What's different about tactical military languages and compilers

by RAYMOND J. RUBEY
Logicon, Inc.
San Pedro, California

BACKGROUND

Until recently, the programming for tactical military computer applications relied on assembly or machine-oriented languages in contrast to the widespread use of higher-order languages in commercial applications. In the past, the tactical military software development process was plagued with many problems, including the relatively high cost on a per-instruction basis, the long development time required, the necessity for highly-trained engineer-programmers, the non-transferability of the resultant programs, and the considerable difficulty of effective program maintenance. The military customer who had to pay the penalties resulting from these problems became convinced that at least part of the reason for the problem was the reliance on MOLs. As a consequence, the military customer provided the impetus and funding to apply HOL concepts and technology to the unique problems of the tactical military software environment. At the same time, NASA, which encountered the same problems in developing real-time software for space applications, took the same approach. The conclusions reached from the resulting HOL studies and developments has led the customer, in many cases, to insist on the use of a HOL unless it could be conclusively demonstrated that such usage was infeasible.

TACTICAL MILITARY HOL ALTERNATIVES

An organization beginning the development of tactical military software has several alternatives as to the HOL it can use. First, an existing commercial HOL, such as FORTRAN, ALGOL, or PL/I could be selected. This alternative has been universally rejected because of the limitations of these languages in tactical military programming. A second alternative is to select and modify a language originally intended for another application to make it suitable for the tactical military software environment. This approach was taken, for example, in the selection of a JOVIAL J3 subset and its modifications for the B-1 avionics software development. A third alternative is to select a more specialized language that was developed in response to requirements of the tactical military software environment or the closely related real-time space software environment. The languages in this category include the Navy's Compiler Monitor System-2 (CMS-2), the Air Force's Space Programming Language (SPL), NASA's Computer Language for Aeronautics and Space Programming (CLASP), and NASA's HAL. Whether the second or third alternative is taken, the language must contain the facilities for performing functions not common in most commercial programming endeavors. Because of the different approaches of the language designers in solving the problems in the tactical military environment, the languages indicated above are significantly different with regard to the way these facilities are provided. The following paragraphs provide generalizations about these facilities.

TACTICAL MILITARY HOL FACILITIES

A tactical military program development is a more varied activity from the software viewpoint than the typical commercial program development. The tactical military programmer begins with an empty computer and must code his own executive program, I/O routine, subroutine library, and diagnostic routines. Thus a tactical military HOL must provide the facilities which, in the commercial environment, are associated with systems programming. Of course many of the facilities in commercial HOLs are essential in tactical military HOLs, including arithmetic, conditional, looping, and transfer-of-control statements.

Besides dealing with the usual numeric data, a tactical military HOL must allow for the definition and manipulation of logical, Boolean, textual, and character data. It also must provide the facility for manipulating portions of data words down to a single bit as well as the usual full data words. All of the conventional arithmetic and Boolean operations on these portions of words should be provided. The programmer uses this facility to operate on the varied inputs and outputs received and transmitted by the typical tactical computer and to create the needed data structures.

Many of the calculations performed in tactical military applications involve the movement of objects in three-
dimensional space. An effective tactical military HOL simplifies the programming of these calculations by allowing for the definition of appropriate arrays and by providing powerful non-scalar operators. Examples of such operators are vector dot product, vector cross product, and matrix multiply.

No matter how rich in facilities a HOL is, there may be some functions that cannot be easily or efficiently performed. Thus many HOLs provide for easy regression to assembly or machine-oriented languages. If this facility is not provided, the object code generated by the compiler must be modified, which is more difficult and more likely to induce errors.

A tactical military computer has specific locations dedicated to a particular purpose through hard wiring. Examples are the dedicated, fixed locations where the computer registers are automatically copied when an interrupt occurs. Through declaration statements, the HOL must provide for the association of programmer-defined symbols with such hardware locations and functions. Similarly, the HOL must allow the programmer to allocate the operational tactical military program and data to specific computer locations according to a declared memory map.

The architecture of many tactical military computers and their application often require the use of fixed-point arithmetic in addition to or in place of the floating-point arithmetic that is universal in other applications. This requires that the HOL have facilities for the declaration of fixed-point data containing both integer and fractional parts, and for the utilization of such data in calculations with a minimum of programmer effort. The scaling operations performed for fixed-point arithmetic calculations become an important part of the HOL semantics. The increasing use of floating-point architectures are likely to solve many of the existing problems before completely effective language and compiler solutions are found.

The real-time nature of tactical military applications requires that suitable HOLs contain facilities for the real-time control of the programs. Facilities are needed to enable and disable interrupts, to correlate specific program actions with particular interrupts, and to indicate the required interrupt levels. Less obvious but equally important is the need for facilities to control the accessing of data by several levels of interrupts. For example, an array computed at one interrupt level should not be referred to at a higher level without checking that the complete array has been computed.

