[*Translating and the Computer 7*. Proceedings of a conference… 14-15 November, ed. Catriona Picken
(London: Aslib, 1986)]

Translation and information processing: a Canadian viewpoint

Benoît Thouin

Programme Co-ordinator and President, CLC Ltd, Quebec, Canada

Translation – at least to some extent – and computer science come under the field of information processing. They both have played and are playing a major role in the information revolution which is dominating this second half of the twentieth century. They both help us realise that information is just as much a resource for an organisation as are people, equipment and capital.

It took quite some time for computers to become really useful to the language fields. Computer science (considered here to include electronic data processing) had to undergo a whole evolutionary process and its place today still gives rise to questions and debates, perhaps not for specialists, but at least for its potential users. This evolution in computer science has resulted in a number of tools for translators; it still is and will be a determining factor in the changes we observe or foresee in the translation environment. The major aspects of this evolution will be described below, together with some of their effects on the language profession. Some attention will also be given to translation and adaptation of software as a new field of activity for language specialists. This will lead to some thoughts on the future of machine translation (MT) and computer-assisted translation (CAT) systems.

THE EVOLUTION OF COMPUTER SCIENCE

Hardware

The concept of generations of computers is well known. The fifth generation, in any case, is the topic of much discussion, and not only in specialised circles. These generations correspond to the successive technological advances made in the production of the main computer units: first generation – vacuum tubes and germanium diodes, triodes and pentodes; second – transistors; third — integrated circuits; and fourth – large-scale integrated circuits, as used in present microcomputers. The fifth generation has more of a software connotation. On the hardware side, which is still at the research and prototype stage, it integrates circuits on an even larger scale and might integrate newer technologies such as laser memory, superconductivity at very low temperatures, and parallel processing.

The successive jumps from one generation to another have until now resulted in a significant increase in processing capacity, the miniaturisation of circuits and a decrease in production costs. These improvements have led to the most extraordinary machine that the human race has ever created. While machines can multiply our physical speed and strength by factors of the order of thousands, the computer is capable of increasing a million times over our capacities for calculation, data storage and information processing, which are much more fundamental human skills, all at relatively low cost and with almost no energy consumption.

The consequences of this on the language profession are numerous. The most obvious one is that computers are small, ordinary and cheap enough so that small groups and even individual translators can afford them and enter the electronic information age. The electronic medium is slowly but surely taking over paper to the point that in some instances the translator uses only magnetic and electronic media. But one should not forget that these media have a lower level of compatibility, readability and security than hard copy.

All this brings new problems and new opportunities to the language profession. Translators now have to worry about investing thousands of dollars in a single piece of equipment, something they never faced before. They do such things as product comparisons, cost-benefit analysis, selection, purchase, installation, training. They see new products every week, do not know what to choose, would like to avoid obsolescence, fast depreciation, and incompatibility with their clients' equipment.

Larger translator offices are created to share the initial investment, but translation permits few economies of scale and new problems arise with overheads, quality control, co-ordination of large projects etc.

New formulas emerge: group purchase of equipment and supplies, networks and co-operatives of translators, translation brokerage and dispatching.

These complex and sometimes dramatic changes in the translation picture are welcome since they force translators to think seriously about work organisation, to select appropriate tools, and not necessarily computerised ones, according to the type and size of text, to get more information and training. But they must also be careful not to put too much emphasis on hardware and equipment, which is the most visible but least important aspect of the information processing revolution.

Software

Application software, those instructions to perform some function for the user, also has its generations, more or less five years behind those of the computer. The first programs were directly encoded in binary numbers or written in a language very close to the internal structure of the computer. Therefore, only specialists could use computers since there was no distinction between using a computer and programming it, and it was necessary to understand thoroughly the structure of a computer in order to write programs. Moreover, a program could never be transferred from one computer to another.

Then appeared the so-called 'high-level' programming languages that could give the equivalent of a number of machine instructions in a relatively understandable statement. Some of these languages, for example FORTRAN, were sufficiently standardised to enable the transfer of programs from one type of computer to another. This second software generation, however, did not give users direct access to the computer; instead, users had constantly to enlist the aid of computer specialists.

