AIR Project: 00176.001

Comparison of productivity gains among leading programming languages for test applications: LabVIEW, HP VEE, and C/C++

Usability Test Results

Lydia Volaitis, Senior Research Scientist
Joseph Dumas, Chief Scientist
Ellen Mastoras, Research Associate

American Institutes for Research
Usability Engineering Group
490 Virginia Road
Concord, MA 01742

October, 1997

Executive Summary

The Usability Engineering Group of the American Institutes for Research (AIR) completed a comparative evaluation aimed at identifying which of three software packages, namely National Instruments' LabVIEW, Hewlett-Packard's HP VEE, or the Microsoft Visual C++ programming language, offers the greatest productivity advantages when used to develop test and measurement applications.

The main finding of the study is that across all tasks, programming with HP VEE was 27% faster than programming the same tasks using LabVIEW, and 35% faster than programming with C++. On individual tasks, programming with HP VEE was up to 79% faster than LabVIEW and up to 79% faster than C++. In many cases, a greater number of participants were able to complete tasks using HP VEE than using LabVIEW or C++.

These results suggest that programming with HP VEE can provide productivity advantages, as measured by speed, over programming either with LabVIEW or with MS C++.

Number of completions and mean time to complete tasks using each software package.

	HP VEE		**MS C++**		**LabVIEW**	
Task:	# who completed (N=15)	mean task time (sec)	# who completed (N=11)	mean task time (sec)	# who completed (N=15)	mean task time (sec)
Init. funct. generator	14	306.67	4	444.82	5	453.6
Change wave shape	13	230.73	5	403.73	8	416.60
Set frequency	14	85.47	11	165.91	12	195.00
Set amplitude.	15	114.20	10	156.09	15	100.73
Set trigger source	15	54.80	11	134.64	12	259.80
Init. wave. analyzer	15	116.20	1	504.82	10	320.33
Capture wave - array	5	386.00	2	465.55	7	463.53
Display wave - array	15	67.07	6	322.18	14	113.73
Display wave graph	13	167.73	N/A	N/A	13	144.60
Code user prompt	13	185.60	7	318.27	14	298.60
Program limit test	1	508.13	10	382.91	6	434.20

Timestamp to ASCII file	4	444.93	1	471.27	6	490.80
Write data to file	11	•	1	•	3	•
Write status to file	10	•	1	•	3	•
Mean	**11.9**	**222.29**	**5.38**	**342.74**	**9.43**	**307.63**
	(75%)		**(49%)**		**(62%)**	

Introduction

The Usability Engineering Group of the American Institutes for Research (AIR) recently completed a comparative evaluation of three test and measurement software packages on behalf of the Hewlett-Packard Company. The study was aimed at identifying which of three software packages, namely National Instruments' LabVIEW, Hewlett-Packard's HP VEE, or the Microsoft Visual C++ programming language, offers the greatest productivity advantages when used to develop test and measurement applications.

AIR's Usability Engineering Group (Concord, MA) specializes in user-centered design and evaluation of software and other product user interfaces. Our primary methodology consists of task-based evaluation, in which paid study participants attempt to complete tasks that typical users of a product or software would normally perform. Typical measures for a usability evaluation include time to complete a task, the number of tasks completed, level of confidence or other subjective ratings. We perform statistical tests of the task times, ratings, and preferences, and supplement those analyses with the participants' verbal assessments of the products' usability, as well as with our own observations.

Our goal in undertaking this project was to design and conduct a test that would fairly evaluate and compare the three software packages. Consequently, we took great care to ensure that no aspect of the test -- evaluation tasks, test equipment, or participants -- was biased toward or against any of the three software applications. In the following sections, we describe our test design, methods, procedure, results and analyses.

Method

Test participants

Because functional tests for manufacturing represent one of the most challenging areas for test-specific application software, Hewlett-Packard chose the manufacturing test domain for evaluating long-term software productivity for the typical engineer.

We therefore selected test and measurement engineers and technicians working in the manufacturing domain to participate in the study.

All participants were contacted and screened by the same professional recruiting agency. Potential participants were identified in a number of ways including: names provided by Hewlett-Packard; purchased lists; cold-calling local business listings; and referrals from other participants. Neither the recruiters nor AIR informed study participants that Hewlett-Packard was the sponsor of the test. Employees of the Hewlett-Packard Company were not allowed to participate in the study.

