A Survey of the Characteristics of Very High Level Languages*

Regis S. Loffman
Energy Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831

ABSTRACT

Although there is no consensus on the characteristics of very high level languages (VHLLs), ample evidence exists that users of such languages have several expectations. As a result of an extensive survey of the literature and communications with researchers, it was concluded that six characteristics of VHLLs are desired. These are that the language (1) be nonprocedural, (2) allow implicit referencing, (3) provide a good interface between user and computer, (4) allow for verification, (5) be extensible, and (6) provide the means for better data representation. It was also concluded that no existing language has all six characteristics.

INTRODUCTION

The field of VHLLs is a new research area, and, as such, new knowledge and understanding are being gained continuously. However, because of their relatively recent inception, there is a diversity of ideas and concepts about VHLLs. The names or labels for them are equally as diverse. A characterization of VHLLs is important because it will provide a basis for common understanding, which will facilitate communication. Improved communication will result in the sharing of research and ideas, thereby clarifying future directions. This paper presents some characteristics of VHLLs as found in the current literature and research activity as a preliminary step in the process of defining VHLLs.

Currently there seems to be no consensus about what a VHLL is or what the criteria are by which one determines whether something (presumably a programming language) is a VHLL. There appear to be many similar ideas with different variations or labels. It is acknowledged that no one language or environment is capable of doing all the necessary or desired tasks, and therefore VHLLs must be extremely flexible.

*Research performed at Oak Ridge National Laboratory, operated by Martin Marietta Energy Systems, Inc., for the U.S. Department of Energy under Contract No. DE-AC05-840R21400.

CHARACTERISTICS

Programming languages have always been described by generic labels. For example, the "generation approach" ranges from first-generation languages (1GL) through fourth-generation languages (4GL) and is now entering the fifth generation. First-generation languages were a very primitive means of utilizing the first computers. Language may perhaps be a too-generous label since it dealt with computers at the hardware level. Fourth-generation languages cover a wide range of capabilities, including more English-like querying of data, report generating, and graphics. They are less procedural than previous generations and are more suited to use by nonprogrammer professionals. Fourth-generation languages are often associated with data base management systems. The two generations in between the first and fourth include assembly languages and languages such as Fortran, COBOL, and PL/1. Fifth-generation languages are the up-and-coming generation; perhaps they are synonymous with VHLLs.

Another programming language label is that of "high level language." High level languages are languages that translate one programming instruction into several assembly-level instructions. They are parallel to the third and fourth generations mentioned above.

Two more advanced types of languages by this scheme of categories are VHLLs and expert system languages. At this point, distinctions become fuzzy because there is less consensus about the true natures of these languages. The rest of this discussion will be focused on characteristics of VHLLs; however, an underlying set of questions is whether expert systems are synonymous with VHLLs, whether VHLLs are required to write expert systems, or whether expert systems require the use of VHLLs?

Three basic characteristics of VHLLs are consistent throughout current literature and research activity. These are that a VHLL should (1) employ declarative statements, (2) allow implicit referencing, and (3) promote the communicability of knowledge.[1] The first characteristic refers to the language being nonprocedural as opposed to traditional languages, which are procedural (i.e., Fortran, COBOL, etc.). The second allows for implicit reference through the use of inheritance capability; that is, a particular characteristic is associated with a set (or class) of objects wherein all subsets inherit the same characteristics without the characteristics having to be repeated. The third characteristic refers to the capability for the user and the computer to communicate. The knowledge is of no value if it cannot be interpreted either by the computer for processing or by the user in understanding the result.

After an extensive survey of the literature and several personal

[1] Carbonelle, J., Carnegie-Mellon University, personal communication, Mar. 3, 1986.

communications, it is evident that three other features are desirable when considering VHLLs. These are that the language (1) verify correctness, (2) accommodate change, and (3) deal with data in new ways. The verification of correctness refers to the capability to check syntax and semantics. Syntax refers to correct format and spelling of commands. (Forth does little syntax checking, but Brodie[2] contends that syntax checking as commonly perceived would limit the freedom and flexibility provided by Forth.) Semantics checking can occur at two different levels, internal and external. Internal semantics is concerned with whether what is being said adheres to the rules of the language. External semantics pertains to whether the system is solving the right problem[3]. This type of verification is not trivial and perhaps implies the need for an artificial intelligence capability within the language.

The second desired capability, extensibility, or the ability to accommodate change, has always been important and requires that the language be flexible. Forth embraces this capability by building new words based on existing words in its dictionary. The third desired capability, the ability to deal with data in new ways, is similar to extensibility in that data requirements and ways to express data are unpredictable and dynamic.

These six characteristics are not all present in any one existing language. They are currently implemented by flexible data structures, abstract data types, knowledge engineering, artificial intelligence, expert systems, graphics, and data base management systems.

Newer programming techniques are focused on different aspects of these characteristics. "Access-oriented programming" enables procedures to be invoked which are triggered by data activity. "Object-oriented programming" (e.g., Smalltalk) groups data into objects. Objects are characterized by a type of behavior which is inherited by subclasses. "Logic-oriented programming" (e.g., Prolog) is concerned with nonprocedural representation of knowledge and is used in inference situations. "Function-oriented programming" (e.g., Lisp) is concerned with transformations applied to data. These transformations are based on mathematics providing a sound basis. These techniques are primarily used in the area of artificial intelligence and expert systems; however, as stated previously, it is not clear what the relationship is between VHLLs and expert systems/artificial intelligence.

---

[2] Brodie, L., Thinking Forth, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1984.

[3] Martin, J., Fourth-Generation Languages, Volume I, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.

CONCLUSIONS

Whatever direction VHLL research takes, the characterization of VHLLs is an important step because it will allow a framework for communication. Based on current understanding, six basic features of VHLLs are apparent. These are that the language (1) be non-procedural, (2) allow implicit referencing, (3) provide a good interface between user and computer, (4) allow for verification, (5) be extensible, and (6) provide the means for better data representation.

Currently, no language satisfies every VHLL characteristic, and no common understanding of the requirements for a VHLL exists. Future languages and tools will either be designed for specific purposes or will be combinations of several tools. Both of these approaches has merit. The first will produce powerful tools; however, they will be limited in scope. The second will produce tools with broader application scope but less power for a particular task.