PRACTICAL EXPERIENCE OF MACHINE TRANSLATION
V. Lawson (ed.)
North-Holland Publishing Company / ©ASLIB, 1982

QUALITY CONTROL PROCEDURES IN MODIFICATION OF THE AIR FORCE RUSSIAN-ENGLISH MT SYSTEM

Dale A. Bostad

Hq Foreign Technology Division
Air Force Systems Command
United States Air Force

The paper gives the background leading to the development of current quality control procedures used in modification of the Russian-English system. A special program showing target language translation differences has become the central control mechanism. Procedures for modification of dictionaries, homographs and lexicals, and generalized linguistic modules are discussed in detail. A final assessment is made of the procedures and the quantitative results that can be obtained when they are used.

The Air Force Russian-to-English MT system, operational since July, 1970, is an immense software package that has translated about 500,000 pages of Russian text to date. Through the years the system has been enhanced by the addition of new routines and extensive modification of the dictionaries and linguistic modules. These optimization efforts were of course subjected to some degree of quality control at all stages. However, in this paper I intend to concentrate on the quality control procedures currently used in the Russian system. They have evolved in recent years as both the software developer, LATSEC, Inc., and the user, the U.S. Air Force, gained insights into the system and realized the necessity of developing new procedures for its modification. Because we are dealing with a natural language translation system, these procedures have some unique features; however, in the main they do not differ significantly from those used in other software applications.

Another term often used for quality control is configuration control management. It has been defined as the control of changes to established baselines.[1] At question is maintaining the integrity of a software system during its evolution. Configuration control management, then, is the administrative structure for initiation, evaluation, and incorporation of changes into a system.

To understand how we arrived at our present position in terms of configuration control management, it is necessary to review some aspects of the system and our approach to modification a few years back.

A great deal is asked of the Russian system: accurate translation of random Russian text in a wide range of technical fields with no pre-editing and only limited interactive post-editing. Hence we rely heavily on the accuracy and resolution capabilities of our linguistic software and it has been our long-standing policy to improve the software as much as possible. And there has been little doubt that the system could be improved. In a five year period the number Of homograph subroutines doubled; during the same period of time stem entries increased by about 30,000. New features were constantly being developed: new routines, new parsing expressions, new macros, new codes. The system was in a constant state of flux; it underwent rapid change as we attempted to address an apparent infinitude of small and large problems that arose.

There was substantial evidence that the system was being improved: readers of the product often commented that they thought the product was better; the dictionaries became so comprehensive that they were often used by experienced in-house translators. There was also evidence from other sources. A study[2] conducted by Battelle Columbus Laboratories concluded that the carry-over effect of concentrated dictionary update was 40% in related technical fields and that significant improvements were obtained in the raw translation output. Yorick Wilks' study[3] showed that the carry-over effect of dictionary and program changes in a different subject area, i.e., from scientific areas to social and political topics, on balance was 20% net improvement.

But there was a growing concern that progress was not linear, that in fact, a certain degree of degradation was occurring. Several events suggested this: frequent linguistic abends after loading a new system; the subjective judgment of post-editors that something that used to work in the system no longer operated correctly; and the fact that certain acute errors were detected after system modification. One question became pressing: how can modification to the system be quantitatively and qualitatively measured? The answer came in 1978 when a program was developed to measure and ultimately to control change. It was called COMPUP (comparator) and became the cornerstone of our quality control procedures. Briefly, the program prints out all target language sentences showing translation differences between two system versions. Review of the output provides quick and accurate data on the rate of change and types of change between two systems and whether the changes are positive or negative, producing, in effect, an improvement/degradation ratio.

When comparators were run against new system versions over a period of months the improvement/degradation ratio was consistently around 7:3. Wilks' figures also showed approximately 30% degradation in sentences changed. The conclusion was that system modification was coupled with a risk factor, that system-wide impact of modification was unpredictable, and that improvement in one area could degrade something elsewhere. This created a cycle of construction and destruction: in the aggregate there was considerable progress, but it was not uniform and we were incurring substantial losses.

At this point we decided that new quality control procedures for modification of the system had to be implemented, with the comparator the chief mechanism of control. The general approach was based on several concepts:

- Zero degradation, or at least minimal degradation, was the goal.
- Baseline systems had to be defined and maintained.
- The rate of modification had to be reduced.
- Permanent data bases for specific problems and routines had to be developed.
- Comparator tests run against random Russian text would be the final test for validation of change and system acceptance.

