HP VEE 4.0

HP VEE 4.0, Hewlett-Packard's Visual Engineering Environment Version 4.0, is available. Is it really compiled? How fast is it? How does it compare with its closest competitors? Here are some answers.

The specs for the latest version of HP VEE call for a machine with at least a 486/66 and a coprocessor. HP recommends that a Pentium 90 or higher be used with 24 MB of RAM for Windows 95 or 32 MB of RAM for Windows NT. It was tested on a dual-Pentium 100-MHz PC with 98-MB RAM running NT server 4.0 service pack 3 and a 486 PC with 30-MB RAM running Windows 95.

This is a short review: Good news tends to be compact. A year ago, HP VEE looked like a good idea but just not quite there. With this release, HP VEE is a major contender in the instrumentation and data-acquisition graphical programming software arena.

Scalability and Management Tools

HP heard the call for scalability and management tools. With this product, new configuration management enhancements were added either directly or through compatibility with industry-standard tools. New organizational aids provide code and task management. You still will need clear thinking and project management tools for really large efforts, but at least there is help in the tools.

Documentation and Presentation Aids

Intrinsic to the HP VEE syntax is the capability to print a meaningful diagram for each level of the system. Until now, printing was tantalizingly close: The language syntax was amenable; the printer support was not. No longer. HP supports all kinds of printers and in smart ways. Additionally, HP VEE assigns logical map squares to the diagram. A text report cross-references the diagram with the salient characteristics of each diagram symbol.

Programmer Helpers

HP VEE adds programmer aids that answer the questions "Where did I put that function? It is a function—isn't it?" The new search engines make short work of a hunt in the spaghetti. Selection of syntax objects continues to be menu-oriented. In past versions of HP VEE, it could be an adventure to find the menu that had what you wanted.

Sure, you could use Help. But by the time you head for the "wimp's way out," frustration already has won. The new organization is much cleaner. Actually it is intuitive.

Instrument Drivers

There are two CD-ROMs worth of instrument drivers. This review team had neither the time nor the stamina to look at even a reasonable percentage of them. Enjoy.

The Real World of Microsoft

Love them or hate them—they're here. With UNIX workstations and high-end PCs supplying all the features and compute power for data acquisition projects of great scale, the price advantage of the PCs is winning. Say PC, and you already have said Microsoft. HP VEE is getting with the program. It's not completely with the program yet, but then neither is Microsoft.

HP VEE allows for calling DLLs, and it goes one better: You can construct an ActiveX control—Microsoft's way for an anybody-can-call-me subroutine. At least that's the way it should work. It actually does work, mostly.

Installing some run-time features may require a professional systems administrator to bail you out. There is no install wizard yet.

HP's ActiveX controls can be embedded in Excel, Word, Paradox, or anything that supports OLE 2.0. This integrates HP VEE data acquisition technology in a nearly seamless acquisition, analysis, and publishing system.

Access to Microsoft data bases and other Back Office services is not directly available. Integrating these services into a HP VEE application is straightforward. A simple interface written in any language of choice, such as C, C++, Visual Basic 5.0, or Borland's environment, can produce a DLL. You call the DLL from HP VEE.

For data-base access, you must be bilingual. In the next release of HP VEE, who knows.

Performance

HP is proud of the performance improvements it has achieved. It should be. In most data-acquisition systems, the instruments are the performance bottleneck. The old HP VEE often could delay the slowest instrument. The new HP VEE is, if not fast, respectable. It is "compiled." It is about 50 times faster than it was before. But appearances can be deceiving.

HP VEE vs LabVIEW and Visual Basic

On its own, HP VEE is a very capable software package. It can do many things and connect to many other software packages. It is state-of-the-art graphical programming.

It does not exist in a vacuum, however. Reviewing HP VEE without mentioning National Instruments' LabVIEW is like critiquing Ford without referring to GM. Yes, it can be done, but it makes no sense.

Regarding performance, remember we stated that HP VEE has improved by 50 times. That's the good news. The not-so-good news is that it still is appreciably slower than LabVIEW except when performing built-in tasks.

When comparing library LabVIEW icons vs built-in HP VEE functions, we found the speed of HP VEE and LabVIEW comparable. Sometimes HP VEE was faster, sometimes LabVIEW. When

you add in the wiring, the actual programming that you would more than likely do, LabVIEW still is clearly faster. But appearances can be deceiving.

“Speed kills” was an old cliché. In software, the new cliché is “Lack of speed kills.” Just how important is performance? Certainly, it is a measure of quality, but it is not the only measure. There is a class of problems—albeit small—that will require the performance of LabVIEW over the performance of HP VEE.