The third generation was that of software packages: general programs equipped with a specialised command language through which users can operate the computer without having to program it. A software package can be designed for a single purpose, such as word processing, bookkeeping or inventory control, or it can combine several applications. A good package has menus, pictograms, or other action-selection features, error-detection routines, easy-to-understand messages, a help facility and appropriate documentation, that make it easy to use. Trends in third-generation packages include improvements in human-machine communication: increased use of graphics and pictograms, touch-sensitive screens, some voice input etc., and the integration of several applications in one package through split screens and interconnection of parallel processes.

Fourth-generation languages began to appear several years after the fourth-generation computers. These languages, built around a database management system, allow the user not only to operate the computer but also to define and implement applications by him or herself much more rapidly than before and with a minimum of help from computer specialists. The emphasis is no longer on procedures to access and process data, which are relatively standardised, but on data definition and organisation. There is also a new emphasis on functional definition of systems rather than their technical structure, the *whats* rather than the *hows.*

The data administrator has emerged as the main user representative as far as computer applications are concerned, and his/her role has become much more important than that of the computer systems analyst.

Fourth-generation languages and software are still undergoing development, but the real phrase à la mode is 'fifth generation', defined by the key words 'artificial intelligence' (AI), with expert systems thought of as a sort of an intermediate step. It is expected that this generation will set itself apart by, among other things, the logical inference capabilities of its systems, human-machine communication in natural language, and knowledge representation systems, at least in some limited fields. Let us not delude ourselves, however; as long as we do not understand the working of our own minds and as long as we have not established that it is possible to simulate it through the limited operations of the computer, artificial intelligence will remain much more artificial than intelligent.

The scope of software applications has also evolved: from military to scientific to business applications before entering the domain of the humanities (including language) and new technologies. The first effect of that evolution on the language profession is that third-generation packages were made available to writers and translators. Nowadays, several user-friendly packages are used daily without too many problems.

The world of software covers many application areas that ignore each other, which results in duplication of effort and lack of utilisation of new developments in other areas. For instance, terminology banks and information retrieval systems have resulted in large and powerful database systems not encountered at all in management information systems. On the other hand, fourth-generation languages and techniques are under-used in linguistic applications. In MT and CAT, integration of general database management systems with specialised linguistic and logic meta-languages is achieved only in a few instances, as is a recognition of various software tools as complementary instead of opposed to each other. The building of more bridges between subdomains of software is certainly a desirable step for the coming years.

Access to the computer

The methods of operating computers have also developed in a spectacular way over the short history of computer science. In the beginning, the user had to be by the computer in order to enter data through switches, but later he or she could prepare data independently by means of unit record devices, particularly punched cards. Batch processing would then most often be done: the user would group the records into batches, bring them to the processing centre, and later obtain printed results.

Developments in the theoretical study of processes, in parallel processing and in communication between processes led to improvements in operating systems, from monoprogramming to batch processing of input/outputs, multitasking and time-sharing. Time-sharing enabled the introduction of online processing through terminals connected to a mainframe by relatively short cables, and later remotely connected by telephone.

Telephone links were used at a later stage to connect computers to each other, in increasingly larger networks. Computer specialists also learned to operate a number of CPUs together in the same location, thus increasing the reliability of computer installations and leading to various concentrations of similar functions within computing centres. A now typical application is the use of a separate computer, called a front-end processor, to handle communications with the terminals.

More recently, the replacement of standard terminals by so-called ‘intelligent’ terminals, which have their own processing capability, and then by true computers (generally microcomputers) has added a great deal of flexibility to all kinds of computer systems. While the user is back to a by-the-computer situation, he/she has immediate access not only to more powerful means at divisional or organisational levels, but also to national and international networks. The result is multilayered systems whose throughput can be maximised by optimising the distribution of data and workload among levels and among computers at the same level. Such distributed processing systems are now so sophisticated they no longer require that all databases be centralised, as was thought to be inevitable. We have not yet explored nor implemented all the new possibilities created by these developments.

