusion and copyrights for nearly all printed Bibles, have been in existence since the nineteenth century.

Any new Bible translation produced for an international language must demonstrate adherence to a rigorous evaluation cycle in order to attain any level of acceptance compared to all of the other existing translations. The 1995 Contemporary English Version (CEV), for example, states, “The drafts in their earliest stages were sent for review and comment to a number of biblical scholars, theologians, and educators representing a wide variety of church traditions. In addition, drafts were sent for review and comment to all English-speaking Bible Societies and to more than forty United Bible Societies translation consultants around the world. Final approval of the text was given by the American Bible Society Board of Trustees on the recommendation of its Translations Subcommittee.”

Translation projects undertaken by Wycliffe take several years with translation reviews by local native speakers as well as experienced external Bible translation consultants. Does your preferred translation supplier — even one who is ISO 9001:2000 certified — provide such complex and rigorous in-depth review and evaluation within the ever-increasing and constant high-pressure turnaround cycles at the end of the chain of the software and documentation cycles?

Faced with having to develop and train the first prototype version of a translation system, would you start with five megabytes of electronic Bible texts published in modern-day Bahasa (Indonesia) in 1994 after a translation and review period of 15 years by such a recognized organization? Or would you first try to locate existing texts in Bahasa and then...
decide to use 10 different 10- to 20-page health and education documents, each translated in approximately two weeks by one of five to ten different translation suppliers? And next month, if you need to develop a prototype system for Shuar (Ecuador), would you use the electronic texts of the modern Shuar Bible completed in 1982 or conduct the same search and identification process for documents available in Shuar? It can take months even to locate 10 documents in such a language.

If only hardcopy texts can be found, then the process of scanning, reviewing and manually correcting the texts one page at a time is extremely time-consuming, especially for languages in which customized OCR software is not available.

And what if you locate a gold mine of electronic texts for the language — but discover that the documents were sent by e-mail in the early 1990s, and the authors removed all of the accent marks in the texts? This was done, rather than moving the accent mark to the immediate right or left character position of the letter, in order to avoid creating the hieroglyphic characters such as =E10 or =E11 that many of us were used to reading in non-English e-mail messages during the first decade of widespread Internet use. Creating an accent-reinsertion script is a nearly impossible task without having an initial text database for that language. Some semi-automatic processes can be put into place based on existing language grammar books and linguistics articles concerning the language, but multiple forms (such as when the words se, sé and sè exist and are common in the language) can lead to an automated process that is less efficient and more error-prone than the manual operation.

Because the Bible is a frozen content source text and because such intense editing and quality assurance methods are followed for the target language versions of the texts, we can place a high level of confidence in the quality of work invested in Bible translations. High-quality reviewed texts are important for training translation software programs and systems, and so these different versions of the Bible are extremely valuable for such development work.

**Multi-reference Translations**

This article started with a mention of the need for multi-reference translations. First, many people assume that the Bible is only available in Old English with thee, thou and ye found throughout the entire text. If this were the case, I would not even be suggesting the use of the Bible for training modern-day language applications. There is an entire range of versions that cover different types of needs and demonstrate the many different ways of expressing the same semantic concepts in different grammatical and terminological clothing. The accompanying examples show grammatical, semantic and stylistic differences that can be found in just a few verses from English language versions.

These different sentences provide a rich sample set for conducting intense empirical research on multi-reference texts and obtaining valuable statistical analyses on segments (words and phrases) across the entire set of Bible databases. One example might be the number and variation of grammatical contexts that the word over occurs in, especially within all of the parts of sentences where it has the same meaning. This is where the multi-reference nature of the database becomes important. Although Web searches across the Internet may find all of the contexts that the word over can occur in, that type of search is often limited to statistical counts. This database provides the semantics-based added value of being able to determine how many of the examples have a specific underlying meaning because they can be directly compared with other ways of saying the same idea. Another example from the verses provided would be research into the frequency of the words until, till and unto, as well as other related variants, based on semantically similar and contrasting contexts.

I will not attempt to show all of the possible linguistic analyses that these few sentences can provide, but obviously a significant amount of data can be gleaned from just a few samples of how to say the same idea in different ways. Such analyses provide linguistic-based statistical derivatives and by-products that translation software and other natural language technology developers need and often seek for implementing certain improvements into their various tools. Such analyses can significantly improve

---

**Sample Texts in Several Translations**

1 Corinthians 7:21

1769 Authorized Version/King James Version (AV): Art thou called being a servant? care not for it: but if thou mayest be made free, use it rather.

1884 Darby Bible (DBY): Hast thou been called being a bondman, let it not concern thee; but and if thou canst become free, use it rather.

1898 Young’s Literal Translation (YLT): A servant — wast thou called? be not anxious; but if also thou art able to become free — use it rather.

1902 Rotherham’s Emphasized Bible (R1902E): Were you a slave when called? Never mind. But if you can get your freedom, rather use it.

1905 Bible in Basic English (BBE): If you were a servant when you became a Christian, let it not be a grief to you; but if you have a chance to become free, make use of it.

1912 Weymouth New Testament Translation (WEY): Were you a slave when God called you? Let not that weigh on your mind. And yet if you can get your freedom, take advantage of the opportunity.

Matthew 27:45

YLT: And from the sixth hour darkness came over all the land unto the ninth hour.

AV: Now from the sixth hour there was darkness over all the land unto the ninth hour.

BBE: Now from the sixth hour it was dark over all the land till the ninth hour.

DBY: Now from the sixth hour there was darkness over the whole land until the ninth hour.

1902 Rotherham’s Emphasized Bible: Now, from the sixth hour, darkness, came upon all the land — until the ninth hour.

NKJV: Now from the sixth hour until the ninth hour there was darkness over all the land.

PHIL: Then from midnight until three o’clock darkness spread over the whole countryside.

WEY: Now from noon until three o’clock in the afternoon there was darkness over the whole land.
the Auto Learn function of a TM tool, for example, or can determine new translation algorithms to include in translation software. I would even say that translation software developers should first exhaust the linguistic potential of information in these databases before trying to use texts found on the Internet to build linguistic hypotheses for translation rules.

**Word Frequencies**

Each word in the Old and New Testaments with its morphological and grammatical status is accounted for and can be traced throughout the reference tool Strong’s Bible Concordance. Each lexical item is referenced according to a numerical referencing system. These numbers are then associated to a lexicon which is also available in English and French. Word counts have also been compiled per lexical item in this concordance. Such a tool is extremely valuable for any type of advanced translation technology research. An electronic version of this concordance is available in the Bible Online software program.

**How Much Text and How Many Versions?**

Customized translation system research usually requires hundreds of megabytes of parallel texts in order to produce any sort of valuable results, but high volume is not the main requirement for multi-reference translation data. An entire Bible (Old and New Testaments) with single-byte Latin characters is around five megabytes of plain ASCII text. The number of electronic versions of the Bible available per target language is constantly increasing. The current version of the Bible Online (BOL Millennium v1.11) comes with more than 60 versions of the Bible on a single CD.

Taking the number of electronically available versions for the two major languages French (11) and English (18) that can be aligned verse by verse, it is possible to produce nearly 200 different parallel version combinations. Two hundred parallel versions multiplied by approximately 60,000 sentences (containing approximately 800,000 words) indicates the amount of text available for conducting research on syntactic and semantic structures to develop, train and improve translation software and systems.

**Conclusion**

The Bible is a translated document that shows significant potential in the area of fundamental translation research for the development of translation software and systems. As the most translated document in history, this single source document with multiple reference translations for each target language is a rich source of linguistic data for natural language research and development in areas such as MT (rule-based, example-based and statistics-based), TM, terminology management, multilingual electronic dictionaries, text/message understanding, automatic lexicon recognition, information retrieval, text summarization, document indexing, topic detection, text mining, multilingual word processing, grammar checkers, style checkers and speech synthesis.

More than 60 electronic versions of the Bible are currently available for 30 languages; hundreds of other Bible translation projects are in progress. The Bible could become an influential source of linguistic material for the next generation of language technology applications. A key issue will be adequate supply and demand channels for such biblical texts with licensing schemes that fit different short-term to long-term research and development needs.

![Jeff Allen leads the software testing and technical documentation departments at Mycom International. He is also a member of the MultiLingual Computing & Technology editorial board. He can be reached at jeff.allen@free.fr](image)

### References


### Bible Translation Web Resources

For more information about Bible translation theory and practice:

- www.anchorlife.org/bible/html/study_bible.htm
- www.lcms.org/cic/alilthose.htm
- www.ntm.org/intl.shtml
- www.sim.org
- www.wycliffe.org/features/500thNT/home.htm
- www.americanbible.org
South Asia has been slow to move to localisation. The elite of the region, those who can afford to buy and use computers, are also very fluent in English. This is only 5% of the total population, but in this region of well over a billion people this still adds up to many more people than in the UK and Ireland. The current use of computers in South Asia, including e-Content, is overwhelmingly in English.

