[*Machine Translation Today: Translating and the Computer 15*. Papers at a conference…18-19 November 1993 (London: Aslib)]

Future Translation Workbenches: Some Essential Requirements

Richard E. Birch
Customer Business Services, Rank Xerox Technical Centre

This paper proposes a number of essential requirements intended to provide direction for translation workbenches of the future. The points made arise from a consideration of the problems and frustrations encountered during several years experience in the use of proprietary and in-house translation tools. The paper will also suggest innovations which may considerably improve the productivity and flexibility of future translation workbenches.

Introduction

It is only by keeping translation software attractive to the increasingly demanding business world that computerised translation will move forward. Approaching the issue from the point of view of industry, this paper will attempt to summarise, albeit briefly, a few essential aspects of translation workbench design which I feel provide important directions for the future.

The points made arise from a consideration of the problems and frustrations encountered during several years experience in the use of proprietary and in-house translation tools at Rank Xerox. I will also suggest a couple of ideas which, to the best of my knowledge, are innovative and could lead to improved productivity and flexibility in future translation workbenches.

Why are we using translation tools?

It is worth briefly reminding ourselves, first, *why* we use computers for translation in industry, and *what* we use them for. The hope is that computerisation will eliminate slow or complex manual tasks in order to reduce translation costs and time to market. At the same time, translation software should effectively maintain or improve quality of translation so as to increase customer satisfaction.

An important point, to which we will return later, is that, in order to achieve faster time to market, companies are increasingly looking at ways of performing most of the translation on early versions of the text, and then translating small incremental updates nearer the product launch date. Translation tools have an important role to play here in reducing the bulk of update translations.

Overview of the translation process

Let me set the scene for what I will say later by drawing a simple model of the translation process. Figure 1 shows, very schematically, the basic elements of most translation processes. At this level, these stages are largely the same for translation of both documents and software-based text.

Note that translation software is required to assist in all the activities outlined in Figure 1. The