We were advancing radical changes in modification procedures. The magnitude of problems in the system and the seeming ease of their resolution by rapid modification led to overloading the system, keeping it in a constant state of turmoil. It was time to step back, reevaluate existing control procedures, and proceed in a slow methodical manner. It was the realization that more was not necessarily better.

Let us now turn to those quality control procedures and examine them in some detail.

DICTIONARIES - STEM EXPRESSION

The incorrect addition of a code to a word or group of words in the dictionaries can have disastrous consequences; yet, we consider dictionary modification the easiest kind to control and that causing the least worry. Mechanical error during coding is largely reduced by two computer programs called NDEDIT and DAUDIT that reject illegal transactions and check for missing fields or conflicting codes. Subjective decisions on particular translations, setting of linguistic relationships, and the addition of syntactic and semantic codes are more difficult to control. Accurate decisions are usually based on common sense tempered with experience with the system. Our procedure is that all dictionary change submissions must be accompanied by evidence in the form of the word or phrase in context, a diagnostic printout, concordance examples, or source evidence from dictionaries, reference books, or subject experts. In addition there is *a* hierarchical screening process so that each submission is reviewed by 3-4 people, with the most experienced personnel making the final decisions. Lastly, dictionary entries are reviewed a final time after insertion into the dictionaries. This is a final check in an effort to weed out inadvertent error.

Certain other features have proved successful in controlling dictionary modification:

(1) No expressions are entered into the dictionaries without prior testing.

(2) All dictionary modifications are accompanied by note cards (N-cards) that document the history of changes to that entry.

(3) Words and expressions vital to the system are called security entries and cannot be modified without a password code. A joint decision by senior developer and user personnel is necessary to modify a key-lock entry.

(4) As in all modification, comparator runs against random Russian text allow us to pass final judgment on dictionary modification. Inadvertent error in modification of any frequent term or expression will be picked up.

HOMOGRAPH AND LEXICAL SUBROUTINES

We define these subroutines as relatively self-contained modules that solve problems apart from the mainstream of abstract linguistic analysis. They contain their own internal logic and errors are easily traced. As with dictionaries, we have developed set procedures for modification of homographs and lexicals.

The first step is to develop a data base for a particular routine by selecting sentences from concordances and using query messages to extract sentences from our library of Russian input text (approximately 50 million words on tape). Initially we strive for a comprehensive solution of a routine: sentences are collected that illustrate all major linguistic decisions within the routine and any other linguistic patterns not covered by the routine. These are called positive tests. The data base also contains negative tests, i.e., sentences which exemplify conditions when the routine does not operate or should not operate. The size of the data base depends on the complexity of the routine and the number of sentences available - we have data bases from 15 up to 125 sentences. The data base is put on tape, it is translated to determine the accuracy of the routine, and then the routine is modified. After modification, the data base is run against the modified and unmodified versions of the routine via comparator. The comparator may also be run against random Russian text, depending on the relative frequency of the word(s) in the routine or its complexity.

It is important to emphasize the development of permanent data bases for homographs and lexicals. In the past, routines were modified in response to events in individual sentences, but the sentences themselves were not retained. All that remained was the abstract logic within the routine, which might or might not have been sufficiently documented. We believe that retention of data base sentences is at least as important as internal documentation. Once a routine has undergone comprehensive modification no further change is ever allowed to that routine without running the modified source against the entire data base. One final point: there should be absolutely no degradation as a result of modification of lexicals and homographs. If any occurs, further debugging is necessary.

GENERALIZED LINGUISTIC ANALYSIS PROGRAMS

The heart of any machine translation system has to be its main linguistic algorithms. In the Russian system, generalized linguistic programs have less modularity, later linguistic decisions are contingent on the success of earlier ones, and the coding is extremely complex and interwoven - the handiwork of several linguistic programmers. This is the part of the system where the so-called ripple effect is most evident and where great care must be exercised in modification.

It is a data processing truism that the best time to influence system design is in the early stages of development. The Russian system is over ten years old and at this point in time it would be unprofitable for us to restructure the system using contemporary programming techniques. We are simply accepting the original design as a given and proceeding from there. This means that our efforts are mainly directed at fine tuning the system rather than making design modifications of significant magnitude. Nevertheless, a great deal of improvement is possible, in our experience, if good quality control procedures are used and enforced.