But look at this web page from the site of ICICI bank, (www.icicicommunities.org/communities/index.asp) India’s second largest bank with over 500 branches across India.

The whole site uses a lot of red, a colour of prosperity and happiness, much used by brides; it also uses a lot of saffron, an auspicious religious colour.

This particular page offers the facilities to make donations to charity. Giving to charitable donations is an important part of Hindu culture, and other Indian banks, but not all of them, include similar facilities. The previous version of this page was relatively sparse, with Indian images, offering not just charitable donations but also the ability to make pujas. Often when people are involved in financial transactions they will stop at a shrine and undertake a small ritual or puja, seeking blessing and good fortune in this transaction – this site offers simple facilities, but some specialist puja sites enable a full ritual, for example, of offering rice or lighting candles. The important thing to note is that the website has been used to express important aspects of Hindu culture. But why is it in English, and not in an Indian language, when a language is a deeply important component of a culture?

Well, some sites do now offer a choice, but historically this has been problematic. There are some 500 distinct languages in South Asia, though the exact total is contestable as the division between dialect and language is not that clear. These languages fall mainly into the three language families of Indo-European across Pakistan, North India and the foothills of the Himalayas, through to Bangladesh; Dravidian in South India, and Tibeto-Burmese in the Himalayas. The languages of Pakistan and Afghanistan are written with extended Arabic writing systems, arising from Islamic influences. All the other languages, as well as those across South-East Asia, are written with alphabetic writing systems that came from a common ancestry, Brahmı – some 15 distinct writing systems in all. We show some examples below, taken from banner lines on web pages.

Tamil:

Hindi (devanagari):

Kannada:

As soon as PCs became available, and it became easy to create new Roman fonts for PCs, the facilities were used to create fonts for South Asian scripts. But it was not that easy. Each alphabet has some 50 or more distinct letters. These are often combined in writing as ‘conjunct’ characters composed of two or more letters. The figure below shows just two examples of this. The two letters on the left are combined to form the conjunct on the right.

Each writing system has many hundreds of distinct characters composed as conjuncts from the underlying alphabet, and this means that there are many hundreds of distinct characters that could be used for printing and display. They could not fit into a single font table, particularly when this local language character set was required to exist alongside the...
Roman alphabet. Compromises had to be made, and each font creator made their own different compromises, aiming to sell fonts on the basis of their visual appearance. Font creators gave little regard to the actual internal encoding for the characters, and often these encodings arose as an accidental by-product of how the keyboard is laid out. The result is that the needs of word processing and desktop publishing are met, more or less, but data cannot be shared across the Internet unless all parties have the same font from the same supplier.

This is bad for localisation in South Asia, with poor quality desktop publishing, and it makes data sharing nearly impossible.

The resolution to quality came with the realisation that computers could be smart enough to work out what compounds were needed to present writing to people, and all that was needed to represent any Brahmi writing system was the base alphabet of around 50 letters. A rendering engine would then combine letters as needed, even moving letters backwards or forwards if needed. This is how the original Indian ISCII standard, developed during the 1980s, worked and this is how Unicode works. Originally ISCII required that special purpose GIST hardware was purchased, and later that special software was purchased. Regrettably these extra costs caused people to select the outdated PC fonts because they were either cheaper to purchase or easily copied from a friend.

The coming of Unicode in the late 1990s should have resolved this, with Windows 2000 and XP including Unicode and Indic font rendering engines. Linux now offers similar facilities. Here is an example of how it should work, adapted and extended from the Unicode manual.

The various letters are typed in the sequence in which the letters would be spoken, using one of a number of alternative keyboard layouts. These are mapped to a canonical sequence of internal codes using some suitable software. The internal codes are stored, and when required they are rendered, forming the conjuncts, selecting the right shape for each letter, and maybe even reordering letters.

But this Unicode capability did not solve matters.

Standards, and their critical importance for exchange of data, do not seem to be widely understood. The historical preoccupation with desktop publishing still dominates; a meeting in 2001 was packed with activists advocating PC 8-bit fonts.

But more seriously perhaps, Unicode was based on ISCII, and ISCII was developed in North India. People from South India have concern about shortcomings of Unicode for the scripts of their Dravidian languages. Tamil Nadu has now gone it alone and defined their own different encoding for Tamil, with sound linguistic reasons for doing this. Clearly the encodings in Unicode need proper review and revision, and the Unicode consortium needs to be pro-active in seeking this.

It is not clear just what the market for local language software in South Asia is. True, there is enough interest in South Asia to pursue the coding issues above, but where is that drive coming from? The National Government of India has long mandated that all government information should be delivered to them in the official languages of India, making available funds to support this, and this has fuelled some of the developments. But it is not sure how much they actually use this information, and what the real need is. An accounting package working in Hindi failed, and there are very few websites with information in local languages.

What has sustained developments is not the commercial market, but rather it has been the passionate enthusiasm of many people believing that this must come about. In particular, people believe that in order to make the benefits of IT available to everybody, IT must spread beyond the 5% competent in English and must work in local languages. A meeting in 1998 in Bangalore drew up the Bangalore Declaration (www.bangaloreit.com) which emphasised this, and led to the Simputer (www.simputer.org) with its emphasis on low price and local language use.

The Indian national government continues to fund developments in local languages (http://tdil.mit.gov.in/), and we are seeing many developments here, from character recognition systems to text-to-speech generation. Most significantly, several projects are localising Linux, complete with rendering engines. There are several projects for localising Linux around the region, though alas these seem to be progressing independently of each other. More commercial products,
word processors and accounting packages, are beginning to emerge. And in 2002 the Indic Computing group was launched, with active email lists and a workshop, for people to share knowledge and experience in localising software to South Asian local languages.

Will this gradual development of local language support slowly generate the market for local software? India has a vibrant software industry, but only about 20% of its software development effort is focused on internal markets - the rest is focused on external markets, often in local branches of major US software companies. This is contrasted with China, where some 80% of their software development effort is focused internally. Could it be that India's very competence in English has inhibited the development of its own local language markets?

Nevertheless, we are very optimistic about the development of these local language markets, with a lot of the needs met by internal software development, but also a considerable amount of imported software localised, at least to the most important of South Asia's languages. The Indian national government has financed 13 centres for localisation, though these have been placed in research institutions. It is not clear just how much localisation advice they are really giving. Much more promising now is the Indic Computing group, and their intention to make advice available on their website http://indic-computing.sourceforge.net/index.html. 

Pat Hall, The Open University, UK, can be reached at p.a.v.hall@open.ac.uk.
Shailey Minocha, The Open University, UK, can be reached at S.Minocha@open.ac.uk.
Venkatesh Hariharan, IndLinux, India, can be reached at venky1@vsnl.com.
Tapan Parikh, Media Lab Asia, Mumbai, India, can be reached at tap2k@yahoo.com.
Internationalisation (i18n) and localisation (l10n) are intricately linked processes. A lot of attention has been paid by localisation service providers to their internal processes, and some of their larger customers have worked hard on their own internal processes. A similar amount of attention has been paid to low level issues of enabling software to support various character encodings by developers. However, in my experience, neither of these advances addresses the needs of most companies that are attempting to deliver international products for the first time, companies that may not have the necessary time, resources, or experience to develop stable processes.

For these companies, the working assumption is that creating and delivering international products are possible, and should not be difficult. After all, many other companies have obviously done it, they reason. It is hard to argue with this kind of logic, yet time after time it is a struggle for companies to “go international” and integrate best practices into their existing processes.

The Hidden Areas

In this article, I will discuss one of the hidden areas where internationalization and localisation processes and techniques overlap and cause trouble. While there are other issues I don’t address here, understanding these issues and applying the results will go a long way towards making sure your international efforts are as effective as possible.

Most important in the success of any localisation effort is a thorough understanding of the state of internationalisation of the product and project. Internationalization and localisation are tightly linked and tasks involved in each of these are often performed by separate groups (see the article at www.i18n.com/article.pl?sid=01/11/03/189205 for a brief explanation of the differences of internationalization and localisation).

The two parts of the project, internationalization and localisation often are separated. They may be done by different teams, even in very widely separated parts of a company’s org chart. For example, internationalization may be part of Product Development while localisation may be part of Operations. It is common for Sales to be the group driving and sometimes even owning both parts, even though they are far from the more traditional charter of a Sales team. Marketing groups may be on their own when it comes to localising outward facing Web sites.

Who actually performs the work? The people involved in internationalization and localisation may be doing so as outsourcers or part of the main company, or a combination of both, and anyone could be located any place in the world. These issues introduce a severe communication problem that needs to be managed carefully.