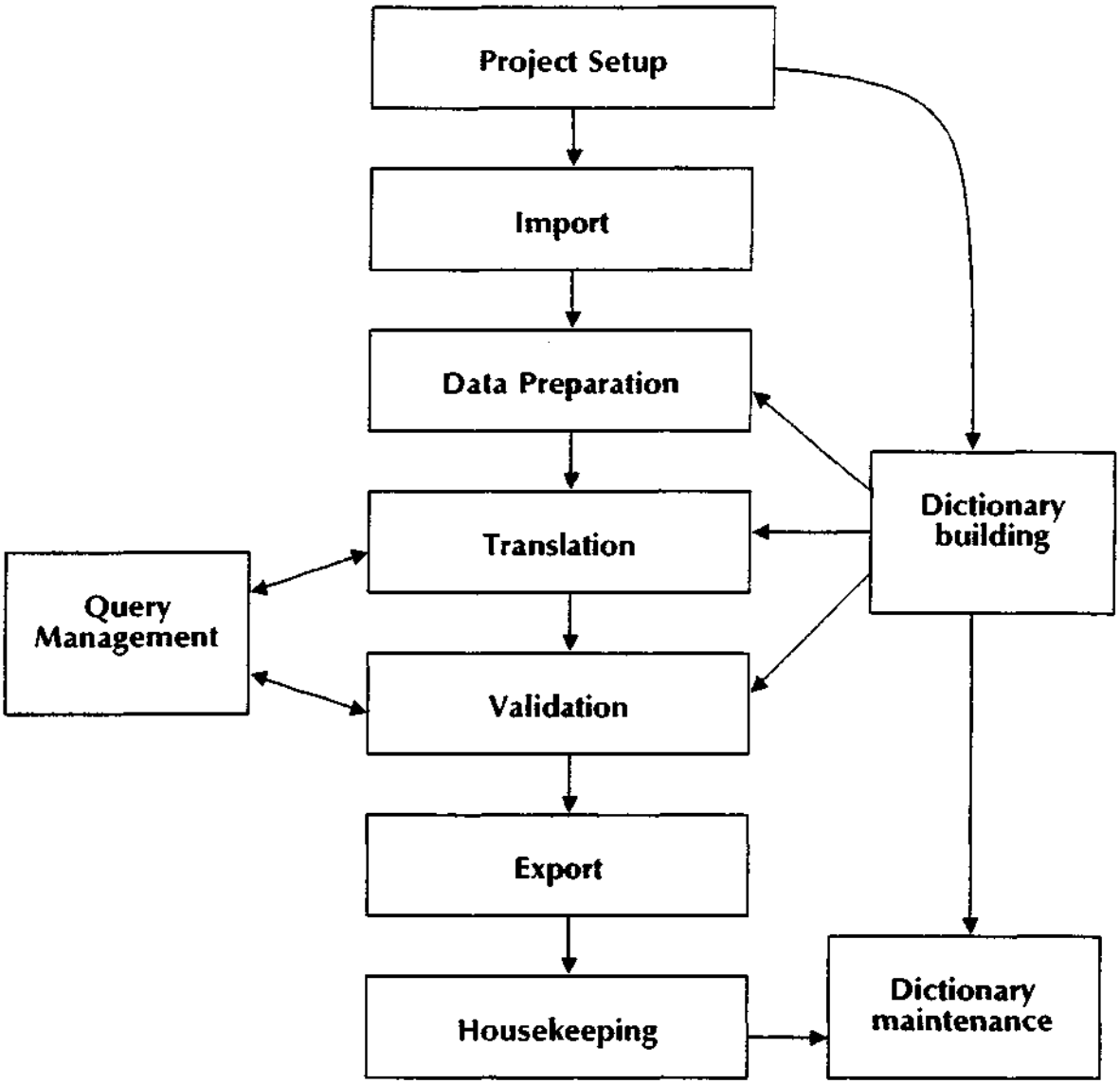

Figure 1: Translation process overview.

translation workbench is far more than just an editing environment. It is needed for administrative activities long before and after the translator actually sits down to edit text.

Project setup involves the creation of directory structures and the completion of other necessary administrative activities prior to the importing of data.

Dictionary building normally takes place in advance of translation, and involves running software which can help identify unknown words or phrases in a corpus, obtaining validated translations for those terms, and preparing dictionaries for access during translation.

During the **import** stage, the data is transferred from the outside world into the translation environment. At this point the data may be converted and tested for conformance to agreed formats. Segmentation of data into manageable chunks may take place at this stage or during data preparation.

During **data preparation** the data is prepared for release to the translator. Processing at this point may include such things as source matching[1], machine translation and change analysis[2], as well as the preparation of reference information such as dictionary items and notes from designers.

Once the material has been prepared, it is handed over to the translator, who will edit or supply the **translation.**

During translation, the translator will usually need to ask questions about the meaning of the source text, ask about appropriate translations for specific items in the source, request more expansion space, etc. These queries need to be dealt with as quickly and efficiently as possible. This is the function of the **query management** subsystem.

After editing, the material must be proofread and then **validated** on behalf of the customer before being prepared for publication or loaded into the software.

The **export** stage removes the data from the translation environment and puts it into the form required for return to the customer.

During the **housekeeping** stage, backups, archives and other administrative activities take place. The main dictionary database also needs to be updated in the light of changes or additions made during translation or validation.

Let us now consider some essential requirements for developing future translation software.

Aim for a generic translation environment

Any large translation department has to deal with translation text from a wide and continually growing set of product environments. Each of these product environments organises and stores its data in different ways.

If tools are written specifically to translate products from a particular product environment, it is usually difficult, sometimes impossible, to successfully adapt them for use later with completely new product environments with which the translation department may have to deal.

Developing one-off solutions for the translation of differing environments, having to significantly adapt tools each time, or using a number of different translation tools can soak up a great deal of time. money and resources:

1. New processes and tools must be developed and tested each time. This can place a serious strain on resources and schedules.
2. Users of the translation tools have to constantly switch between new environments. This creates a very high retraining cost, and also affects the productivity of the user, who can become confused by the differences between each process and environment.
3. The processes and tools used for translation have a short life and never reach a high level of stability and robustness. This creates significant costs in troubleshooting, downtime, support and rework - over and over again.