To be invited to participate in the study, the respondent had to:

- work for a company that manufactures automatic test systems and whose primary purpose is the design or configuration of such systems;

- develop systems primarily for use within their own or a sister company;

- be an electrical engineer or technician responsible for the development of test systems used for testing electrical components or assemblies in a product development or production environment, rather than, for example, process control or scientific data acquisition;

- use either VXI or similar modular systems; GPIB or HPIB; or PC plug-ins such as instruments on a card inserted into a personal computer or an accessory chassis;

- be experienced in using either HP VEE, LabVIEW or C/C++ software for test and measurement programming.

AIR administered additional screening criteria. Because this was a test of productivity, and not a test of initial ease of use, we did not want programmers new to any of the software applications to serve as test participants.

More importantly, within each group we wanted study participants to have comparable expertise in programming. Then if the results of the test favored one software package, we would be able to say that it was not because engineers in that group were more experienced than the others. Because it was difficult to assess experience based on the participants' subjective ratings of themselves, we devised a post-hoc method of assessing participants' expertise. In this analysis we took all of the following factors into consideration:

- the participant's self-assessment as a programmer of HP VEE, LabVIEW or C/C++
- the participant's familiarity with instrument communication
- whether the participant took a course in a software application (also how long the course, and how long ago)
- experience in using the software to control laboratory instruments
- self-assessment of familiarity with VXI*plug&play* drivers
- self-assessment of familiarity with VXI
- number and length of programs written using the software package in the past year
- AIR test administrator's assessment of participant's expertise with the software

• AIR test administrator's assessment of participant's expertise with VXI*plug&play* drivers[1]

In total, forty-one test and measurement engineers participated in the study: fifteen were experienced users of LabVIEW; eleven were experienced with C/C++[2]; and fifteen with HP VEE. Their expertise is summarized in Table 1.

Table 1. Categorization of participant expertise with each software package.

	Category of Expertise		
Software	**beginner**	**intermediate**	**expert**
HP VEE (N=15)	33.33%	53.33%	13.33%
LabVIEW (N=15)	36.36%	45.45%	18.18%
Microsoft C/C++ (N=11)	6.66%	66.67%	26.66%

These data show that the main difference among participants in the three groups was that a greater number of C/C++ experts participated in the test, relative to HP VEE and LabVIEW programmers.

Finally, if any participant could not successfully complete five of the fourteen programming tasks correctly in the time allotted, we did not use his data in the analysis, and replaced him with a more experienced engineer[3].

Tasks

We faced two challenges in selecting tasks for participants to perform during the study. The first challenge was to select tasks for the evaluation that accurately reflect everyday uses of the software. The second was to ensure that the tasks selected did not favor any software package or application over another.

AIR received a preliminary task list from an HP technical consultant. We then sent this list to an independent (i.e., not from HP) test and measurement industry expert to (1) approve it as representative of the actual usage of the software for the manufacturing domain, and (2) to look for biases for or against any of the three software applications or packages. This consultant also selected and provided us with equipment to use during the testing, again ensuring that the equipment selected would not bias the evaluation. A second HP technical consultant set up and configured the equipment, and served as a subject-matter consultant throughout the study.

The three main task scenarios we included in the evaluation were as follows:

[1] None of the participants had used VXI*plug&play* drivers prior to the test.
[2] It proved to be quite difficult to locate engineers experienced in programming test and measurement applications using C/C++ both in the Concord and Palo Alto areas.
[3] We replaced a total of six participants including 3 LabVIEW, 2 C/C++, and 1 HP VEE programmers.

- Use the software to communicate with a standalone function generator to create and display an output signal of a particular frequency, shape and amplitude. Then capture the waveform using a VXI waveform analyzer.

- Perform a limit test on the captured waveform. This includes writing code to prompt a user to specify a maximum voltage. Then write code to compare that voltage to each of the values captured with the waveform analyzer. If any value exceeds the user's input value, the test fails. Display the words PASS or FAIL accordingly.

- Create an ASCII file containing three kinds of data: the date and time from the computer's clock; the data samples collected by the waveform analyzer; and the PASS or FAIL status from the limit test.

We partitioned these three main task scenarios into fourteen smaller tasks. Each engineer or technician attempted to complete the tasks on the one software application with which he was most experienced[4].

Software and Equipment

HP VEE (version 4.0), LabVIEW (version 4.1), and Microsoft Visual C++ (version 5.0) software packages were installed on a PC running Windows95.

Participants were required to complete the tasks using VXI*plug&play* instrument drivers, which had been pre-configured for them.