Measures of Success

There is an old saying – “You can’t manage what you can’t measure”, and an enormous part of the techniques and management is without metrics at all. I propose at least two places where metrics would make processes work much smoother. Space doesn’t allow me to propose specific metrics here, but I hope to encourage everyone to consider these issues carefully.

The consensus definition of internationalization is usually along the lines of “prepare a generic, locale neutral product that is ready for localisation”.

This generic definition is crafted to ignore the vast differences in coding styles, systems, and tools used to create a product that needs localisation.
**Coherence and Cohesiveness**

What does it mean for products to be internationalised? How can we describe the level of internationalization in a way that we can track it over time, and eventually correlate it to some sort of metrics of the subsequent localisation efforts?

The vast majority of software products consist of a number of components that interact with each other. Depending on the skill of the software architects, these components may be nicely layered. The technical terms for this are coherence and cohesiveness, terms that refer to how well the blocks fit together, yet stand independently.

**Architecture and Internationalization**

Many software engineers and managers, without a formal training in computer science, tend to be unaware of these concepts. Yet, their importance to QA, maintenance, and new feature additions is well known in academic circles.

Certain aspects of these features are important in internationalization work. Let’s use a semi-fictional example of software and examine its architecture with respect to internationalisation issues.

What I call a software module may range in size from a few lines of code to complex subsystems. The important aspect for our purpose is that a module always accepts input from another module, transforms the input to some output format, and then passes the output either back to the first module or to a different third module.

**Character Encoding**

Software modules often need to pass plain text data among themselves. As localisation professionals familiar with character encoding are aware, there are large numbers of character encodings in common use and they are not always compatible with each other. Managing this technical single issue is a core part of internationalisation engineering.

Because the software modules are often created independently of each other, each may have its own expectation of how the textual data is formatted at the input, internal processing, and at output stages.

What this means, is that even though each module may (or may not) be internally consistent with respect to how it manages text encoding, the system as a whole may not be. In fact, it probably isn’t. Textual data may undergo various encoding transformations along its overall path, and these may not be consistent, correct, or sufficient.

In practice, it is rare that a software team will be able to describe this situation in detail either after or especially before internationalisation commences. But it is also clear that a thorough understanding of the data path of locale-related data is important. This is a ripe area for identifying and applying metrics.

**Website Application**

Our semi-fictional example is of a Web application that is used to receive and respond to customer emails in high volume. This example illustrates many of the principles we are concerned with. Before we look at how to measure internationalization status, let’s take a look at what the system does.

Data arrives at the system as ordinary e-mail messages, via the standard SMTP protocol. E-mail messages are then retrieved from a mail server via the POP protocol, just as most of our desktop mail clients do. In this case though, the messages are broken down into constituent parts, and the parts stored in a database, which might be Oracle, SQLServer, Postgres, or any number of other database systems.

Once the e-mail message is in the database, a Customer Service Representative (CSR), using a Web browser based interface, can display the messages and compose a response to the original sender. The response may include text that is created by some sort of natural language processing modules that analyse the data. The final response is stored back in the database, and then sent on its way via SMTP, just as any ordinary email is.

**The Modules**

With that brief description of functionality, let’s identify and analyse the actual modules that the message goes through. Remember we are just focusing on the text for now. Similar analysis needs to be done on other locale-related data, such as dates and times, domain names and anything else a particular application needs to be aware of.

Here are some of the modules that are external and not under the control of the software developers (beyond the choice of which version to use):

- **Incoming SMTP server**: This software is supposed to receive messages according to a series of internet specifications, called RFCs. But the RFCs have conflicting and subtle issues regarding character encoding. Without identifying and tracking these, all downstream data flow may be suspect!
- **POP Server**: The POP server is also supposed to behave according to another set of RFC specifications. These RFCs may or may not impact character encoding integrity. Without a careful understanding of these issues, downstream data may also be suspect.
- **Network transport layers, HTTP, etc.**: These protocols also introduce their own ideas of what character encodings are identified, expected and allowed.
- **Choice of Web server**: The ways a browser interacts with a web server to identify what the user’s preferred locale settings are, are complex and ill defined. They may vary by server implementation as well.
- **Web server scripting language**: The web pages that the CSR sees are necessarily generated dynamically. This means there are programmes that communicate with the web server that are responsible for generating the HTML that is sent to the browser. The choice of programming languages and tools for this task introduce another set of variables about which character encodings are supported (perhaps correctly and completely, perhaps not).
- **Browser scripts (Javascript)**: The HTML that is ultimately sent to the browser may contain Javascript for client side processing of data. Javascript implementations vary across browsers though in some key ways with respect to text handling.
Server and client side Java Virtual Machine. Similarly to
the client-side Javascript implementation issues, the choice of
Java as a programming tool, either on the server or the client
raises issues. Java relies on Java Virtual Machine (JVM) to pro-
vide services that traditional programmes have received from
the operating system directly. However, there are variations in
the services that are available and the correctness of the imple-
mentations across different versions of various JVMs.

Database drivers. The layer of software directly responsible
for communicating with a database is called database driver. This
module is responsible for converting text data as presented by
the application, to the form that the database can actually use. It
also converts in the other direction when records are retrieved
from the database. The supported conversions may be limited,
and again, the available support may not be always correct.

Database Character encoding capabilities. Independent
of the database driver, the set of character encodings that can
be stored in a database is limited.

Client side browser. Different browsers support locale-
related display and formatting issues differently, if at all.

Client side fonts. Independent of the browser's algorithms
for displaying text, a font may or may not be available for a
given character and encoding.

Client and server operating systems. The locale-related
services that an operating system, which all of the above mod-
ules must run on, are far from standardised. There are usually
a lot more issues to be discovered in modules close to the
operating system.

Each of these modules is certain to have issues related to
whether or not text passing through will behave as expected. In
some cases, the specifications are ambiguous. In some cases,
the implementation may be incorrect. In others, data may be
left in a format that is difficult to use. And other modules pre-
sent in the system may be an out of date version.

This is just a description for the plain text data. Similar
analyses need to be done for all of the types of locale-sensitive
data you may have identified in your system, which will surely
be numerous.

Without understanding in detail how the code your developers
are responsible for interacts with and take advantage of the ser-
vice present in modules you can’t change, it is impossible to say
what the level of internationalisation is in your project.

Recommendations

For each type of locale-sensitive data in your system, create
a detailed dataflow diagram for all modules that the data pass-
es through. Draw each module as a node on a graph, and
connect the nodes with all input and output data flows.

Label each connection with an arrow indicating the direc-
tion and information about the state of the data at that point,
such as the encoding scheme for text. Label each node with
the module name and the type of transformation (or “NONE”)
that takes place there.

If more than one transformation takes place, then split the
node and add new connections until there is only a single
transformation in each node.

System analysts among the readers will recognise what
results as a Dataflow diagram in the Yourdon tradition. Except
this time, instead of using it to design a new system, we are
using it to understand an existing one.

This type of understanding of the system’s architecture pays
big dividends in understanding where internationalization
development effort needs to occur, where the trouble spots
are, and how complete the internationalization effort is.

A spreadsheet or other tracking system can be set up with
entries for each element (node, connector) on each data flow
diagram. Bug reports, project schedules, etc. can be tracked
to a specific point in the chart.

Over time, this type of data can be analysed to see where
there are tricky spots in the code or overall data flow turned
out to be. Finally, once this data analysis has taken place, it
can be used to predict troubles with the localisation process
itself.

Barry Caplan has more than 10 years
analysing complex software systems for
internationalisation issues and building
cross-functional global software teams that
produce. He can be reached at
bcaplan@i18n.com
While there are many theoretical approaches to software localisation discussed in present day literature, there is a lack of applicability of the specifics of this process to actual locales. I am going to document some key linguistic issues, character code issues, cultural issues and local conventions that are involved in a typical software localisation process, applying all these considerations to the specifics from English to Spanish.

Writing Systems

Writing systems specify key linguistic and non-linguistic issues such as word structure, direction of writing, punctuation, collating sequences and so forth, together with the variants of each language. There is a need to account for a number of these issues when localising a product from English into Spanish.

English and Spanish use the Roman writing system, a phonological and alphabetical writing system. Alphabets are a foremost concern for the localisation process since the appropriate management of characters is essential for representing and displaying the correct information. Spanish adds to the standard Roman character set special letters and diacritical marks. A software application localised into Spanish must be able to handle Spanish special characters and diacritical marks. The Spanish alphabet added, since the fourth edition of the academic Dictionary in 1803, three more characters to the 26 English ones: <ch>, <ñ> and <ll>, both in their uppercase and lowercase forms. In 1994, attending to the petition of several international organisations, the Asociación de Academias de la Lengua Española agreed in its X Congress in Madrid that the characters <ch> and <ll> were no longer treated as a two-character combination corresponding to one unique character, but as two characters (Real Academia Española, 1999). The Spanish language also uses diacritical marks: <á>, <é>, <í>, <ó>, <ú> and <ü>, both in their uppercase and lowercase forms, which introduce a change in meaning and in pronunciation.