1. 'Source matching' is a Xerox term corresponding to 'repetitions processing', 'translation memory' or 'example-based translation'. A translated string retrieved via source matching is referred to as a 'recall'.
2. 'Change analysis' is a means of detecting unchanged text during updates and automatically transferring the appropriate translation into the target text. It will be dealt with in more detail below.

Here are some suggested solutions to these problems:

1. Design a single translation system capable of handling translation of software text from any product, and documentation from any publishing system. This should considerably reduce development, testing, support and retraining costs (both financial and resource) and thus considerably reduce the basic translation costs. In addition, and significantly, since the same tools will be reused for all products, the tools will become stable, and that stability will bring further dividends in terms of reduced troubleshooting and support requirements.
2. Specify standardised data interfaces for documents, extracted software text and simulators. This is a key enabler for the previous point. This means that data is always passed between product team and translation in the same format. Also, standard interfaces to simulators (see below) need to be developed and implemented, so that data and commands can be passed between them and the translation environment.
3. Standardise and integrate, wherever possible, interfaces and functionality provided for translation of documentation- and software- based text. Software and documentation translation can differ in a number of (sometimes significant) ways. Nevertheless, wherever common functionality can be found it should be used for both. In particular, such things as editor functionality should be standardised. (Indeed, it may be helpful to make translation editor functionality mimic other editing environments used extensively for other activities in the office, such as for writing mailnotes, reports, etc.. since this will reduce the interference factor for the translator. This would mean allowing for a choice of look and feel for the translation editor.)
4. Consider the selection of a standard publishing environment in which to publish translations, and develop documentation in this format or develop programs to convert other publishing systems into this format. (After conversion, checking will need to take place in order to ascertain that all features in the native environment have converted correctly, and to make changes if necessary. At this point, it would be possible to test for translatability issues and appropriate changes to enable translation (eg. increase line height for Japanese, or base line height for Arabic). This could be an effective way of reducing off-standard costs - bear in mind that off-standard costs are otherwise multiplied by the number of languages in which translators experience difficulties!)

Allow flexibility between batch and interactive translation

Machine translation is best suited for simple, sublanguage texts where information is explicit. For such texts, machine translation can give important productivity gains. As your source text moves from simple text, such as parts lists or service manuals, to such things as training manuals or marketing brochures, where stylistic variation, reliance on context and language complexity become relatively more important, the productivity of machine translation is rapidly eroded by the increasing need to post-edit.

A large number of companies deal with both simple and complex texts. For this reason an ideal translation workbench should allow an organisation to translate using machine translation *and/or* interactive translation, wherever each is most appropriate. This may not mean designing your own machine translation system as part of your workbench. You may be able to simply provide filters and links to access an existing system.

Create user-defined systems

The translation tools will need to be flexible enough to allow a slightly different approach where required, be it for technical or organisational reasons or simply user preference. It should be possible to configure the translation workbench to meet the needs of the organisation's ideal work process. The workbench should not dictate or restrict the choice of work process.

As much information as possible about how the system works should be user-defined, rather than hard-coded. It should be possible to modify the user's or organisation's preferences by the use of definition files or property sheets which are easy to understand and modify. For example, users should be able to tell the system how to segment text, sift non-translatables. assign statuses, present items in the reference area, skip through the source text, etc. etc. The translation tools should include a workflow management system which is easily adapted for different types of organisation.

Enable fast and simple administration

With a number of existing systems there can be high costs in terms of time and resource for administrative activities such as importing and exporting of files, data preparation, dictionary management and dictionary building. In fact, this is a key problem for the use of computerised translation tools in general. Only companies which can afford to carry these overheads can invest in the translation tools which will bring them greater productivity.