In almost all cases, either will have the performance to do the task. Most likely, if HP VEE is not fast enough, LabVIEW will not be either. Most tasks requiring high performance are butting against the real-time OS question, not an HP VEE vs LabVIEW issue. The more serious concern is whether either package can break the real-time barrier. That’s where almost all the performance questions really lie.

Both packages are fast enough to do many things. Depending on what you want to do, either or perhaps neither can perform. The real issue of speed comes down to display. How much interaction is required? How fancy, graphically intensive are the displays? That will slow either program to a crawl.

And let’s not forget Visual Basic, the other competitor for instrumentation and data acquisition. Anything with the power of Microsoft behind it certainly qualifies as a 600-lb gorilla. The power of Visual Basic lies in its third-party libraries. It is certainly not graphical programming; and for anything but screen and report layouts, it is good old text. Still, it is another way of doing business and cannot be ignored.

The area of difference between HP VEE and LabVIEW is the architecture of the graphical programming. HP VEE customizes its icons with a label rather than changing the form of the icon as LabVIEW does. As a result, the LabVIEW programmer approaching HP VEE wonders where the differences are. The HP VEE programmer approaching LabVIEW wonders how they can ever remember what all the symbols mean.

The approach to flow structures in the two languages also is completely different. HP VEE is based on the flow chart: Everything you need to understand a layer or a level of abstraction is visible. Structures are constructed much the same way you would in a flow chart.

For example, in HP VEE, a case structure would have an if/then/else object with a bunch of ELSEIF tests. From each of these tests, a line would lead to the next step. LabVIEW embodies the structures as windows in the syntax. In LabVIEW, the case number would be constructed as an integer. The construction would be up to the programmer. Then, each of the cases would reside in a frame, all sharing the same screen window.

Only one frame of a case statement is visible at any time. This results in a completely different look and feel. In HP VEE, the test for the case is built into the dispatch structure. In LabVIEW, the construction of the case index can be separated from its use as a dispatch mechanism.

In LabVIEW, the programmer is assisted in focusing on the specifics of each case action at the expense of seeing all the cases in a single view. In HP VEE, you can spread the case actions across the sky.

While the syntax of HP VEE allows for the unstructured constructs, it is difficult and error-prone. HP VEE code tends to be as structured as LabVIEW code; the syntactic form is just different. A picture is confusing in this case. To see the picture of a LabVIEW program requires a computer. This is not necessarily bad. It is simply an unavoidable outcome of the LabVIEW syntax.

HP VEE is ideally suited to develop applications with external instruments accessed by GPIB or RS-232 interfaces. And let's not forget Ethernet as the latest interface for accessing equipment.

In past reviews, we have noted that LabVIEW and HP VEE are not the same. That remains true. LabVIEW arose, in part, from the need to control data-acquisition boards that were inside the computer running the LabVIEW program. HP VEE arose, in part, from the need to control instruments that were, or could be, stand alone. The performance needs were different. They remain different.

None of this is to say that either tool is in some way limited. It just points out how the differing architectural design approaches for data-acquisition systems led to differing importance on performance.

HP VEE does many things at a higher level than LabVIEW does. There are two classes of higher-level functionality. There are data-analysis capabilities that naturally complement external instruments, such as automatically combining the data and time samples acquired from an oscilloscope and displaying them on a graph or taking their spectrum and displaying them with the correct units. This would take many sub-VIs and a bit of code in LabVIEW. HP VEE has the capability to easily log data to a file for later recovery, processing, or display.

While the full complement of bit- and byte-level functions is available, clearly they are there to get you out of a jam, not as a normally used capability. LabVIEW, coming from a background of controlling data acquisition boards in the computer, has all the low-level control needed for that task.

HP VEE operates on higher-level constructs. In LabVIEW, there is a close mapping between LabVIEW data types and those of traditional procedural languages like C and PASCAL. HP VEE has some of the traditional constructs. Missing are unsigned integers, 8-bit integers, and the like. HP VEE adds various waveforms and a full complement of arrays with functions for treating them as arbitrary arrays, matrices, or determinants.

If you are working primarily with instruments external to the computer and you want the kind of data acquisition, processing, and presentation that HP VEE supports plus a shorter learning curve, then go with HP VEE. It has a lot to offer, and you probably will be much more satisfied with the over-all solution than a hybrid GPIB/LabVIEW solution.

On the other hand, if you already are well versed in LabVIEW and predominantly using internal measurement boards, you probably will be more satisfied with LabVIEW. Remember that a key element in this comparison is the speed difference between HP VEE and LabVIEW.

