What Will Programming Be Like in 1998?

N. M. Demos
Performance Software Group
Baltimore, Maryland 21228

Programming has come a long way in the last ten years. By trying to look at the next ten years, programmers and managers can assess the skills and tools they will need in order to be productive in the next decade. 4GL's as currently implemented on the HP3000 today are only the beginning. The ideal programming methodology of the future will include a very powerful application generator that will. (1) Be able to draw on all the previously written application code where appropriate. (2) Be able to provide the programmer with easy to use design templates that he can use to easily generate his program, including screens. (3) Be able to build and modify databases. (4) Be able to automatically create utilities to maintain the databases. (5) Be able to check every programmer action for internal consistency as he enters it. (6) Be able to help the programmer generate test data. (7) Be able to provide all levels of documentation automatically, tailored to the installation's requirement and standards.

For the first time since the advent of assemblers, there are things happening to radically alter the way programming is accomplished. Although these tools are not complete yet, there is enough available so that we can guess how these tools will operate. The programmers and, particularly, the programming managers and MIS directors who fail to learn about these concepts and tools will be ill prepared to perform their tasks in the not too distant future.

One dictionary definition of programming is, "To provide a computer with a set of instructions for solving a problem". That is a very broad definition and would cover such trivial tasks as setting up Job Control statements, etc. We think of programs as not only solving problems, but also as accomplishing a non-trivial task and being easily repeatable. Up to this time most programs have been coded by specialists in programming, whether in-house or purchased. When one considers the computer tasks that need to be accomplished, it becomes clear that quantum leap in program productivity is required. This will require a combination of the following:

1. Much more extensive use of purchased software. This means that the users may have to be more flexible in their requirements and that the software will have to be better and more adaptable to the individual user environment.

2. More user "programming". New tools and an increasing computer skill level will make it feasible for the user to do the simpler programming tasks, such as data extracting and formatting for reports.

3. More sophisticated programmer tools. New tools, of the type that are just beginning to appear on the market, will make the

initial programming task and maintenance programming much more productive.

It is to this last point (and partially to the second) that this paper is addressed.

There will be new methods of communicating with the computer, such as voice recognition, that will make the programmer's job easier for his own interaction with the computer, but may make programming more difficult, because the user will also want to use this technology. We do not see the major productivity improvement being in this area. Rather, we think the major new tools will employ similar techniques that have been used to the benefit of the user. Using data bases to organize and store data relevant to the programming task, i.e. data dictionaries, is just coming into wide use. The dictionaries are getting more sophisticated. They can or will soon be able to store in an organized fashion not only information describing the data structure and location of data fields, but also modules of code. They will be active versus passive dictionaries. That is, changes in any definition in the dictionary will trigger a change to all related (dependent) definitions where this relationship is known.

Application generators closely integrated with dictionaries will give the programmer the ability to conveniently extract all relevant previously done programming work and integrate it into his current program. At the same time, it will be easy for him to enter his new program modules into the dictionary for reuse by him or use by others.

The application generator, to be most effective, will be front ended by a very well designed and integrated prompting and editing system. It will include the capability to interactively run his code and have an extensive debugging system that he can move in and out of easily. An extensive help system will save time searching through reference material. Syntax checking will occur on the fly, so that corrections can be made immediately. Internal consistency checking will prevent the entry of unused or redundant code. Extensive prompting will optionally be employed to assist the programmer in insuring that all necessary parts of the program are completed.

This type of facility will not only apply to the program itself, but also to its production environment. The application generator will already have, as part of the information necessary for programming, most of the information needed for execution of the program. This will not only apply to the program itself, but to the entire system of which the program is a part.

If we look at the way a programming task is accomplished in the future, we can envision someone sitting at a display with a keyboard and voice facility. It will probably have a mouse associated with it. The user of this system might very well have on the screen what we refer today has icons. He will manipulate data, code and tasks interchangeably, to fashion the solution he needs to complete.

In this new way of doing things, not only will there be only one occurence of each piece of data, but data derived from other data will be defined as such,

so that a data definition may consist of algorithms operating on the basic data. Information will always be based on the most recently input basic data.

This methodology is technically feasible today, but there does not exist a comprehensive system that can be used now to achieve this kind of environment. Most of the things discussed above involve applying the present technology not only to the organization and retrieval of the data itself, but also to the algorithms used to derive it.

There are already several systems in place that are significant steps in this direction. A prime example is the system used on Apple's Macintosh.

The kind of systems we are describing here have another advantage that may not be obvious. They will impose on the programmer a discipline that will make his programs much easier to maintain, particularly by another party.

HP's recently announced Virtuoso is an application generator that is an example of a tool that will give the programmer the capability to integrate into a new program modules of source code previously entered into a dictionary. Although HP is supplying Virtuoso with a "sample" library of COBOL routines, the programmer can build his library with source modules in any programming language. Virtuoso is meant to be a tool to build a large application system with standardized modules integrated in a standard way. Thus it is an example showing one of the capabilities mentioned above for the new programming methodology.

The New Wave environment is another facility that shows us how programming will be accomplished in the future. "The HP New Wave environment provides the tools that support the user's everyday function and it is also the user's view into the entire office systems network." New Wave looks very much like Apple's MacApp for its Macintosh computer. The concepts of both MacApp and New Wave envision the following concepts:

1. Object oriented programming. An object is a piece of data, a collection of interrelated data or a task operating on data.

2. A common user interface. All programs will have the same look and "feel" to the user.

3. A common structure. All programs will be designed to the same standards and employ similar methodology. The techniques in this area will be similar to the application generators mentioned above.

4. Data and code will be stored only once and be employed as appropriate throughout the system. For instance, if the user changes a spread sheet entry, not only will the affected numbers on the spread sheet change, but the correct effect will be automatically made to any other place in the system where data is used or derived from the changed entry. An object may be not just a data item, but a combination of data items and/or algorithms that operates on the data.

The goal of all this is to give the user an integrated system. HP uses the word "seamless" to describe this level of integration. That is the user will not be aware of running a particular program - he will be accomplishing the task in the most effortless, accurate way possible. At the same time, the programmer will be able to concentrate only on completely unique code and data elements for new applications or application extensions. If these new techniques work correctly, he will be able to be confident that any change to a routine will automatically be propagated throughout the system. Of course, any bug he introduces will also; powerful tools can sometimes result in powerful errors.