We decided to use VXI*plug&play* instrument drivers in order to ensure that the chosen I/O software layer did not present an advantage for any specific application software. This presented a problem in using PC Plug-in cards in the test. PC Plug-in cards were eliminated from consideration because HP VEE, LabVIEW and C++ use different methods of communication to access the same PCPI card, which might introduce a variable not specifically linked to the programming languages being tested.

Engineers can use application software to talk directly to instruments or use drivers. Where speed and functionality allow, engineers prefer the ease of use of instrument drivers. This created another potential bias. LabVIEW in Windows95 can use three types of drivers: LabVIEW (now called GWIN VXI*plug&play*) drivers, LabWindows drivers, and WIN95 VXI*plug&play* drivers. HP VEE in Windows95 can work with three types of drivers: VEE drivers, LabWindows drivers, and WIN95 VXI*plug&play* drivers. Microsoft Visual C++ in Windows95 works with LabWindows (recompiled in 32 bit) drivers and WIN95

[4] We originally attempted to conduct the study as a within-subject comparison, where each participant would perform identical tasks on two different software applications. However it proved to be too difficult to find participants with equal expertise in two software packages or applications. Therefore we changed the study to a between-groups comparison, where each participant used only one application.

VXI*plug&play* drivers. All the test equipment we used had a WIN95 VXI*plug&play* driver available.

Connected to the PC were the two pieces of equipment used to generate and capture the test signals: a Hewlett-Packard 33120 standalone function generator, and a Tektronix VX4240 waveform analyzer card that was mounted in a cardcage.

Procedure

All test sessions were held at AIR's Usability Labs in Concord, MA, and Palo Alto, CA. We used the same software, procedure, and test equipment in both locations[5]. The same two AIR test administrators conducted the sessions in both cities.

Each engineer participated in a one-on-one session in which he attempted to complete the twelve tasks using the software package with which he had most experience.

The test sessions included the following steps:

- the participant read and signed a participation agreement
- the participant completed a pre-test questionnaire verifying his qualifications to participate in the test
- the administrator introduced the participant to the study and introduced him to the test equipment
- the participant performed a practice task to get used to the testing environment
- the participant attempted to complete the 12 evaluation tasks
- the participant completed a post-test questionnaire regarding his impressions of the test and of the software

The test sessions were held in a testing room with the participant and test administrator seated beside each other in front of a work table. The computer monitor, function generator, and cardcage containing the waveform analyzer were all on the table in the participant's view.

An AIR test administrator had the participant complete one task at a time. The tasks were printed on cards and placed in view of the participant. The participant read each task out loud, and then, once he understood it, attempted to complete it. Each of the tasks built on the previous one, so the participant was actually constructing a large program one component at a time. We set a time limit of eight minutes to complete each task. We had working solutions to each of tasks written in C++, LabVIEW, and C++, that had been coded in advance by an HP technical consultant. If the participant did not complete a task in eight minutes we marked a Time-out for that task, then directed the participant to load, study, and run the pre-coded

[5] The only difference in the setup was that we used a twenty-one inch monitor for testing in Concord, MA and a seventeen-inch monitor for testing in Palo Alto, CA.

program so that he could programming subsequent tasks[6]. The administrator used a stopwatch to measure the time it took for a participant to perform each task.

We did not provide participants with manuals or other printed documentation, nor did we provide them with verbal help or hints for completing the tasks. We did allow them to use any available on-line help. We did this to equalize test conditions for all participants. We told them that help for LabVIEW and HP VEE (including help for the instrument drivers) was available from within the applications; and that help for the C/C++ instrument drivers was available from the Windows desktop.

Our main measures of usability included the time to complete each task, and whether the participant completed the task within the allotted time. We report the results below.

Results (general)

In the next section, we describe the global issues that we feel slowed participants down and impeded them from completing tasks with each of the software packages.

LabVIEW

One of the most problematic aspects of LabVIEW has to do with the presentation and organization of the function call palette. We identified several aspects of this palette that increased LabVIEW programmers' time to complete tasks, particularly for the first seven tasks in which participants had to select function calls from the palette to configure the function generator and waveform analyzer.

First, we observed that there are so many functions in the palette that participants needed to drag the window to the top of the screen to see the whole list at once. A second problem is that the icons representing different functions all look extremely similar. Therefore, the best way to distinguish among the icons was to look at the text in the title bar at the top of the function call window. However, we noticed that it was difficult for programmers to look at this title bar without losing their place in the function call list. Finally, the lack of space between each icon slowed participants down while they searched through the list.