Additionally, there is a cutoff point in the size of translation jobs, below which it is deemed more cost-effective to translate manually because of the administration costs. (Manual translation of a small update to a product can then become problematic because all the information needed for source matching and change analysis at the time of the next revision is lost. Manual translation also makes it harder to maintain translation quality in terms of consistency and validated terminology.)

Translators' workbenches need to reduce administrative activities to the minimum in all aspects of the translation work. It must be possible to import, prepare, export and manage jobs quickly enough that the size of the job makes no difference. One way to achieve this is to reduce as far as possible the number of points at which the user has to intervene in order to run the software. However, I also wish to make an innovative suggestion here. It involves the concept of the 'job profile'.

The job profile is a collection of definition files and parameters which define how the translation processes work, and the work flow for a given job or collection of jobs. There should be a single, simple interface for the user which groups together all the information for viewing or modification.

Amongst other things, the job profile should define:

- rules for segmentation of text during data import
- target languages for the data preparation stage (this will automatically create directories and target files for all languages simultaneously)
- whether or not source matching, change analysis or machine translation should be used, and if so, what databases should be used for comparison, how information should be ranked for presentation to the user, what text, if any, should be automatically inserted into the target file, etc.
- rules for the filtering of non-translatables

- what stages the jobs will move through during translation, and what criteria must be satisfied to move from one stage to another
- what wrapping rules and default editing parameters are appropriate for the current job.

Restricted access rights should be associated with certain types of information, and certain defaults should also be available. If product specific changes have to be made, only the appropriate parts of the profile should be modified. It should be easy to access and change such information.

One of the key benefits of the job profile is that it can be easily copied from project to project with minimal changes (often no change). When dealing with similar types of text, the use of user profiles can thus reduce the administrative setup time considerably - once the data has been imported, the whole process of data preparation for any number of target languages could be initiated by a single click on *a* button, since all the information needed by the system is already contained in the user profile.

Enable decentralised translation

Especially where new markets are opening quickly, translation departments are likely to need to do the translation itself in remote locations, while administering and processing the data from a central location. The translation system of the future must enable this interaction between the remote translator and the central hub. The platform on which the editing software is based should be easily obtainable, cheap and portable. It must also be robust, and allow for simple installation and remote troubleshooting, while providing as much useful functionality as possible to the translator.

Design truly multinational software

Current ALPS and Systran systems as used by Xerox are unable to cope with requirements to translate into languages beyond the basic Western European set. We already translate into East European languages, including non-Latin character sets such as Russian and Greek. Future requirements include Middle Eastern and Far Eastern languages, where character shape and text direction are far more complicated than those found in Western European languages. Indeed, there seems to be a significant, general growth in the importance of non-European markets for the software industry.

While designing translation environments, care must be taken to avoid locking the user into a restricted set of languages. This means doing the following for the editor:

1. enabling appropriate character code storage for languages such as Japanese, Chinese and Korean, or for documents containing a mixture of languages, for which 8-bit character sets were never really suitable.
2. providing modules to render non-Latin characters appropriately if they are context sensitive (eg. Arabic) or make extensive use of ligatures (eg. Arabic- and Indian-based and South East Asian languages).
3. enabling more complex placement of diacritics or vowel signs than that found in European languages.
4. enabling bidirectional text input and editing for languages such as Arabic and Hebrew.
5. enabling intelligent algorithms to improve the efficiency of input for languages such as Japanese, Korean and Chinese.

Other factors also need to be borne in mind. For example, algorithms which parse segments for dictionary lookup must not be hindered by the fact that languages such as Japanese and Thai do not separate words with spaces, as we do in English.

Improve translation context

The translator needs to understand the context of a sentence or word they are translating for two fundamental reasons:

1. To understand clearly the meaning of the text and its relationship to surrounding text
2. To verify that the translation fits in with space and formatting constraints and the surrounding text.

The translator needs to see text with as much contextual information as possible. Presenting text in paragraphs, rather than segment by segment, is a good first step towards improving contextual information.