Language specialists now have access not only to various system configurations, but to numerous makes and models of hardware and software of all kinds. It is very nice to have several alternatives, but too much is too much. Over 300 makes of microcomputers, a few times as many models, thousands of pieces of software ...

Moreover, any single user can have his/her computer custom-modified through chip programming and/or software: new character generators, multiple keyboards, printer drivers, special instruction sets, anything you can think of. Such customisation is more like ‘modifications not authorised by the manufacturer’ (did you read your warranty card?) than just selecting the colour or a few options on a new car. For one standard that is painfully set up, five are violated, if they ever existed. Multilingual character set definition, for one, is near chaos. Even standard pieces of software will modify such things as the BIOS (Basic Input/Output Supervisor) and make the most innocent and plain microcomputer behave like a non-standard one.

For sure, standards deserve what happens to them. How can one comply with a standard for French characters where ë and ï do not exist, and where ten accented letters are present only because ten standard characters (square brackets for instance) have disappeared? Indeed, stupid standards are in essence self-destructive.

Shall we give up compatibility between systems? Or create new standards that deserve this name? Sometimes the latter might be a difficult decision. One should admit, for example, that the old 8-bit byte is obsolete in a truly multilingual environment. The 16-bit and 32-bit architectures, plus lower memory and storage costs should help us make the step up to more powerful character representation systems.

Methodologies

Our ability to develop and implement systems that satisfy the user's requirements has always lagged far behind our technical skills. Until the Seventies, there was no defined systems development methodology. Phased approach and systems theory, for example, did not exist, although some successful project leaders had used similar methods more or less intuitively.

An important concept that emerged was that of the *system* – a set of inputs, procedures and outputs, and of human, material and financial resources, which perform a well-defined function. Only recently has information been considered as another type of resource involved in the development and operation of a system. The recognition that a systems development project is in itself a system with its own resources, inputs, procedures and outputs led to the phased approach of systems development, or the 'systems development life cycle'.

This approach was the first clear methodology available for recognising the need for a new system, and then developing and implementing that system. With a few variations, the approach consists of seven phases, which are as follows:

1. **Project initiation** Analysis of the situation and identification of problems, user requirements and new opportunities, ending in the decision to develop or not to develop a new system.
2. **Feasibility study** Examination of various possible system solutions and comparison of these in terms of their technical and economical feasibility. Recommendation of one solution, based on cost-benefit analysis, including estimates of the development and operating costs.
3. **Systems analysis** Definition of *what* the system will do: description of the various functions and their interactions within the system, definition of inputs and outputs, all from a user's non-technical point of view.
4. **Systems design** Technical description of *how* the system will work: screen and report layout for inputs and outputs, physical

organisation of data and files, programs, modules and interfaces.

5. **Programming and testing** Writing of programs and testing of individual modules, followed by more general testing of the interfaces between modules.
6. **Implementation** Physical installation of the system, data conversion, user training, and acceptance testing by the user under normal operating conditions. Final acceptance of the system.
7. **Post-implementation evaluation** An on-going review, detection of problem areas and maintenance of the system, in whole or in part, when any major problems are detected or improvements projected.

Since 1980, ‘structured’ techniques have been more and more widely used to develop and to maintain computerised systems.

Data flow diagrams represent the circulation of data and the processes that use and produce information. Each process can be further broken down at lower levels, to give not only a general overview of the system but also an accurate representation of details.

The data dictionary contains a definition of all data at the elementary and higher levels. It ensures that each piece of information used or produced by the system, or created temporarily within a process, is clearly and consistently described. Where necessary, it also contains a statement of the validation conditions for data elements.

Structured language is a means of describing processes. It is a subset of a natural language that imposes a strict use of designators as they appear in the dictionary, together with the primitives and connectors of structured programming languages like Pascal. Structured language also prohibits the use of vague verbs to describe the action performed over input data flows. It can very well be thought of as a system description Language for Special Purpose (LSP).

The structured-module representation of the processes describes how each process is divided into programs, modules and routines. It shows the arguments received and the results produced by each program component of the system.

