Application of Extensible Languages to Specialized Application Languages

Jean E. Sammet
IBM Corporation

It is becoming increasingly clear that one of the major thrusts of the programming language field is in the direction of specialized application languages - sometimes called special purpose languages. These specialized application languages have already been developed for such diverse fields as civil engineering, electrical engineering, computer assisted instruction, systems programming, equipment checkout, movie animation, etc. Of the roughly 165 languages currently in use in the United States, approximately 85 of them might be classified as specialized. (See the list in Reference 1.) For description and illustration of many of these, see Chapter 9 in Reference 2.

All languages are applications oriented. In fact, this is probably the best way to categorize the set of programming languages. However, some application areas are obviously larger than others, e.g., scientific computation (numeric and non-numeric) and business data processing. The main concern here is with the more specialized languages. While each person may have a different idea of what the categories of specialization should be, there can be little dispute about the wide diversity and large number of them.

There are many reasons for the increased emphasis in this direction. Many users tend to find languages such as PL/I and ALGOL 68 too large for their more specific applications; they prefer their own vocabulary, and also the increased efficiency which can come from a smaller more specialized compiler. There are a number of different techniques which are used to develop these specialized languages: at one extreme is the design and implementation of the language independently of anything else in existence; what might be considered the other extreme is the modification of an existing language (and its compiler) to suit the specialized application. One of the possible techniques - although obviously not the only one - is the use of extensible languages.

Many of the advantages of extensible languages apply to their use in developing these specialized application languages. For example, the common core and extension mechanism permit the development of different but similar or related languages. Also, an extensible language would permit an individual or organization to easily make minor changes to a specialized language which exists. In fact it may even be true that the only practical application of extensible languages is to produce specialized application languages and their compilers or interpreters.

The biggest problem - and the one to which the extensible language developers should address themselves - is the incredible variety of syntax and semantics that exist in these languages. It seems unlikely that any of the existing or potential extensible language systems would be powerful enough to permit the generation of all the specialized languages now in existence. This inherent limitation seems quite justifiable on the grounds that it is futile to expect a single extensible language system to generate all other languages and that is what would be required to cope with the myriad of specialized languages now in existence. However, if there is not sufficient care taken, the other extreme may exist, i.e., an extensible language can be used to generate only one, or at most a few closely related specialized languages. In such a case, the extensible language probably serves relatively little purpose. Naturally, the breadth of languages which the extensible language system can generate is a matter of judgment, technology, and efficiency just as it is for the design of any programming language.

One of the illustrations of a related system which was not developed via an extensible language but which is composed of a set of related languages is ICES, used for civil engineering. (See for example Reference 3.) ICES also contains a Command Definition Language which permits the user to define his own language within the ICES framework.

It is probably true that a single extensible language must restrict itself to a particular type of application area. Even within these, some of the variations in style can be enormous. It often turns out that the style of a language is based more on the personal views of the person who developed it than on any rational basis. Thus languages which are different in purpose and application area may have a similar syntax. Conversely, the same application area will surely breed different syntactic styles.

SUMMARY OF REQUIREMENTS FOR SPECIALIZED APPLICATION LANGUAGES

This summary indicates the types of facilities already known to exist. Naturally no single specialized language has all these requirements; however, at least one language has each requirement. Note that systems programming is one particular application area and its requirements are significantly different from most of the others.

In some cases the languages encourage or require implementation via an interpreter rather than a compiler.

Syntax

1. Free form.
2. Blank characters and/or punctuation allowed to be critical or not critical as delimiters. (Single choice made for each language.)
3. Fixed tabular form as in report writers or decision tables.
4. Rigid form with required parameters, i.e., essentially macro style.
5. Specialized vocabulary for key words.
6. Identifiers and verbs can be single characters or lengthy.
7. Full power of language like PL/I (for systems programming).

Data

1. Implicit, i.e., all data is assumed to be of a particular type, and/or clearly defined by position in statement.
2. Some declarations (perhaps with default conditions).
3. Data types specialized to application, e.g., length.
4. Wide variety (for systems programming).

Semantics

1. Specialized computational routines.
2. Non-standard use of common characters (e.g., plus sign).

Program/Control Structure

1. Generally quite simple with no block structure nor compound statements.
2. Very powerful (for systems programming).

REFERENCES

(1) Sammet, J.E. "Roster of Programming Languages - 1971", Computers and Automation, Vol. 20, No. 6B, June 30, 1971.

(2) Sammet, J.E. PROGRAMMING LANGUAGES: History and Fundamentals, Prentice-Hall, Inc., 1969.

(3) Roos, D. "An Integrated Computer System for Engineering Problem Solving", Proc. FJCC, Vol. 27, Part 2, 1965.