For translation of software messages, Rank Xerox has used a system for several years which shows the text in the editor within the display box into which it must fit. Given the lack of expansion space typically provided for foreign languages by English-speaking UI designers, this is an invaluable tool. It means that the translator can immediately tell if the translation doesn't fit. and can try an alternative translation or abbreviation. This avoids an extremely long-winded, cyclical process of translating, loading up software, retranslating, loading up again, and so on.

Rank Xerox also has a simulator of the user interface, linked online into the translation editing environment. The simulator automatically keeps in step with the message shown in the editing environment. This is invaluable for dealing with things such as adjectival agreement (eg. the word 'enabled' appearing on its own must be translated differently in some languages depending upon the gender or number of the text it qualifies).

Facilitate the sharing of data

In most systems, translators currently work in relative isolation. There is no simple mechanism for immediate sharing of useful information.

For example, if a translator makes a change to a dictionary, that change should be communicated immediately and automatically to other translators dealing with the same target language (but *not* those dealing with other languages).

There should also be an automatic way of providing documentation translators with the appropriate translation for a screen-based icon which was translated previously.

Similarly, query management systems are often unwieldy and labour intensive, and comprehension queries tend to be raised many times over. In an ideal world, translators would immediately know whether a comprehension query had already been raised about a particular piece of source text, and would be able to subscribe themselves to that query for as long as they felt they needed to know the answer. Queries should travel quickly and intelligently across networks or modems to and from defined addressees.

Allow for future enhancements

Software must be written in such a way that future enhancements can be easily added without major rework. The aim is to avoid having to develop or find a new translation environment again - just simply refine what we have.

Enhance user friendliness and efficiency

For a variety of reasons, it is not hard to find translation systems where user friendliness could be improved. For example, systems may have been put together quickly to meet schedule requirements, they may have been based on tools originally intended for other purposes or they may simply have been developed without due attention to users' needs.

All systems I have so far used leave a good deal of scope for human factors improvements.

There are those who would say that real productivity gains are made by looking at ways of preprocessing the data, rather than paying attention to the user interface. I agree that one must seriously look at the preprocessing in order to gain productivity (in fact, I will be making some recommendations along these very lines a little later). Nevertheless, I feel that productivity can be significantly improved by attention to the user interface in the following ways:

1. The greater the efficiency of the user interface, the greater the productivity of the translator or validator. This is especially true for the highly repetitive actions which occur during editing. Careful thought should be given to the way the translator/validator works, and how the system can be made to respond to the user's needs as quickly and as simply as possible. For example: How can appropriate dictionary information be inserted into the target document faster? How can the user access the parts of the text they want to work on faster? How can the user raise queries faster?

2. User friendly interfaces will reduce the time spent in training and retraining users. This is particularly relevant where translation staff is subject to high turnover or where the translators have to work with a number of different translation environments in the course of their work. The interface to the translation tools must be as simple, informative and *intuitive as* possible. Great attention needs to be paid to the way the user sees what they are doing, in order to allow for easy use of what could, otherwise, become a complicated system.

3. Streamlining and simplifying user interfaces for administrative tasks such as preparing the translation data, terminology development, handling of queries, etc., can often greatly reduce the overheads in cost and scheduling associated with the actual translation exercise. The points at which the user intervenes need to be kept as simple and few as possible.

One potentially very useful tool for translators would be a 'morph modifier'. This would allow translators to quickly and easily post-edit word endings, agreements, etc. Automatic adjustment of endings, etc., when pulling items from the dictionary into the target text would also improve productivity.

Minimise the work of the translator

If we are to achieve simultaneous multinational launch of products into the worldwide marketplace, we need to ensure that translation is as efficient and productive as possible. One of the key areas in which translation productivity can be tackled occurs before the translator actually sits down to edit text. The objective of the data preparation stage should be twofold:

1. To reduce, as far as possible, the amount of text the translator has to translate
2. To provide as much assistance for the translator as possible while they are editing or supplying text (without swamping or slowing them down!)