All these structured techniques include a mathematical means of checking for consistency and balance between the various components of the system, both at the same level and between two consecutive levels. Data flow diagrams, the data dictionary and structured language are used mainly at the systems analysis stage, while structured-module representation is a systems design technique. Data flow diagrams can also be used to describe the existing systems, and the various drafts of the proposed ones, in the early stages of the development cycle.

Structured techniques do not in fact represent a new methodology. They only provide a more handy means of describing the existing system and of representing it under development during the two critical and complex phases of systems analysis and design.

It is only lately that there has appeared what can be considered as a real second-generation methodology, known as prototyping. Unlike the phased approach, which can be described as the vertical development of a whole system in five to seven phases, prototyping consists in the complete and relatively fast development and implementation of a prototype, followed by a horizontal expansion to the full system.

The prototype can be one of the many functions of the full system, selected because of its priority to the user, its self-contained character or its strategic position in the overall flow of information; or it can be a very simplified version of the whole system, without the special or difficult cases, just to see whether the general principle of the system is sound or not.

The availability of a fourth-generation package is important to prototyping, because it accelerates the development and implementation of the prototype, and provides a highly flexible tool to rework the data structure as needed in the course of expansion to the full system.

An advantage of prototyping often encountered is that the user is provided with ‘something that works’ very early in the development process. But more important is that the prototype allows one to see exactly what kind of a system one has defined and permits early adjustments that can't simply be thought of over a pile of abstract diagrams and definitions.

Methodologies appear to be far behind the development of computer hardware and software. Other emerging techniques, like automatic production and proof of programs, are almost always absent from linguistic systems. The reason given is that linguistic applications are too specialised to be developed in a general framework adapted to more classical environments. The evolution of systems development methodologies has little affected the language profession. Let us hope it will, be it only as will be seen below to help language specialists better participate in the adaptation of software.

User involvement

The client’s participation in the development and utilisation of computer systems has also dramatically evolved since the beginnings of computer science.

At first, the computer specialists were the only designers/clients/users of computer systems. Then appeared other people: clients of computer specialists and consumers of computer services. They would be more or less afraid of those complicated and extraordinary devices and would have to rely totally on specialists, like someone who feels ill and can only believe the medical science that says there is a cancer there and that some precious but defective part of his/her body has to undergo some strange, painful but certainly successful treatment using the latest and extraordinary ‘omega ray’ generator.

With the almost simultaneous advent of the third generation of software and of the phased approach to system development, there appeared two different types of involvement of the user and/or client. Buyer and user of custom-developed software would be consulted at the project initiation and systems analysis stages, and would approve the deliverables of all phases. (A deliverable is any program, document or other concrete result that must be delivered during or at the end of one phase of the development process.) On the other hand, the user of third-generation software gets a user-friendly and well-documented package, but only a few people supposed to be representative of the target clientele are involved in its development, on a consulting basis, as members of test groups etc.

With fourth-generation software, the end-user becomes the main agent of the design and implementation process. The role of the computer specialist remains important in the general framework – building the fourth-generation environment — but becomes limited to special support upon request as far as applications are concerned.

Through their greater involvement in the development of systems, users have learned the golden rules of good computerisation: (1) define your requirements (qualitative and quantitative); (2) then look for or define the software that will satisfy these requirements and (3) last (and least), choose one set of hardware compatible with the above software.

The composition of research and development teams in linguistic systems has followed that evolution, from rather closed teams of mathematicians, computer scientists and theoretical linguists, to include eventually translators, writers, lexicographers and terminologists. The very nature of machine translation systems has changed, from fully automatic to human-aided machine translation, to computer-assisted human translation. Language specialists have become more and more interested in the development of their own working tools with the result that they get much more satisfactory products to work with, and offer less resistance to change.

Technology integration in the work environment

The evolution of computer science has led to the integration of systems along two different lines. One is the mix of several applications in one package. For example, accounting packages will include book-keeping, journals, general ledger and financial statements but also extensions and links to other applications such as purchase orders, billing, inventory control, production scheduling and material requirements planning etc. in an all-in-one type of superpackage.