Generally speaking, when using external equipment, the speed bottleneck is the remote instrument, and faster software is just going to sit around and wait anyway. Also, it usually is

true that buying all your software, hardware, and peripherals from the same company will result in better support. So if the majority of your inventory comes from HP, you might as well make it 100%.

As systems integrators, JPL's Measurement Technology Center frequently writes software that will be transferred to and maintained by its customers. Often, we are asked what is the best language to write it in—Visual Basic, C, C++, LabVIEW, HP VEE? We answer that the software should be written in the language most comfortable for the person who will maintain it. That way it will be maintained.

In this release of HP VEE, much has remained. This is a good thing. There have been far too many times where a vendor has "upgraded" a product to an unrecognizable mess. That has not happened. What was good was kept.

In summary, pick your software and its backer—HP VEE or LabVIEW. At this point, you cannot go wrong with either. Or go with Visual Basic—not visual at all but it has the power of Microsoft behind it. But that's another story.

Disclaimer

This article was written by the authors as private individuals and not in conjunction with JPL.

About the Author

Edmund C. Baroth, Ph.D., conceived, developed, and is the Technical Manager of the Measurement Technology Center, part of the Measurement, Test and Engineering Support Section of the Jet Propulsion Laboratory. He holds a bachelor's degree in mechanical engineering from City College of New York and master's and doctorate degrees in mechanical engineering from the University of California, Berkeley. Dr. Baroth has received five NASA Certificates of Recognition and a NASA Group Achievement Award, has produced two patents and has one pending, and has been a faculty member at California State University. Jet Propulsion Laboratory, 4800 Oak Grove Dr., Mail Stop 125-177, Pasadena, CA 91109, (818) 354-8339. e-mail: ebaroth@jpl.nasa.gov.

sidebar/no art

Background on HP VEE

HP VEE is HP's visual programming tool for engineering problem-solving. It provides the capability to gather, analyze, and display data without conventional, text-based programming. Version 1.0 was released in 1991. Version 2.0, released in late 1992, extended its capabilities. Version 3.0 entered the market in February 1995 with major changes. The current version, 4.0, was released in February 1997.

Data-flow diagrams, or programs, are created in HP VEE by selecting and placing graphical objects on the screen and connecting them with lines. The program is a collection of icons, each of which executes compiled code to perform a function such as transforming data. They interact with the graphical user interface, communicating with instrumentation or other networked

computers, executing compiled code or system calls, or evaluating conditions which control program behavior.

These icons communicate with each other primarily by passing data back and forth to each other along “wires” drawn between them. In some cases, they communicate by passing control flags to indicate that events have occurred. These programming elements are predefined from a series of hierarchical menus or defined by the developer.

HP VEE implements a significantly different visual paradigm than does LabVIEW. Rather than divide the interface and the code on separate panels, HP VEE integrates the entire view of the application on one panel. The developer can promote interface elements to a panel view that serves as the general user’s application interface. This interface effectively restricts the view of the application to only those elements needed for input and output.

For development purposes, these interface elements are visible and functional within the program code itself. For expediency, many programs simply dispense with the panel view. Otherwise, it makes debugging easy by allowing the I/O to be viewed along with the program execution.

HP VEE has three types of menus from which to select:

Pull-down menus that allow you to access objects, perform actions, and set states.

Object menus that are specific to the selected object.

Cascading menus that are submenus available from the main menus and object menus.

Some menu options bring up dialog boxes. Menu options are accessed by clicking and dragging the mouse. Objects are manipulated easily in the same manner. They also can be collapsed into icons or double-clicked to reveal their Open Views that allow you to see and edit an object’s configuration. For instance, the number of input or output terminals may be set from an object’s Open View.

Managing code size and structure mainly involves reducing the complexity of a panel to a reasonable level. HP VEE allows a hierarchical decomposition of a problem by encapsulating groups of code elements into what they call a User Object or a User Function. In either case, the encapsulated code not only can be reduced to an icon on a panel, but also can be duplicated and called throughout the program, much like a subroutine.

Duplicating a User Object makes a separate copy of the code, which then can be modified to produce a slightly different version. On the other hand, User Functions store the code in a single separate location, allowing multiple copies of its icon representation to call the same piece of code. In this way, one change to the User Function code affects the change for every procedure that calls it. Both features allow the developer to save code in Object Libraries for later use.

If the desired functionality is not available through a built-in function or cannot be easily created with a User Object, the developer can link and call compiled code written in C, FORTRAN, PASCAL, or BASIC, but some restrictions apply.