It seems to me that there is the potential in the future to greatly improve the productivity of update translation by an integrated approach which includes the concept of 'change analysis'.

Figure *2* illustrates the composition of an example document or software text extraction which has been updated. (The percentage figures for each component part are chosen so as to clearly illustrate the points which will be made.) The figure shows a document or software update of which 70% of the

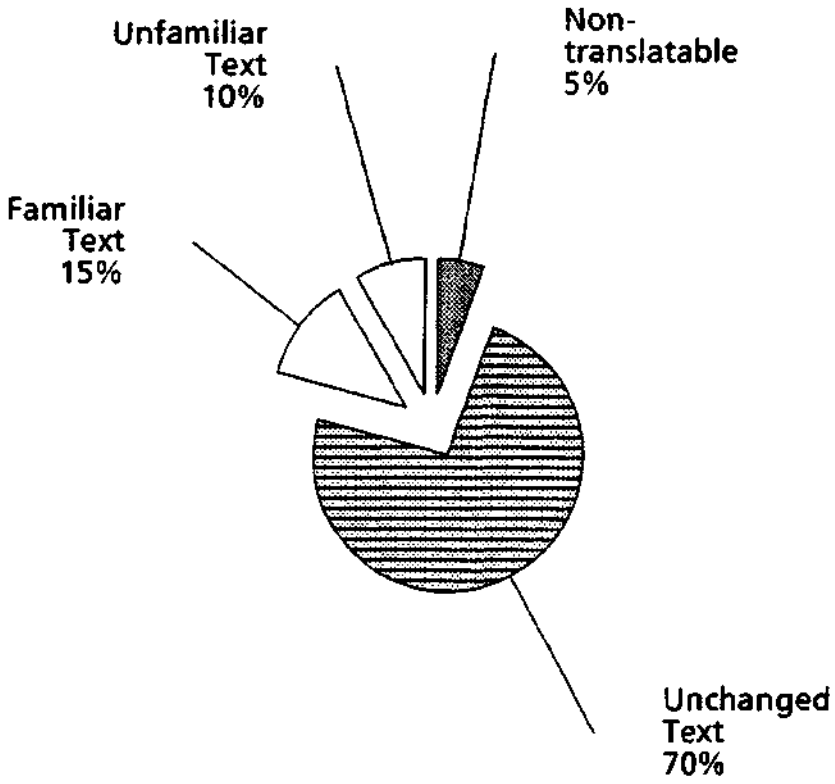

Figure 2: Composition of the example update.

source has not been changed since the previous version. A further 5% of the text is comprised of segments which will not need to change in translation. Of the remaining 25%, 15% of segments have

source text which will exactly or approximately match that of previously translated text held in databases - it is therefore labelled 'familiar text'. The remaining 10% is labelled 'unfamiliar text'.

If the text in our example is simply sent to a machine translation system, there is no way of carrying forward into the new document the lessons learned from post-editing the previous mistakes. All the mistakes made last time will be repeated and will have to be post-edited again. In addition, machine translation systems are still only of limited use for complex text.

The predominant way of tackling this problem at present is through the use of source matching. Source matches are obtained by matching the current source text in one way or another against that of previously translated databases. Where an exact or approximate match is found, the corresponding target segment is presented to the translator in a reference area or in the target text itself.

Many existing systems already rely on source matching (ie. repetitions or translation memory) to supply potential translations for all of the unchanged and familiar text indicated on the diagram. Source matching, however, provides only guesses at possible translations, usually taken *in isolation* from the context of the segment, therefore they must all be checked carefully. Where the recalled text is not exactly what is required, post-editing must take place or a new target segment must be typed in. This is far from ideal when you consider that 70% of the source text had not changed anyway.