The second line of integration is much more important in its scope and effects. Computer science has not only evolved dramatically as a field in itself, but also has imposed its tools (computers) and its methods (especially programming and top-down analysis) on other fields of activity. Here are a few examples of this remarkable and apparently unique phenomenon.

From the chisel to the pen, to printing, to the manual then electric typewriter, the evolution of writing instruments has now reached the computer stage. A word-processing system is nothing more than a computer specially programmed to that effect. It is thus not always necessary to buy a specialised piece of equipment; in fact, a word-processing program is part of the free package that comes with the purchase of most microcomputers.

Oral communications are also handled by computers: telephone exchanges and Programmable Branch Exchanges (PBXs) are simply computers with telephone receivers as terminals. Computer-controlled dictating machines have functions similar to word processing that allow for moving around paragraphs, deletion of portions of text etc., the operation being symbolically displayed on-screen while its equivalent is performed on tape. Drawing boards have become computerised as well, offering features such as automatic calculation of various sections and movement simulation on the screen, all in full colour and with more precision than conventional drafting and reproduction methods. In manufacturing, assembly lines have become gigantic robots, that is, computers which have sensors, articulated arms and various devices as peripherals, instead of the usual keyboards, screen and printers.

The adoption of computers and their methods in fields other than computer science leads to an easier integration of various functions and opens the door to a new synergy. Consider, for example, the development of workstations which group together the functions of a computer terminal, word processor and telephone receiver, and computer-aided design and manufacturing (CAD/CAM) applications by which plans designed through computer graphics are directly used in manufacturing to operate robots. Computer science has given birth to office automation, teleprocessing or ‘telematics’, computer graphics and robotics – the integration of these new technologies in our working environments depends only on our imagination.

Besides the implementation of integrated workstations for language specialists, the integration of several advanced technologies has opened new opportunities to translators and given much research work to terminologists. Information, as the raw material necessary to disciplines of communication, appears to be not only a renewable, but a self-multiplying resource.

TRANSLATION AND ADAPTATION OF SOFTWARE

Our interest in computer aids for translators should not make us forget that with the advent of computers and computer science, data processing, word processing, office automation, teleprocessing and robotics, more work has come to the 'in-tray' of technical translators and a new specialisation has emerged. Moreover, since information systems have invaded more and more domains of activity, it has become a must for any specialised translator, if not for every translator, to be familiar with at least the fundamentals and the basic terminology of information systems. This demonstrates the effect of computer science on the language profession.

Added to the ever increasing use of computer means as an aid to translation, this has resulted in a closer symbiosis between the fields of translation and information technology. Besides that, the multiplication of computer systems and their use by ordinary people in a variety of locations and situations has resulted in a quite new and specialised activity: the translation and adaptation of the computer systems themselves.

Given the wider definition of a system (set of inputs, procedures, outputs, and various resources), it is clear that the translation of a computerised information system includes, but is not necessarily bound to, the translation of user documentation. Indeed, translation of the documentation is only the first of three levels of translation one can distinguish in information systems.

The first level consists in translating only the documentation of the system without modifying anything in the programs. More often than not, only the user documentation (and the operations manuals in the case of larger systems) is translated, while the system and program documentation is left aside. This implies that only individuals fluent in the language of origin of the system can understand it in all its aspects and modify it (which, by the way, implies rewriting the documentation of the modified parts).

This level of translation of a computer system is still quite frequent. Nevertheless it is very primitive and leads to problems of various kinds. As an example of these problems, let us mention the necessity of keeping the commands and messages of the system both in the source and target languages in the translated documentation, to quote exactly what the system displays and expects and to make it understandable for the target language reader, leading to either hybrid or heavy text.

Such a phrase as *la commande SUPPRIMER* will become in English 'the SUPPRIMER command', or 'the SUPPRIMER (delete) command', to be more explicit for a reader who has little knowledge of the source language. It is the same for sentences like *Le système affiche: 'Voulez-vous vraiment supprimer cette page (O/N)?'*, that would be translated, 'The system displays: "Voulez-vous vraiment supprimer cette page (O/N)?" (Do you really want to delete this page (O for yes – OUI, N for no)?)'.