Another global issue with LabVIEW relates to the mechanics of wiring objects. For one thing, identifying the terminals of the VI objects proved time consuming. The on-line help explanation for wiring was useful. However, it appeared to us that it was difficult to wire objects *without* referring to help. This resulted in yet another window on the screen (see below), and longer task times as participants continually referred to the wiring diagram within the help window.

[6] If the participant was close to completing the task at 8 minutes, we let him continue a little longer. We felt that completing the task was a better measure of the software's usability than marking a time out if the participant was close to completing the task.

In addition, the wiring of one object to another also increased task times. The difficulty lay in the extra care needed to draw the wire in LabVIEW. In addition, participants needed to select the exact location on the VI icon in order to connect to the proper terminal.

A third global issue with LabVIEW involves window manipulation. The monitor screen typically became very crowded with the number of windows participants had open at one time. The excessive number of windows is caused in part by having separate windows for the diagram and front panel of the program. Additionally, in order to select a function call, participants had to go through two cascading menus, thus opening two more windows.

Finally, selecting a function call became even more complex because the cascading menu disappears when users move the cursor away from the window. To avoid this problem, users can click on the upper left of the window in order to "tack" it onto the screen. However, many participants did not always make use of this feature, so the function call window would disappear before they found the correct call.

C/C++

We identified two major reasons for the longer task times of C/C++ programmers. First, the increased times reflect the programmers' needs to refer frequently to the instrument driver help files in order to incorporate the correct syntax into their code. Typically, the programmers cut and pasted code from the help file into the Visual C++ workspace. Then they adapted the pasted code to the task, for example by typing in the correct address of the instrument driver.

Cutting and pasting from the Help file delayed them in completing the task in a number of ways. First participants often didn't know how to declare the variables they had cut from the examples. Second, they sometimes cut the examples in such a way that resulted in syntax errors. Accordingly, the second major reason for longer task times of C++ programmers resulted from their need to be more vigilant about syntax and punctuation errors.

HP VEE

In observing programmers using HP VEE, we did not identify anything as pervasive as confusing function palettes, or the errors inherent in writing code. Rather, we identified a number of smaller problems with HP VEE that contributed to longer task times for some tasks.

The most common problem with HP VEE was that many category labels and menu options were ambiguous to participants. Although we will discuss these instances in a task-by-task basis below, we offer the following example in this section for clarity: HP VEE programmers had some difficulty in displaying their captured waveform graphically simply because there were so many options available, and moreover, because the difference between the options were unclear. Specifically, we watched programmers lose substantial amounts of time trying to decide which of four display options -- XY Trace, Strip Chart, X vs. Y Plot, or Waveform (Time) -- they should select to display their data.

We noted that HP VEE programmers also had some difficulty with specific aspects relating to manipulating objects. One troublesome aspect was that some participants complained that objects take up too much space in the Main window. Others complained about the lack of a feature that allows users to group objects in order to move them at the same time. A third issue was that participants initially didn't realize they needed to bring up a pop-up menu to delete a line connecting objects. Instead, they tried to select the line and press Delete on the keyboard.

In terms of editing function panels, we observed that the following issues caused participants difficulty.

- Exponents are replaced with zeros when participants selected an input field
- Scientific notation seemed to be confusing to many participants
- Participants were not sure what the error box on the Panel tab indicated

Participants also complained about the lack of an "Undo" feature after they had added a component to their program that they realized was unnecessary.

Results (task by task)

In the following paragraphs, we report the results of our analyses of participant performance on each task. We present two kinds of data. The first is the number of participants who completed each task using the different software packages.

We also report the times it took participants to perform each task. There, we have included the results of an Analysis of Variance (ANOVA) of the mean times it took participants using each software package to complete each task. ANOVA is a statistical test that measures differences among group means. It returns an F value that has a probability, for example, $p < .01$, associated with it. A probability of $< .01$ means that there is less than one chance in 100 that a difference in the means this large could have happened by chance. In the data we present below, this would mean that differences between group means can be attributed to the software the participants used.

ANOVA does not tell you *which* means are different from each other. Therefore, we performed a second statistical test, namely, Fisher's Least Squares Difference (LSD), to show us which combinations of means are statistically different from each other. This type of test is known as a post-hoc test. In the results below, we indicate which pairs of means are statistically different with a probability of $p < .05$.