The notion of a collating sequence is closely linked to the notion of an alphabet. Special characters and accented characters affect the manner by which sorting and searching are undertaken. Since <ch> and <ll> are no longer treated as one character, <ch> collates in the alphabetic letter <c> between the words that begin by ce and ci; while <ll> collates in the alphabetic letter <l> between words beginning by li and lo. In Spanish, character variants are treated as equivalents, for example, the characters <á> and <a> map to the same position in the sort order, because they represent the same character. It is important to point out that this is not the case for the Spanish character <ñ>, which is not a specific variant of <n> with an accent mark, but a different character or grapheme, which sorts after <n> and before <o>.

The morphology and syntax of a particular language affects text processing and information retrieval functions when software products are localised for different markets. English is an analytical language while Spanish is considered as a moderately synthetic language where nouns inflect for number and gender, whereas verbs inflect for mood and tense. However, Spanish has lost all trace of case inflection and it is very prone to the use of prepositional complements, while English is keen on pre-modification and compounds. This deeply affects the process of localisation from English into Spanish in areas such as text expansion and concatenation of strings.

One of the problems related to the localisation of strings/messages is length restriction. English text strings
usually expand when translated into Spanish. Experts advise to leave between 30 and 40 percent more space to avoid truncated strings. Another problem is concatenation. It is recommended not to concatenate strings when localising from English into Spanish, since English allows more flexibility than Spanish when putting strings together. Morphological and grammatical rules should be considered, such as the rules for forming the plural of nouns or the conjugation of verbs, gender of nouns and word order – for example, in English the adjective is placed before the noun, whereas in Spanish the adjective is placed after the noun. English nouns do not indicate gender, whereas in Spanish the gender of a noun influences the spelling of other words in a sentence.

There are also non-linguistic conventions such as punctuation. There are differences in punctuation between English and Spanish that extend to the field of grammar rules, for example the opening question and exclamation mark in Spanish at the beginning of a question and at the end, respectively.

**Character Code Issues**

The computer does not handle characters as written symbols; instead, they are internally represented as numeric codes, since computers can only process binary data. The appropriate management of characters is essential for processing and displaying the correct information and for the transmission of data. When localising into Spanish, the character code used must be able to cater for special characters and accented characters for its adequate representation, making it easier to handle collating sequences, sorting, searching, displaying and editing of text. ISO 8859-1 (ISO Latin 1), ISO 10646 and Unicode are character codes that cater for all characters needed to write in English and Spanish and which are advisable to use to facilitate the information interchange.

**Cultural Issues**

When including icons, pictures, gestures, colours and so forth in an application’s user interface, it is advisable to make sure that they are universally recognisable and neutral, not culture-dependent. Certain images that are commonplace in the US may be meaningless, confusing or offensive for the user of another culture. For example, the “V” sign represented using two fingers. Depending on the direction of the palm, it acquires a different meaning, either victory or an insult. However, in Spain, this second meaning is lost as it simply means two. Colours usually have strong associations in different cultures. If the interpretation of a particular colour is not the one originally intended, the user may feel uncomfortable using an application. For example the expanded orange hand has a political meaning in Ulster. The use of colours in flags is a particularly delicate issue. When localising a product from English into Spanish, the use of colours is unlikely to represent a major problem; however, there are always exceptions, such as the expression in English to go green which means to go green with jealousy. This expression is meaningless in Spanish since in this culture, the colour green stands for hope.

When localising for Europe, where countries have a strong sense of sex equality, both men and women should be depicted in the same roles and gender in job titles should be avoided. This is easy for uninflected languages such as English, where job titles do not imply a particular gender. However, in other languages with more inflectional forms such as Spanish, job titles translate into male or female. The best advice is to mention both genders.

**Local Data Conventions**

It is advisable not to assume that countries that share the same language also share the same local conventions, for example GB and the US, or Spain and Mexico. The following are some of the most outstanding local data conventions that need to be taken into account when localising a product from the perspective of two locales, GB and US into Spanish with the local conventions of Spain:

The thousands separator in Spain is a <,> while for GB and the US it is a <,.>. On the contrary, the decimals separator in Spain is a <.> while for GB and the US it is a <,.>. In Spain, the currency now in use is the euro <€>, which is placed after the digits: 500 €; while in Ireland, GB, and in the US, the euro <€>, the pound <£> and the dollar <$> are respectively placed before the digits: £500, $500, and $500.

Spain, GB and US all use the Gregorian calendar. The position of the components in a date varies from country to country. For instance, in most European countries, including GB and Spain, the format of a date begins with the day of the month followed by the month and the year, i.e., dd/mm/yy. In the US, however, the month comes first, followed by the day and the year, i.e., mm/dd/yy. Nowadays, there is also a tendency to unify all date formats and place the year, with its four digits at the start, followed by the month and the day, such as 2002-02-06 or 2002/02/06. Telephone numbers, postal addresses, paper sizes and abbreviations have their conventions in Spain, GB and US. As regards units of weights and measures, Spain uses a metric system while in GB and the US it is the English system.

US, GB and Spain share the QWERTY keyboard layout; however, the layouts differ: The Spanish keyboard provides a key

### Further Reading


for the special character <ñ>, and provides functionality to write accented characters and diacritical marks by means of composed sequences. Apart from local conventions, there are local practices such as tax, salary and accounting rules that depend on each country’s regulations. Each country has its own system of taxes and economic and political rules that should be followed. Montserrat Bermúdez Bausela is a junior lecturer at the Universidad Alfonso X El Sabio (Madrid). She teaches Translation and New Technologies and also Software Localisation and is doing research in multilingual content management of software and web pages. She can be reached at mbermbau@uax.es
The world of Chinese computing consists of a plethora of character sets and character encodings, containing tens of thousands of characters. Manipulating and converting between these is a complex undertaking, far more difficult than it first appears. Many people have heard of Simplified and Traditional Chinese, but are confused by their differences and similarities. Are they used to write separate languages? Are they mutually intelligible? Are simplified characters a recent phenomenon unique to the People’s Republic of China?

A pervasive problem when dealing with Chinese text is the “translation” of documents between traditional and simplified characters. Accurately converting between Simplified and Traditional Chinese is more involved than merely mapping between character sets. Understanding the role of character variation and evolution over the 3,000-year history of written Chinese is essential to understanding the intricacies in converting between the character types. Character differences are not the only issue, however. Language use — vocabulary and dialectical differences — also becomes an issue, a fact well known to Chinese translators.

This article gives a brief history of Chinese writing and a discussion of features of written Chinese that affect accurate Chinese-to-Chinese conversion, including character and vocabulary (or lexical) variation. In a later article I will expand on this discussion and present the technical issues involved: character sets and encodings and the methods and data used to attack the Chinese-to-Chinese translation problem.

**Terminology**

To avoid confusion I refer to the various languages spoken in China (201 according to the latest release of Ethnologue) as “dialects,” following common use. However, it is important to understand that most of the languages spoken in Greater China are not dialects of each other. They lack mutual intelligibility and often have significant differences in grammar and phonetics. The more accurate term, regionalect, can be awkward for non-specialists and will not be used here.

When we talk about written Chinese, at least for our purposes here, we refer to the standard written language, which is based on the Beijing dialect of Mandarin. The other Chinese “dialects,” such as Cantonese, Shanghainese or Taiwanese, are predominantly spoken and are rarely written except in magazine articles and novels. These are problematic insofar as they contain sounds and grammatical differences that are not easily expressed in Mandarin. Chinese characters are also not the only writing system in use in China. Others such as Tibetan, Uighur, Yi and Nüshu are used by ethnic minorities or small communities and are recognized by the Chinese government to various degrees (and all but Nüshu are encoded in Unicode). These are obviously used to write different languages in the heterogeneous language environment that is China, but with the spread of Chinese characters to other parts of Asia, it is worth pointing out. We will return to the topic of representing non-Mandarin words with hanzi later on.

The phrase Greater China refers to the countries and territories in East Asia where people speak a Chinese dialect and use...
Chinese characters for written communication: the People's Republic of China, Hong Kong, Macao, Taiwan and Singapore.

For conciseness, when I refer to China, I mean the People's Republic of China or Mainland China, excluding the Hong Kong and Macao special administrative regions (SARs). From a linguistic perspective, it is convenient to treat the Republic of China, Taiwan, as a separate entity. The political differences between Taiwan and China are, for the most part, irrelevant to the topic at hand.