There are several ways to translate in this type of situation, but all of them are either tedious or incomplete. Moreover, one cannot limit oneself to translating more explicitly at the first occurrence and less explicitly afterwards, since computer systems documentation is often used for reference purposes and very seldom read from cover to cover.

The second level of translation includes partial reprogramming of the system in such a way that at least the messages and reports, and sometimes the commands of the system, are in the target language and that the system seems to understand and speak the user's language.

Usually, this reprogramming affects only data areas in the system and not the algorithm itself. Field lengths must be redefined according to the target language wording. A little bit more delicate is the problem of commands abbreviated in short mnemonics, since two different commands in the source language may have the same abbreviation in the target language. Synonyms must be found, not always a satisfactory solution. In some cases, a report that takes up a certain number of lines in the source language will require more or less in the target language, resulting in slight modification of the algorithm (new statements to add or delete lines, more or fewer items per page etc.).

But this is little compared to the third level of translation, the only complete one, which tends to make the translated system as a whole look like a genuine system in the target language. In addition to what has been discussed before, this implies including the alphabet (and diacritics) of the target language in the keyboard, screen, printer, communications protocols and internal character representation of the system. This can be very complex since, for example, the traditional one-byte representation of characters is often insufficient, some letters change shape when followed or preceded by some others, left-to-right writing is not universal etc.

Some physical indications on the equipment may have to be translated. Moreover, the very principle of the system may have to be adapted to the new linguistic and cultural environment. For instance, the operations performed by an accounting system will be different from country to country (even those that use the same language). This latter case resembles the change of technical specification of any device so as to adapt it to another environment, which also reflects in the documentation.

It should be noted that the third and to some extent the second level of translation imply that translating and adapting a computer system is in fact an activity of system development, which requires more than linguistic knowledge and renders necessary the use of system-engineering methodology. Since few translators (in fact few individuals) master both fields, multidisciplinary teams become necessary, and must be properly organised and managed.

Several system development teams include one or more technical writers to prepare the user and technical documentation while the system is being designed and implemented. The concurrent development of versions in several languages or the design of truly multilingual systems brings new opportunities not only for writers, but for translators and terminologists, to do their communication work under new and exciting conditions.

The fact that so many software packages have been adapted to other languages has contributed to the development of foreign-language facilities on several computers and operating systems. It has also helped several translators to participate in the information systems revolution.

FUTURE DIRECTIONS FOR MT AND CAT

Looking at the general evolution of computer science, let me suggest some directions for research in MT and CAT. Certainly not all of them will prove fruitful, but they deserve to be honestly investigated.

Clearly, the integration of sophisticated database management systems, of fourth-generation languages and of parallel processing in MT and CAT systems should be examined.

Research on LSP and expert systems will at some time have common features inasmuch as experts use special-purpose languages in their domain. AI programming languages like Prolog and LISP might be interfaced with linguistic tools such as Q-systems and ATN.

Serious theories of translation, as well as psycholinguistic theories, should find an echo in actual MT systems. Research on knowledge-based systems could be made more effective by putting together the contributions of information representation and storage activities such as thesaurus building, lexicography and terminology.

As a final example of possible co-ordination of efforts, I can refer to the very many computer or high-tech firms that use or plan to use machine translation to translate their documentation. I am sure that in some cases, the documents to be translated describe a computer system whose formal description (data flow diagrams, data dictionary, description of procedures) is available somewhere in machine-readable form. I suggest that this description could be somehow used as a knowledge base, helping the MT system better to translate such things as the operator’s and user’s guide, overall system documentation etc.

These are only a few of the many possible directions for research in MT and CAT. In the meantime, we should not forget that translators and other language specialists require right now better tools to do a better job, to be more productive and to be happier. Such things as the translator’s workstation can very well be developed and made ready to include MT as another module.

AUTHOR
Benoît Thouin, President, CLC Ltd, PO Box 420, Station A, Hull, Quebec, Canada J8Y 6P2.