I propose that future systems can gain significant productivity benefits for update translations by introducing the concept of change analysis. If a program is run on the data to ascertain which parts of the text are unchanged from the previous version, and label those appropriately, the translation tools should be able to automatically insert the previous translations from the database into the target document. It is important to bear in mind that approaching the task in this way relies on a high degree of certainty that the translation supplied was exactly the same as that associated with this very piece of source text the last time around. If the translation system labels these items of text, the translator should be able to automatically skip unchanged text if desired. The translator should still be able to make edits to unchanged text, if they wish - for example to ensure that unchanged text preceding or following on from changed text flows correctly - but if there are, say, ten pages of uninterrupted text which have not been changed, the translator should be able to skip right past them. (Again, this is usually acceptable since, unlike source matching, the system has used contextual information to find the exact same translations as were used previously for these source segments.) Anything the translator misses should be captured during the validation stage.

Implementation

How would one implement such a system? It may seem that this is not a trivial problem for documentation text, but programs and methods do already exist which may achieve reasonable results.

Rank Xerox has already implemented such a screening system at the level of complete software messages, via the use of message identifiers (IDs). We ask the product teams supplying software to associate each message with a unique ID which is unchanging throughout the product life. We use this ID to locate the previous translation of the message or icon in our database, and compare the text and properties against the new version. If there is no significant change, we can automatically insert the previous translation into the new database and the translator does not have to see the message.

Additional productivity benefits

Change analysis can have additional productivity benefits. Sometimes, for example, only a small change may have been made to a long message. In some cases it might only be a letter which has been put in upper case, or a article which has been added to the source - things which usually don't

affect the translation. In other cases, it may only be a comma which has changed, but this may affect the translation significantly. In these cases, the change analysis program should draw the attention of the translator immediately to the actual changes made to the source, so that the impact to the previous translation can be quickly assessed. Otherwise the translator may spend a lot of unnecessary time scrutinising the text to find the changes.

An integrated approach

Change analysis does not, by any means, do away with the need for source matching. It should be seen as only one of a series of operations on the text during the data preparation stage. Having identified the unchanged text in our example above and copied the old translations to the target file, we have 30% of segments remaining.

Our example shows that 5% of segments contain text which will remain unchanged in translation (for example, numbers). Determining what constitutes a 'non-translatable segment', and how to deal with it, should actually be done on a language by language basis. For example, decimal numbers need translation in some European languages but not in others. All numbers may need to be translated for Saudi or Thai markets. The key point is that, if the tools contain the appropriate information, non-translatables can be transferred automatically to the target document and leave the translator with more time to address the real translation work.

Continuing with our example, we now have 25% of segments left. This is all text which has been changed in some way. Of this, three fifths will actually match against the source of other databases during source matching. Alternative recalls[3] should be ranked and provided either in a reference area, or a mixture of target file and reference window, according to preference. Whereas, with most current systems, the translator has to check that recalls are valid in their context for 80% of our example, use of change analysis has reduced this activity to 15% of the text.

The remaining 10% of segments in the example could be translated in one of two ways. They could be translated interactively with the assistance of dictionary information in the reference window, or they could be sent off to *a* machine translation system and post-edited - whichever is most appropriate for the material and circumstances in question.

Summary

Specific innovations proposed in this paper include:

- the use of *change analysis* to reduce the amount of work involved in update translation
- the use of *job profiles* to reduce the need for administrative intervention
- the representation of available space for software messages within the editing environment so as to eliminate the cyclical approach to ensuring that text fits on the Ul
- the use of simulators linked online to the editing environment, to improve the context available to translators.

3. A recall is the foreign text retrieved from the previous database during source matching which corresponds to the source segment matched.

General recommendations include the following:

- Allow for a mixture of machine translation and other forms of translation within the same document
- Give the user greater control over how the system is used by making it more data-driven
- Reduce the complexity and amount of administrative support wherever possible
- Enable decentralised translation from a centralised administrative hub
- Design software in such a way that it can easily support, or be extended to support, any number of languages, according to the changing demands of the business
- Improve the context available to the translator by whatever means possible
- Enable immediate and intelligent sharing of useful information among translators and support staff
- Enhance the user friendliness and productivity of the translation environment wherever possible
- Minimise the work of the translator by carefully examining ways of building the target document automatically.