I will refer to Chinese characters interchangeably as hanzi or characters. Transcriptions are provided in Pinyin romanization (with tone marks) for Mandarin and modified Yale romanization (with tone numbers) for Cantonese.

Where are Traditional Chinese and Simplified Chinese used? Traditional Chinese is found in Taiwan, Hong Kong, Macao and in most overseas Chinese language newspapers. Simplified Chinese is used in Mainland China and Singapore. These divisions are by no means hard and fast. “Traditional” characters are used in China when printing classical works or when printing material for foreign audiences. The overseas edition of the People's Daily (Kènmìnrìbào 人民日报) newspaper is printed with traditional characters. “Simplified” characters are seen in Taiwan and Hong Kong as well, though never (even after Hong Kong's return to Chinese rule in 1997) in official publications.

**What’s in a Name?**

The terms Traditional Chinese and Simplified Chinese are pervasive. Unfortunately, this leads to no end of confusion since it implies that these are separate languages, which they are not, and it implies that there is a strict dichotomy between the types of script, which can lead to false assumptions:

There are always fewer strokes in a simplified character than in its traditional counterpart. If you take the definition of simplified to be GB 2312-80 (China's character set defining simplified character forms), then this is a false assumption. The “simplified” hanzi 亻 contains nine strokes, while the “traditional” variant 亻 in Big Five contains six.

There is one type of Traditional Chinese. This statement implies that the traditional character forms used in Taiwan and Hong Kong are identical, which is not always the case. For example, in Taiwan the hanzi used for the word wire, xiàn, is 線. In Hong Kong, the variant 線 is used instead. The standard traditional form used in China is also 線.

The type of script can be used to determine the “language” of the text. The language of a text cannot be reliably determined by looking at the script. Taiwan, Hong Kong and Macao all use “traditional” characters, yet Taiwan’s official language is Mandarin, while Cantonese is predominantly spoken in Hong Kong and Macao. Hence, the type of script must be separated from the notion of “language.”

Because of the overloaded nature of these terms, I will instead talk about “simple” and “complex” character forms. You will also see the terms “full-form” and “simple-form” used.

**The Chinese Writing System**

The origin of the Chinese writing system dates back almost 3,000 years. Understanding how it evolved will help understand the complex issues faced when converting between the simple and complex character forms. This section gives a necessarily brief introduction to the history of written Chinese.

**Oracle Bone** (jiágǔwén 甲骨文). Dating from the late Shang dynasty (1711–1066 BCE), these characters were inscribed on tortoise shells and ox bones which were used for divination purposes. Being carved into hard surfaces, they are characterized by thin straight strokes with few curves. The Oracle Bone characters are logographic: each character represents a word or concept. Often these were formed on pictographic principles, with the glyph looking like the entity it is meant to represent. Later characters would be used to represent words with different meanings but identical or similar pronunciations. Altogether some 4,500 individual Oracle Bone characters have been identified.
existence, contained dedicatory text, or like the Shi Qiang vessel shown above, contained genealogical information.

**Seal Script.** The Seal Scripts are first seen during the late Zhou dynasty and the Spring-Autumn and Warring States period (722–221 BCE). There were two types. The Great Seal (大篆 大篆) script was used during the early years of the Qin state and is similar in form to the bronze inscriptions. This was then greedily simplified to form the Small Seal (小篆 小篆) script. The character forms are more uniform than the Great Seal counterparts, and the incredible number of character variants found in the Bronze and Great Seal scripts were unified to a single form. China was unified in 221 BCE under Emperor Qin Shihuang (秦始皇), who undertook a series of reforms, one of which was a reform of the writing system. This was done by his prime minister Li Si (李斯), who categorized and standardized some 3,300 characters. These characters became the zhèngzì 正字 or standard form used for all official publications. Documents written in other script forms were burned.

The first systematic compilation of Chinese characters, the Shuówénjiêzì (說文解字), was compiled in 121 CE by Xu Shèn 許慎. It contains 9,353 small seal standard form characters with 1,163 variants.

**Official Script.** The official script (吏書 隸書) or clerical script (俗字 俗字) was used during the Qin 秦 and Han 漢 dynasties (206 BCE–220 CE). While the Small Seal script was the form endorsed for official documents, a shorthand form was used by the clerks and other literate people for notetaking and other unofficial communication. This script eventually became so widely used that it gained official approval, replacing the Small Seal script entirely. The official script was easier to write than the rounded Seal style characters, using straighter (though often undulating) lines, and the structure of individual characters was simplified.

**Standard/Regular Script.** The standard, model, or regular script (楷書 楷書) is a direct descendent of the official script and is essentially the character form that has been used for the last 1,800 years. It represents a further simplification of the lishù forms, further straightening the lines and departing wholly from the shapes of the seal forms. These kāishū characters also take a regular, square shape with straight and level lines. Two offshoots of lishù are seen and are important when we discuss simplification.

1) The grass characters (草字 or 草書) served as a shorthand used when writing speed was more important than legibility. Characters were formed by omitting strokes entirely or linking similar strokes and can become so abstract that only the original writer can read them after they have been written. Grass characters are often seen in Chinese calligraphy and paintings and are still used today by calligraphers.

2) The running script (行书 行书) is the handwritten form of the regular script and is both easy to write and read. It lies in between the regular script forms and the abstract grass characters in that strokes may still be connected though rarely, if ever, elided completely.

Throughout each stage in the history of written Chinese, the language has been plagued with the proliferation of variant forms. Each major step in the evolution can be viewed as a response to this, and it is a problem that exists to this day. This entropy of the writing system is one of the reasons some scholars both in and out of China have suggested that hanzi be abandoned in favor of a phonetic or alphabetic script, as was done in Vietnam and Korea. Whether this will ever happen in China is open for argument. The issues are many and complex and far beyond the scope of this article.

**Simplification Through History**

Character simplification has several dimensions. The most obvious is the reduction in complexity of a character by simplifying its shape and/or reducing the number of strokes required to it. This is not the only method of simplification, however. Reducing the number of characters used in the language is also a common simplification method. This can be done by selecting a single standard from among a number of variants or through the elimination of multiple homophones characters (characters that have the same sound) by choosing one to represent the whole. In each case the simplest form (in terms of shape or stroke count) may be chosen. However, the simplified form may not be the least complex character form; instead, it may be the more common form. Hence, the most important thing to keep in mind when dealing with simplified character forms is this: When it comes to Chinese character simplification, “simple” does not necessarily mean lower character complexity.

In other words, historic simplification must be seen as the simplification of the writing system as a whole, not just the simplification of individual characters used in the writing system. The traditional distinction between complex and simplified characters is really the distinction between characters approved for official use (the zhèngzì 正字 or “correct” characters) and everything else, the vulgar (俗字 俗字) or popular (俗體 俗體) characters used for non-official purposes. Any
character that was not considered “correct” was “vulgar,” regardless of stroke count.

Character variation is a rampant problem in the Chinese writing system (some 40% of the characters in the venerable 18th-century Kangxi dictionary are variant forms). As characters were used in different regions for different purposes at different times, variation invariably occurred. Therefore, one of the goals of writing reform through China's history has been to minimize or eliminate some of these variant (or yìzì 異體字) forms. As we saw with Emperor Qin Shihuang's draconian efforts at writing reform over two millennia ago, this is a topic important to the Chinese.

For example, the number of variations found in printed publications prompted the Chinese Ministry of Culture and the Committee for the Reform of the Chinese Language to jointly release the List of Chinese Character Forms for General Printing (Yínhù Tōngyōng Hánzì Xīnghào Biāo 印刷通用汉字字表) in early 1965. This contained 1,196 characters in the Song style with standardized stroke count and stroke order. All written publications, with the exception of certain historical works, were required to use these forms. Taiwan has produced similar tables of official character forms. These become important when we discuss character encodings.

**Modern Writing Reform in China**

Many attempts were made to develop a phonetic alphabet for Mandarin and other Chinese dialects around 1900 (though these efforts were first begun by Jesuit missionaries in the seventeenth century). The Latin and Cyrillic alphabets were proposed, as were alphabets based on the shapes of characters. One of these (zhùyīn zìmū 註音字母) was adopted and promulgated by the Ministry of Education of the Republic of China in 1918. This scheme, later known as zhùyīn fúhào 註音符號, popularly called bopomofo, is still in pedagogical use in Taiwan, but was replaced by Pinyin in China after 1958.