Task 1a. Initialize function generator

For this task, participants were required to use the software to initialize the HP 33120 function generator. This required them to locate and select the VXI*plug&play* instrument

driver, to locate the appropriate instrument from the instrument driver menu[7], and to program the function accordingly.

The table below shows how many participants were able to initialize the function generator (PASS) within the 8-minute time limit, and how many timed out (TIME-OUT), using each software package.

	HP VEE	MS C++	LabVIEW	Total
PASS	14	4	5	23
TIME-OUT	1	7	10	18
Total participants	15	11	15	41

As can be seen, a greater number of participants (14 of 15) completed the task using HP VEE than those using either of the other two software packages.

We performed an analysis of the average time it took participants using each software package to perform the task. The mean task times, the standard deviation, and ANOVA results are presented in the next table:

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	306.67	148.41	$F(2,40) = 6.39$,
MS C++	11	444.82	105.50	$p < .001$
LabVIEW	15	453.60	107.62	(statistically significant)

The ANOVA and post-hoc testing (Fisher's LSD) show that participants using HP VEE were able to initialize the function generator significantly faster than those using LabVIEW (32% faster), and than those using MS C++ (31% faster).

Task 1a. summary

HP VEE's speed advantage can be attributed to several causes. The first is that selecting a function call from HP VEE's tree structure menu was by far easier than selecting the same call from LabVIEW's graphical palette (see general issues, above).

In addition, HP programmers were further helped by not having to explicitly insert an instrument address into their code. In contrast, both LabVIEW and C/C++ programmers had difficulty with coding the instrument address. Although the entire GPIB address was printed on the task card, several participants were unsure what portion of the address they should use in their code.

[7] Part way through the test we discovered an unequal number of instruments in the LabVIEW Instrument Driver Library and in the HP VEE Instrument Manager. We immediately removed the extra drivers from the LabVIEW software and equalized the number of instruments in each software package. However, this may have contributed to longer task completion times for the first seven LabVIEW participants in two tasks involving instrument drivers.

Some LabVIEW programmers were confused by the default address displayed on the VI's front panel, which is similar, but shorter, than the address we gave to participants. Other participants made errors in typing the address.

Another aspect of LabVIEW that may have contributed to increased task times was the absence of the function palette when participants viewed the front panel of their program. Without the function palette, participants were unable to access the list of function calls for the function generator.

HP VEE programmers did encounter a few problems with this task, especially as this was their first introduction to the VXI*plug&play* drivers. For many participants, the first problem occurred after opening the Instrument Manager window and selecting the Function Generator option. Instead of selecting the Plug&play Driver button, they double-clicked on the Function Generator option or selected the Add or Edit buttons.

Some HP VEE programmers also were unsure how to add a transaction to the transaction box. Participants didn't realize that they needed either to single-click within the box to bring up a menu or double-click to add the transaction directly.

Task 1b. Change output waveform from sine to triangle

Participants had to locate and select the proper function panel, and then write the program to change the shape of the HP 33120 output waveform from a sine wave to a triangle wave.

The table below shows that 13 out of 15 participants were able to change the shape of the output waveform using HP VEE, 8 of 15 completed it using LabVIEW, and 5 of 15 completed the task using MS C++.

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	13	5	8	26
TIME-OUT	2	6	7	15
	15	11	15	41

The mean task times for each group are presented in the following table:

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	230.73	153.122	$F(2,40) = 8.49$, $p < .001$ (statistically significant)
MS C++	11	403.73	122.13	
LabVIEW	15	416.60	120.56	

HP VEE participants changed the shape of the output waveform 43% faster than those using MS C++ and 45% faster than those using LabVIEW. ANOVA and post-hoc tests showed the differences in the means to be statistically significant.

Task 1b. summary

As in Task 1a, HP VEE's advantage can again be attributed to easier access to the function calls.

It is worth noting that some HP VEE and LabVIEW programmers had difficulty indicating the preferred shape of the wave, particularly when they tried to pass a variable (an integer that corresponds to the waveshape) into the function call. Often, they didn't know which integer to use. Although LabVIEW's context-sensitive help indicates what integer to use, most LabVIEW programmers missed this information.

Task 1c. Set output frequency to 6 kHz.

Participants had to select the function from the instrument menu to change the output frequency of the function generator from 1 kHz. to 6 kHz. As can be seen in the next table, the majority of participants in each group were able to complete the task.