1. Problem Definition and Analysis

A baseline system is defined such that the software developer and the user have exactly the same system. Then a linguistic problem is defined - in our particular case 90% of the time by the user - based on analysis of large amounts of translated material. An attempt is made to establish a data base using production sentences, concordances, error collection files, or queries run against the 50-million-word data base. Sufficient sentences must be obtained to exemplify the parameters of the problem; no modifications are made on the basis of a few sentences. If a satisfactory data base is not obtained the problem is shelved.

The software developer writes a verbal analysis of the problem, listing all programs affected by the problem down to specific subroutines, estimates the degree of improvement that can be expected if the problem is tackled, and gives an estimate of the time and personnel required for writing and testing the solution. On the basis of this information the user decides whether to proceed with modifications, and if the go-ahead is given a priority is set for the problem. For ease in debugging, no more than 5-7 abstract linguistic problems are worked on during a three-month system modification cycle.

2. Testing and Documentation

The solution is tested against the data base developed using both positive and negative tests. Debugging continues until the projected degree of improvement - within reasonable limits - is obtained with minimal or no degradation. The source code is documented and a final report is written stating whether the solution reached its projected degree of improvement and, if not, the reasons for the shortfall.

The system developer then sends an updated system version to the user, accompanied by a report specifying all linguistic changes in that system version. Finally, the control data base is delivered to the user and it is input on tape and becomes part of the permanent library of problems and solutions.

3. Evaluation and Acceptance

a. The user runs comparators against the original controlled-text data base to verify the developers' results.

b. The user runs comparators against random production text. This is the crucial test: verification of whether the positive results of change in the data base carry over to random input without degradation. Approximately 30,000 random sentences are run comparing the two system versions.

c. The comparator results are judged. There has to be zero or minimal degradation. Depending on the user's appraisal, the new system version can be accepted, accepted with immediate minor debugging, or returned for major debugging until the degradation level becomes negligible.

d. After final acceptance, the modified version becomes the new baseline system.

The discussion so far should give a good idea of our quality control procedures. But what kind of quantitative improvement in the product can be expected by using them? Comparators run against random text in all subject areas show that the total impact of modification - dictionary, lexical, homograph, and generalized linguistic - will affect 1-3% of sentences. This figure may seem low, but in the perspective of 5,000 pages per month the number of improved sentences can be substantial. Obviously, the greater the frequency of a word, phrase, or linguistic structure that has been modified, the greater the impact on the final product. Comprehensive modification of lexicals and homographs has resulted in improvement from 10 to 50% in the data bases; enhancement of recognition of long-form adjectives as predicates yielded a net improvement of 35% in the data base. Again, visible impact is largely a matter of frequency: a 50% improvement in a high-frequency homograph will have substantial impact; 35% enhancement in a relatively infrequent subject/predicate structure will be less noticeable.

Organizations that achieve a high level of quality in their products first establish an acceptable level of quality and then build a mechanism that assures this level is maintained. The quality control procedures described above attempt to do this and the results have proven to be quite satisfactory. I wish to stress however, that quality control is not simply evaluation of the final product. It involves a lot of detailed administrative work that is not glamorous: collecting error sentences, keeping data base tape libraries of routines and problems, keeping historical records of problems worked on or set aside. In short, it is an on-going process throughout the evolution of a system as new baselines are established.

Configuration control management of a machine translation system is ultimately the responsibility of the user. He has to get deeply involved in the development and control of the system because he has to judge the final results and live with the product. The easy way out is to forget about quality control management, or if you have it, slack off and not enforce it as should be done. However, the price you pay may be high. A commitment to improving a machine translation system is a commitment to quality control.

REFERENCES:

[1] Boehm, B, et al. (1978) Characteristics of Software Quality. (TRW Systems and Energy, Inc., Redondo Beach, CA)

[2] Battelle Columbus Laboratories (1977) The Evaluation and Systems Analysis of the SYSTRAN MT System. (Report RADC-TR-76-399)

[3] Wilks, Y., and LATSEC, Inc. (1979) Comparative Translation Quality Analysis. (Final Report, Contract F33657-77-C-0695)

Perry, W. (1977) Effective Methods of EDP Quality Assurance. (Q.E.D. Information Sciences, Inc., Wellesley, Mass.)