Attempts to simplify the writing system can be seen as early as 1930, when A Glossary of Popular Chinese Characters Since the Song and Yuan Dynasties (Sòng, Yuán Yúlài Súzǐ Pǔ 宋元以來俗字譜) was published by Liu Fu and Li Jiariu. This book cataloged 6,240 simplified forms (súzǐ 俗字) found in several publications since the Song dynasty. A further list of some 2,400 simplified characters was published in 1935 under the auspices of Qian Xuantong, who a decade earlier had proposed the replacement of the traditional character system with a phonetic system based on zhùyīn zìmū. In August 1935 The First List of Simplified Characters was promulgated by the Ministry of Education of the Nationalist government, containing 432 simplified characters. While these were used by some publications, the list was repealed in early 1936 after significant opposition from conservatives in the government.

Writing reform was a major objective of the Communist government when it came to power in 1949. The belief was that the complexity of Chinese characters was a major impediment to literacy, and that simplifying the character forms, or doing away with hanzi altogether and adopting a phonetic alphabet, would make great strides for improving the literacy rate of China and help bring China into the modern age.

However, after 1949 the desire to eliminate hanzi in China was tempered, and two approaches to language reform were started: one to alphabetize the language, the other to simplify the characters already in use. In July 1950, Mao Zedong moved the simplification effort to the fore and relegated alphabetization to secondary importance. The Hányǔ Pīn yīn 汉语拼音 system was officially promulgated in 1958, and further steps to alphabetization have ceased.

The goal of the Committee for the Reform of the Chinese Language, which was organized in 1952, was to collect and standardize the simple character forms already in common use. Liu and Li's catalog served as an important source for this effort, as did a two-volume dictionary of simplified character variants published in 1951. In December 1955 the committee published the List of First Group of Standardized Forms of Variant Characters containing 810 sets of variants, each set containing two to six hanzi each for a total of 1,865. From these, 1,053 variants were eliminated.

In January 1956 the committee released the first official list of simplified characters, entitled the Scheme for Simplifying Characters (Hánzì Jiànhuà Fāngàn 汉字简化方案). The first of its three tables contained 230 simple forms already in common use; the second contained 285 proposed simplified forms; and the third contained 64 simplified character components (piànpáng 偏旁). The latter two lists were provided for discussion and were adopted into the standard soon thereafter.

In response to confusion over the usage dictated in the 1956 scheme and to prevent the explosive creation of new simplified forms (defeating the purpose of simplification), a revision was issued in 1964, providing 2,236 simple characters replacing 2,264 complex forms. This was reissued with corrections in 1986 and is the version that continues to be used today. This list, The Complete List of Simplified Characters (Jiànhuàzì Zǒng Biǎo 简化字总表), consists of three sections:

1. 350 simple forms which cannot be used as simplified character components, such as 兒 er for 兒.
2. Simplified side components and simple form characters that can be used as side components. These are subdivided into two parts: 132 characters that can be used both as side components and as characters (the simple form 安 for 安 can also appear as a side component, as in simple form 陈 for complex 陈); and 14 side components that cannot be used alone, such as the simple form of 言, i, which can only appear as part of another character: 言 compared to 言.
3. 1,753 simple form characters derived from the components in (2).

Several methods were used for simplification for the 1964 revision:

1. Adopting existing common use súzì to represent the complex form, such as 無 for 无.
2. Coalescing homophonous characters. This can lead to significant ambiguity, though one attempts to minimize this by using characters with great semantic variation to provide a contextual clue to the correct meaning. For example, the complex character 習, meaning behind, was simplified to

44 Localization Reader 2003-2004
the homophonous 后 后, which means queen. It is worth noting that there is evidence that 后 was being used for 后 in classical literature.

3) Using part of a complex character to represent the whole, such as 电 for 電.

4) Adopting an existing grass-style character as the standard form, such as 门 for 闩.

5) Using a new, less complex character such as 万 for 萬. Note that the new characters may actually have existed for millennia, but were adopted as the simple forms of complex characters.

Further efforts at simplification were made in the 1970s, but for various reasons these failed to gain currency and were withdrawn. These simplifications were much more radical than those proposed previously and did not have the linguistic foundations shared by the previous efforts.

The definition of simplified side components and hence a set of simplified radicals has made dictionary use even more complicated. The standard set of 214 Kangxi radicals used by most complex-form dictionaries (such as those found in Taiwan and Hong Kong) is different from that used by many simple-form dictionaries, which use the 189-radical system developed by the Chinese Academy of Social Sciences. This requires people working with simple and complex form dictionaries to know two radical sets or to use a table to map between the two.

Has simplification worked? While statistics must be interpreted with caution, looking at literacy rates in Greater China can be telling. Taiwan, Macao, Hong Kong and Singapore have literacy rates (defined to be the percentage of the population over the age of 15 who can read and write) of 95%, 90%, 92% and 94% respectively. Compare these to an 82% literacy rate in China. Even with the population and sociological differences between these countries, character simplification has not been the panacea to China’s literacy rate that intellectuals believed. Statistics comparing literacy rate versus age in different countries indicate that it takes longer for students learning hanzi to achieve the same literacy level as their counterparts using alphabetic scripts. And, of course, generations who are taught to read and write the simple character forms face difficulties when they set out to read classical literature (or indeed anything published before the mid-1950s) or documents originating outside of China. In essence, to be fully literate in the modern world you must learn the simple and complex forms.

When it comes down to it, the success or failure of simplification is irrelevant. We have to deal with the fact that there are different writing systems in use now for Chinese, and those of us developing software or writing documentation targeted at the growing Chinese market need to be aware of and handle these differences. In my second article I will discuss the way Chinese text is represented on the computer and the issues faced when attempting to convert between the regions in Greater China. As you can see, this is much more than merely mapping individual characters from character set to character set. Usage conventions and vocabulary differences need to be accounted for, not to mention the confusing overlap of character repertoires found in the multitudinous character encodings in active use through Greater China.

For Further Information

The best general overview of Chinese writing in English is Modern Chinese Characters by Yin Binyong and John S. Rohsenow (Sinolingua). The authors cover the gamut, from the origins of Chinese written expression through the methods of character formation, semantics and even how they are written. Yin was a member of the Committee for the Reform of the Chinese Language from 1956 through 1979. His insight into the history of modern Chinese is invaluable.

Ping Chen’s Modern Chinese: History and Sociolinguistics (Cambridge University Press) describes the history and social factors in language use within Greater China, including the use of hanzi in writing various dialects.

For an alternative view on the use of hanzi, not only in Chinese but throughout Northeast and Southeast Asia, William Hannas’ Asia’s Orthographic Dilemma (University of Hawai’i Press) lays out the thesis that the continued use of hanzi, especially in this technological age, is an anachronism. For a somewhat more balanced perspective, though still with an anti-hanzi perspective, John DeFrancis’ The Chinese Language: Fact and Fantasy (University of Hawai’i Press) is an approachable read.

The women’s script, Nushu 女书, is very interesting. Used by a small group of women in Southern Hunan province for a very long time (the exact origins of the script are still a mystery), its study presents a fascinating view into the evolution of a written language and the social aspects. William Chiang’s We Two Know the Script; We Have Become Good Friends: Linguistic and Social Aspects of The Women’s Script Literacy in Southern Hunan, China is one of the few texts in English to describe this interesting writing system.

In a paper for the 18th International Unicode Conference, “The Extremes of Typographic Complexity: Character Set Issues Relating to Computerization of The Eastern Han Chinese Lexicon 誠文漢字” Shuxuwenjizı, Richard Cook presents the history of this influential dictionary and describes issues he faced while computerizing the small-seal characters defined in it. See www.unicode.org/iuc/iuc18/papers/b14.pdf

Readers interested in Literary Chinese (the written vernacular starting during the Warring States period of the Zhou dynasty) should consult Michael Fuller’s An Introduction to Literary Chinese.
Technical terminology is not the only thing to worry about. Foreign names are a major source of ambiguity when translating documents. When writing a foreign name, the Chinese will use hanzi based solely on their phonetic value. This makes it difficult to recognize foreign names when processing Chinese text (since you are looking at characters used for their sounds, not their meaning), but translating them is more difficult. Each region in Greater China uses different conventions. For example, the name Bush is written differently in China, Taiwan and Hong Kong: Bùshén 布什, Bùxì 布希 and Bou³-syu⁴ 布殊 respectively. And after the September 11 attacks, no fewer than 11 different transliterations of bin Laden could be found in various Chinese newspapers around the world.

Common, everyday words are problematic as well. The word used for taxi is different in Hong Kong, Taiwan and China: dik1si6 的士, jìchéngchē 計程車 and chǔzū qìchē 出租汽车 respectively. In Hong Kong, Cheut¹jou¹ Hei²che¹ 出租汽車 (the complex form of chǔzū qìchē 出租汽车) refers to a rental car, not a taxi — a very different concept. Therefore, it is important to not mix these up when converting documents.