The debugging and validation of a tactical military software system is the most expensive and difficult part of the software development cycle. A tactical military HOL must assist in these debugging and validation efforts. First, the language itself must have a minimum number of error-prone features or syntactic constructions. Second, it must enable the creation of a compiler that can perform a considerable degree of compile-time fault diagnosis. Finally, it must provide for the generation of run-time diagnostics, particularly those produced by simulated execution of the object code obtained without any changes in the object code itself. While such features as these are desirable in a commercial HOL, they are mandatory in a tactical military HOL.

One problem that has not been completely solved in any existing tactical military HOL is the definition of a computer-independent and general input/output facility. This problem is caused by the widely varying input/output devices in tactical military systems, the considerable differences in the input/output modes of tactical military computers, and the stringent constraints on real-time input/output operations. Indeed, some language designs are based on the assumption that input/output functions will continue to be coded in the appropriate MOL and therefore do not provide input/output facilities. Other languages provide basic input/output facilities little different from those provided in commercial HOLs. Regardless of the input/output facilities in the HOL, many compiler implementations contain custom-tailored input/output statements that are both computer and application dependent.

**TACTICAL MILITARY COMPILER CHARACTERISTICS**

A single commercial computer installation may execute a hundred programs in the course of a week; a single tactical military computer program, on the other hand, may be the only program executed in a hundred tactical military computers. While the savings in programmer labor made possible through HOL usage more than compensates for compiler inefficiencies in the commercial environment, this is not necessarily true in the tactical military environment. If a tactical military compiler generates code that requires 50 percent more memory than the equivalent MOL program, then memories 50 percent larger for all one hundred computers have to be purchased. The efficiency of a tactical military compiler is usually measured by determining the increase in memory and execution time of the object code it generates compared with an expertly-coded MOP program. A high efficiency, usually on the order of 80 percent for both time and space, is crucial to the acceptance of a tactical military compiler. Therefore, the typical tactical military compiler has more extensive optimization features than most commercial compilers. There is usually more emphasis on local optimization than global optimization because this approach appears to offer the biggest payoff at the lowest cost and because the effect of many global optimizations can be obtained by appropriate modifications to the source code.

The task of creating a highly efficient compiler is facilitated because the usual emphasis on compilation speed is absent in the tactical military environment and because the compiler usually executes on a larger general-purpose computer rather than on the tactical military computer itself. This enables the use of optimization algorithms
What's Different About Tactical Military Languages and Compilers

that take a long time to execute and require a large amount of compiler storage.

Many tactical military HOLs contain features that allow the programmer to control the optimization that the compiler performs. For example, the programmer may specify that space optimization is more important in one portion of the program, and time optimization in another portion. This is desirable because the minor-cycle portion of a tactical military program may be executed 50 times more frequently than the major cycle portion, and unless the appropriate areas are delimited, the compiler does not have sufficient information for effective optimization.

Compilers for commercial applications are supplied by the computer manufacturer; tactical military compilers are often developed by the same organization that develops the operational software or are supplied by the military customer. While a commercial compiler may be used in hundreds of installations, a tactical military compiler may be used by only one or two organizations. The cost of tactical military compiler development can therefore be a significant portion of the total tactical military software cost. Considerable attention has been paid to methods of reducing compiler cost; in particular the meta-compiler approach has been under study. Although the meta-compiler approach has shown considerable promise, most tactical military compilers have followed relatively conventional designs. Usually the code generation module is written so that it may be easily replaced when a compiler for another tactical military computer is needed.

Because it receives much less usage than a commercial compiler, a tactical military compiler would be very similar in reliability to the first release of a commercial compiler unless special testing precautions are taken. Anyone familiar with the lack of reliability in early releases of commercial compilers can appreciate how much effort must be expended in tactical military compiler testing. Even with extensive testing, those responsible for the validation of operational tactical military programs pay considerable attention to the object code.

FUTURE LANGUAGE AND COMPILER TRENDS

The use of HOLs in tactical military software development will continue to grow, largely because of pressure from the military customer to reduce development cost, simplify maintenance, and provide visibility into software behavior. The recent rash of overlapping language definition efforts will come to an end and language standardization will become more important, again largely because of pressure from the military customer. The language or languages that prove to be successful in large, significant tactical military software developments will have the greatest chance of being selected as the industry standard, regardless of any theoretical virtues or faults. This will parallel the Air Force's selection of JOVIAL as its standard command and control language.

The greatest improvement in languages will come in the features that aid development of reliable operational software. Diagnostic directives, required redundant statements, and compile-time limit and validity checks will be added to existing languages and their use will become mandatory in an effort to reduce debugging and validation costs. The current emphasis in language design will thus shift from defining succinct and elegant ways to describe the procedures that should be executed. Instead, the emphasis will be on creating forms that more clearly (and even redundantly) describe those procedures.

The cost and long development times for tactical military compilers will be the chief inhibiting factor in the use of HOLs in tactical military applications. Efficiency considerations will grow less important as more improvements are made in optimization methods and as the cost of computer hardware itself continues to become cheaper.

Finally, the tactical military computers themselves will become more suited to the HOL approach. Initially, this will take the form of computers whose organization and instruction set facilitate the writing of compilers. Ultimately, the tactical military computers will execute an HOL directly, obviating the need for a compiler.