I hope that this article has given you a feel for the complexity faced when looking at the Chinese writing system. In the next part I will tie much of this information together, showing how you can account for these variables when writing (mostly) automatic translation software for Chinese-to-Chinese conversion. Until next time,再见。

Thomas Emerson is a software architect at Basis Technology. He can be reached at tree@basistech.com
.NET Internationalization: RegionInfo and TextInfo Classes

Bill Hall

RegionInfo and TextInfo are two small, informative classes belonging to the Microsoft .NET System.Globalization namespace. RegionInfo, as the name suggests, provides language-independent information about world regions, and TextInfo contains information about the writing system of a .NET CultureInfo object. In this article we will examine the properties, methods and uses of these two classes.

RegionInfo

RegionInfo provides the information about a particular world region that is independent of any of its languages. Whereas a .NET CultureInfo instance can be created from an RFC-1766 identifier specifying language and region (en-US, ar-SA, ja-JP) or just the language component itself (en, ar, ja), RegionInfo depends upon the two-letter ISO-3166 country (region) identifier (US, SA, JP) as its constructor string. As a result, the object created is limited to information that applies uniformly across the region, even if many languages are used there. For example, Switzerland has four official languages, but whether the favored language is German, Italian, French or Romansh, everyone there uses the Swiss Franc SFr. as the local currency symbol.

You can also use a Windows LCID as a constructor. Thus, either of these two expressions

RegionInfo r = new RegionInfo(“US”)
RegionInfo r = new RegionInfo(0x0409)

creates a RegionInfo object for the United States.

Once created, the properties in the table “RegionInfo Properties” can be examined.

The accompanying table shows the output for Russia; the execution line is RegionInfo RU. If you forget the identifier (in this example RU), a default for the current thread is used and a short help line is provided. By the way, if you get a ? mark for

<table>
<thead>
<tr>
<th>RegionInfo Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrencySymbol</td>
</tr>
<tr>
<td>CurrentRegion</td>
</tr>
<tr>
<td>DisplayName</td>
</tr>
<tr>
<td>EnglishName</td>
</tr>
<tr>
<td>IsMetric</td>
</tr>
<tr>
<td>ISOCurrencySymbol</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>ThreeLetterISORegionName</td>
</tr>
<tr>
<td>ThreeLetterWindowsRegionName</td>
</tr>
<tr>
<td>TwoLetterISORegionName</td>
</tr>
</tbody>
</table>

Currency symbol associated with the country/region.
RegionInfo that represents the country/region used by the current thread.
Full name of the country/region in the language of the localized version of the .NET Framework.
Full name of the country/region in English.
Boolean expression indicating whether the country/region uses the metric system for measurements.
Three-character ISO 4217 currency symbol associated with the country/region.
Two-letter code defined in ISO 3166 for the country/region.
Three-letter code defined in ISO 3166 for the country/region.
Three-letter code assigned by Windows to the country/region represented by this RegionInfo.
Two-letter code defined in ISO 3166 for the country/region.
A WINDOWS VERSION

Writing a Windows .NET demonstration program requires providing the user with a list of available regions. However, RegionInfo has no convenient method API that provides such a list, but you can leverage the output provided by the static CultureInfo.GetCultures method provided duplications are removed. For example, on my .NET installation, I can find several Indian locales — Gujarati, Hindi, Kannada, Konkani, Marathi, Punjabi, Sanskrit, Tamil, Telugu and so on — all sharing the same regional information. The trick is to enumerate specific cultures, extract the country identifier from the RFC-1766 identifier to create a RegionInfo object and then check to see if it is already in the list before inserting it. Specifically,

```csharp
CultureInfo [] ci = CultureInfo.GetCultures (CultureTypes.SpecificCultures);
foreach (CultureInfo c in ci) {
    try {
        string [] names = c.Name.Split (-`-`);
        MyRegionInfo r = new MyRegionInfo(names[1]);
        if (!cbRegions.Items.Contains(r))
            cbRegions.Items.Add(r);
    } catch {
    }
}
```

In this example we are counting on an RFC-1766 identifier having the form xx-YY, possibly followed by variant information. In some beta releases of .NET, such variants were prefixes, contrary to the RFC-1766 specification. For example, in .NET Beta 2 identifiers were found such as Cy-az-AZ for Azerbaijan using Cyrillic script. With .NET versions 1.0,
that is no longer the case; for Azerbaijan you will find az-AZ-Cyr and az-AZ-Latin for the two scripts in use there.

Note that we use CultureTypesSpecificCultures in the code above since these are the only ones that contain region specific items. (Makes you wonder why RegionInfo is not a property of CultureInfo, doesn’t it?)

An accompanying screen capture shows the Windows demonstration program for India. The program with source code can be downloaded from www.mlmassociates.cc or www.csharpi18n.net.

TextInfo

TextInfo has information about the writing system of a given .NET CultureInfo object. Unlike RegionInfo, TextInfo is a property of CultureInfo. So, if you want one, you ask for it as shown below rather than use the “new” operator.

```
// Assume we already have an instance (ci) of CultureInfo.  
TextInfo ti = ci.TextInfo;
```

See the accompanying table for a description of the main properties and methods of TextInfo. Note: As far as I have been able to determine, no ANSI code page is actually an ANSI standard. The use of the word ANSI (along with OEM for Original Equipment Manufacturer, to describe console code pages) is a part of Win32 nomenclature that has caused enormous confusion to developers (and everyone else) since the early days of Windows.

The ToLower and ToUpper method are overloaded and can be applied to characters as well as strings.

The properties given above are a carryover from Win32, where they were accessible by calling GetLocaleInfo for a given LCTYPE. Indeed, CultureInfo along with its various properties and contained classes in a crude sense represent a wrap-up of the Win32 locale model including the LCID and all the APIs that depended on it, although the .NET realization is much nicer, more complete and easier to use.

The code page properties shown above played an important default role in Win32 (and probably will continue to do so in .NET) by providing default conversions to and from Unicode in lieu of any other information. For example, suppose I ask a non-Unicode-enabled program, say one written to run on Windows 98, for a piece of data that is normally stored internally on the system in Unicode (permanent locale data is such an example — it is stored in a Unicode file locale.nls). When the information is returned, it must be converted to the local code page of the process, typically the Windows system default code page. In most cases, the result is satisfactory, but occasionally the automatic conversion has to be overridden. Such problems arise much more rarely in .NET since the system is cleanly Unicode throughout.

The methods in TextInfo, namely ToLower, ToUpper and ToTitleCase, are affected by CultureInfo but in a very limited sense. As far as I can determine, only the TextInfo object obtained from a Turkish or Azeri (Latin) CultureInfo object is affected. In this case, the dotted lowercase i is mapped to and from its uppercase dotted form, and the undotted form maps to and from its uppercase undotted form I.

TextInfo is of no value in monocasing French, Spanish or Italian in the traditional way because of loss of accent marks in upper case. Rather the monocasing methods of TextInfo mostly follow Unicode standards. Unicode specifies case formats for all characters where case is relevant except for the undotted i, and in this instance .NET (and Win32) fills in when the language is Turkish or a related language.
Further Reading

An understanding of CultureInfo is helpful when working with any member of the .NET System.Globalization namespace. You may want to read the following article for more information:


Related articles include:


The Program

The TextInfo Demo screen capture shows the TextInfo demonstration program in action. You can select a particular culture, and you can type in a string to see the effect of the uppercase, lowercase and titlecase mappings. You can download it and the source code from www.mlmassociates.cc or from www.csharpi18n.net

Bill Hall is a globalization consultant, a teacher and a speaker on internationalization and a member of the MultiLingual Computing & Technology editorial board. He can be reached at billhall@mlmassoc.com
Hiring software quality engineers can be a challenging task, arguably harder than hiring software developers. For developers, job requirements can be easily identified based on the programming language and technology in use at the hiring company. For quality assurance (QA) engineers, however, job requirements can be less specific and more far-ranging, especially with regard to internationalization.

What makes a good software quality engineer? Unlike developers, for example, a tester for a Java program need not necessarily have in-depth coding or operating system administration experience. Attention to detail is a good quality for manual testing, but it is probably not necessary for automated testing. For internationalization and localization testing, defining requirements becomes more complex. Is knowledge of foreign languages an absolute requirement? Is technological background a necessary condition? Or are both equally important?

**INTERNATIONALIZATION QA TEAM**

Before building a team for internationalization testing, consider the nature of the international business. Although there is no single rule to follow to structure a team, some options are more suitable than others. In building a team, the first step is to differentiate between two types of testing.

Internationalization testing is functionality testing performed on the base product using supported encoding, possibly including Unicode. The purpose is to verify that the code handles the locale and characters of the supported languages. This testing must be done on the base product before localization.

Localization testing is functional and linguistic testing performed on each localized version of the product. The purpose is to verify that the functionality works after the product has been localized, as well as to verify the linguistic quality of the translation.

Normally, internationalization testing must be completed before localization, except in the case of simultaneous translation when localization occurs in parallel with software development of the base product. Depending on the international business strategy, the company may have different testing needs. For example, many American companies choose not to deliver localized products for every foreign country where they sell their products because small markets with little revenue do not justify localization expenses. Others rely on in-country partners taking care of all aspects of localization. In this case, only internationalization testing is required.

One option for creating an internationalization team is to consider internationalization to be part of the base product and to make testing internationalization features a shared responsibility of the entire QA group. In this scenario, QA engineers have guidelines that help them to write procedures that test locale and encoding and to include a section for internationalization testing in their test plans. Although no dedicated internationalization group is required, each tester must understand locale and encoding and assume responsibility for testing the international features of his or her assigned component.

If automated scripts are available, they should be data driven so they can run with data of each supported language. Release engineering can produce two parallel builds, one for the base product and another with a pseudo-translation of the user interface to verify that the product can be localized. This procedure is especially effective when partners do translation because problems can be found and fixed in advance, thus decreasing required support.
Testing by the overall QA group assures maximum test coverage because each tester knows the operation of his or her component best. And because it does not require dedicated internationalization resources, it is less expensive. On the flip side, combined internationalization testing can slow down testing of the base product, which is normally a higher priority for companies whose largest share of revenue comes from the domestic market. Because each quality engineer must have the uncommon knowledge of encoding, locales and double-byte input, additional training is likely required.

Another option is to create a dedicated internationalization testing group in addition to a group for regular testing. For internationalization testing, quality engineers develop specific internationalization test plans and test procedures, rather than include them as part of base-product feature testing.

Independent internationalization testing results in quicker testing without penalizing functional tests for the base product. Also, it can ensure that a larger number of encodings in different localized platforms and software environments get tested. Liabilities of dedicated testing include increased costs, duplication of test procedures and inability to cover functionality in as much depth if testers are responsible for multiple components.

Alternatively you can combine the previous two options: 1) have the regular QA group test common encoding and locale using data from different character sets while performing the functional testing; and 2) have a smaller, dedicated internationalization group test the more complex encoding, typically the double-byte languages, and perform other internationalization-specific testing such as pseudo-translation or locale-switching for multilingual products.

This provides the advantages of the first two options. On the downside, it requires coordination between the two groups and sharing the same hardware.

How do you choose the best option? To summarize, if your software is not localized (or localized in only a few languages) and does not support many encodings, you may want to incorporate internationalization testing into regular functional testing. If your software supports several languages, especially the double-byte languages, dedicated internationalization testing may be appropriate. A combination of international testing shared with functional testing and with a dedicated international group is most efficient when your software is multilingual and based on Unicode.

### Localization Testing

The first step is to differentiate between two types of testing: functional and linguistic. The two types of testing require different skills. Linguistic testing requires a good knowledge of the language, whereas functional testing requires knowledge of the product and technology. The second step is to identify the level of localization, user interface translation only, or additional software modules for a specific region, such as taxation for e-commerce.

A first option is a localization quality group with the responsibility for both linguistic and functional testing. Bilingual engineers may be required.

Functional testing can be performed faster when testers have knowledge of the specific language. Linguistic testing is more accurate.

Unfortunately, persons with knowledge of software QA and foreign languages are difficult to find. A more expensive alternative is to have a quality engineer for functional testing and one native speaker for linguistic testing. Native speakers need support when technical problems block their testing. Even in the case of one qualified person performing both kinds of testing, the situation is not ideal because the focus on the text in each screen is different than the focus on the functionality behind the screen.

This option is preferable when the size of the translation is not too big and there are additional software modules. This option is also probably more efficient in the case of simultaneous translation and software development.

### Internationalization QA Team Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Advantages</th>
<th>Disadvantages</th>
<th>Best when</th>
</tr>
</thead>
<tbody>
<tr>
<td>No dedicated international</td>
<td>More functional test coverage. No additional costs.</td>
<td>Penalizes the base product testing. Requires more training.</td>
<td>No localization. Few supported languages and encoding.</td>
</tr>
<tr>
<td>Dedicated group.</td>
<td>Speeds the testing up. More encoding and platforms coverage.</td>
<td>More costs. More work on test plans and test procedures. Less component testing coverage.</td>
<td>Many languages, including double-byte.</td>
</tr>
<tr>
<td>Testing divided between the</td>
<td>More functional test coverage. Less additional costs. Speeds the testing up. More encoding and platforms coverage.</td>
<td>More coordination needed. Possible increases of hardware costs.</td>
<td>Unicode and multilingual applications.</td>
</tr>
<tr>
<td>regular quality group and</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a small dedicated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>internationalization group.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Localization QA Team Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Advantages</th>
<th>Disadvantages</th>
<th>Best when</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedicated group for both linguistic and</td>
<td>Quick functional testing. More accurate linguistic testing.</td>
<td>Difficulty to find resources, incurring additional costs.</td>
<td>Small size of translation projects. Parallel translation and software development. Localized products include additional software modules.</td>
</tr>
<tr>
<td>functional testing.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional testing</td>
<td>No need to hire native speakers. Faster testing. Leverage with the internationalization testing resources.</td>
<td>Need of detailed test procedures for linguistic testing. Need to support linguists.</td>
<td>Partnership with localization companies. Several languages.</td>
</tr>
<tr>
<td>in-house and linguistic testing outsourced.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Both functional and linguistic testing</td>
<td>Neither internal costs nor hiring process. Faster turnaround.</td>
<td>More support for the external resources. Less test coverage, probably not high software quality.</td>
<td>Less complex software products. Mostly based on user interface.</td>
</tr>
<tr>
<td>outsourced.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In a second option, the quality engineers perform the functional testing only, and the linguistic testing is outsourced to independent translators or a localization agency.

The advantage is that there is no need to hire native speakers or search for bilingual quality engineers, and this makes the hiring process simpler. The turnaround time is fast since the two kinds of testing can be done in parallel. Resources can be leveraged if the same group can perform internationalization and functional localization testing. In this case, the linguistic testers, who lack product knowledge, would need detailed test procedures and probably a good amount of engineering support.

When a company has a good working relationship with a localization company that has successfully completed several projects, this option saves time and money in the long run because the localization company has knowledge of the product. This option also ensures a faster turnaround for simultaneous translation of multiple languages.

Another option is when both functional testing and linguistic testing are outsourced. Localization companies or in-country partners have the responsibility of functional testing in addition to the translation. In this option, no internal resources are required, so the project can turn around faster.

Disadvantages include additional technical support required for external resources, additional coordination and the need for a program manager on the software company side. Translators and testers require training on the product that may need to be repeated for subsequent projects. The functional testing may lack extensive coverage.

If localization consists only of translation of the user interface, if the product is intended for final users, if testing relies mostly on the user interface and if the underlying technology is not too complex, then this option is the best one.

**Hiring Process**

One common mistake made by many companies is to hire native speakers for internationalization and localization testing who do not have any technical background or QA experience. Language knowledge is not enough. Programs can be complex, and their functionality is not always accessible from an easy user interface. Too often, translators who are hired for functional testing find themselves unprepared for the task. The result is poor quality and incomplete test coverage.

For internationalization testing, knowledge of system locale and how the different operating systems or programming languages handle character sets is essential. Being a native speaker of a foreign language does not guarantee this knowledge. For linguistic testing, especially, a technical background is helpful to be able to quickly go through the functionality that brings up the screens.

In hiring internationalization testers, some level of technical knowledge should be set as a requirement, as well as knowledge of QA methodologies. Testers must be able to write test plans and procedures and to understand different test types and cycles to ensure a high-quality product. Hiring managers are often frustrated because of the lack of native speakers who have language knowledge and technical knowledge. In this case, good training is essential.

With good training, translators who are interested in gaining technical experience can become successful quality engineers. For this audience the training concentrates on the QA methodologies that do not differ from the base code testing and that are essential for a good test coverage and good quality product. In addition, some peculiar aspect of localization testing must be considered, such as to identify the critical areas where the translation is most likely to break functionality. When native speakers or people who are interested in learning new languages can be found among QA engineers, training can be focused on learning encoding and locale settings.

Building a successful internationalization team is difficult but not impossible. The key is to find motivated people and to provide good training for translators who want to further their technical background and for QA engineers who want to learn an important and growing aspect of software development. For everybody, internationalization testing can be fun.

Gabriella Spatolisano is director of QA and globalization at ATG. She can be reached at spatolisano@atg.com

---

**Gabriella Spatolisano is director of QA and globalization at ATG. She can be reached at spatolisano@atg.com**