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	14	11	12	37
TIME-OUT	1	0	3	4
	15	11	15	41

The mean times to complete the task are presented below:

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	85.47	114.32	F(2,40) = 3.05, p.=.059 (marginally significant)
MS C++	11	165.91	84.133	
LabVIEW	15	195.00	155.90	

The ANOVA approached significance (p=.059), suggesting differences among users of the three software packages. The post-hoc test also suggested that difference in the time it took for HP VEE and LabVIEW participants to program a change in the output frequency, with HP VEE participants programming the function 56% faster than LabVIEW participants.

Task 1c. summary

HP VEE's speed advantage is again most likely due to the ease of selecting function calls from the menu.

Task 1d. Set waveform amplitude to 1 volt

All participants completed this task, which again was a matter of locating the correct function panel from the driver menu options and programming the function.

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	15	10	15	40
TIME-OUT	0	1	0	1
	15	11	15	41

As expected from the Pass/Time-out analysis, we found no significant differences among the three groups of programmers changing the waveform amplitude, as can be seen in the following table:

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	114.20	85.21	
MS C++	11	156.09	129.14	$F(2,40) = 1.28$, $p < .3$ (not statistically significant)
LabVIEW	15	100.73	50.05	

Task 1d. summary

There is no notable advantage for any software package. The only issue we noted was that a number of participants were unsure of whether the output amplitude would be expressed as peak-to-peak voltage.

Task 1e. Set trigger source to immediate

Again, the majority of participants were able to complete this task, with little difference in performance among the groups.

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	15	11	12	38
TIME-OUT	0	0	3	3
	15	11	15	41

Setting the trigger source appeared to be slightly easier for HP VEE and C++ programmers, as indicated by both groups achieving a100% task completion rate.

This performance was mirrored by HP VEE participants' fast task times, presented below. The ANOVA showed the means to be significantly different from each other, indicating that the differences in programming had to do with the software the participants used. Post-hoc tests showed that HP VEE participants changed the trigger source significantly faster (79%) than LabVIEW participants, and 59% faster than those using MS C++. The test also showed that C++ users programmed the task significantly faster (48%) than those using LabVIEW.

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	54.80	28.41	F(2,40) = 17.89, p <.0001* (statistically significant)
MS C++	11	134.64	40.58	
LabVIEW	15	259.80	149.13	

Task 1e. summary

There was little to note regarding this task. The average task times for programmers in all three groups continued to decrease, suggesting effects of familiarity and practice.

Task 1f. Initialize Tektronix waveform analyzer

This task required the participants to locate, select and initialize the Tektronix waveform analyzer driver. Here, there was a substantial difference in the number of individuals who were able to complete the task: All HP VEE and most LabVIEW programmers successfully located and initialized the waveform analyzer, however, only one C++ programmer was able to complete the task.

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	15	1	10	26
TIME-OUT	0	10	4	14
FAIL[8]	0	0	1	1
	15	11	15	41

The ANOVA showed a significant difference among the means. Post-hoc tests revealed three significant comparisons among the groups: HP VEE programmers initialized the waveform analyzer 77% faster than C++ programmers, and 64% faster than LabVIEW programmers. The data also show that LabVIEW programmers initialized the device 36.5% faster than those using MS C++.

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	116.20	85.08	F(2,40) = 40.79, p <.0001* (statistically significant)
MS C++	11	504.82[9]	70.78	
LabVIEW	15	320.33	146.88	

Task 1f. summary.

[8] One LabVIEW participant gave up on this task well before the time limit, and could not be convinced to continue. Hence, we scored his completion status as a Failure.

[9] The mean time for C/C++ participants performing this task is actually above the 8 minute limit, as we allowed the programmers even more time to try to complete the task.

This task required participants to locate the instrument driver for the Tektronix waveform analyzer. The waveform analyzer help files are presented in a different format than the function generator files. This new format appeared to be problematic for the C++ programmers, who seemed to rely on the help files to program their tasks.

Task 1g. Capture waveform as an array

This task proved to be quite difficult for participants in all three groups, as indicated by the numbers in the table below. The task required that participants write code to capture the triangle wave as an array of numbers. More LabVIEW participants completed the tasks than participants using either HP VEE or C/C++.

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	5	2	7	14
TIME-OUT	10	9	8	27
	15	11	15	41

The tasks times for all three groups were quite long. HP VEE programmers appeared to be somewhat faster than LabVIEW or C++ programmers, however, there were no significant differences among the means.

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	386.00	137.46	F(2,40) = 2.17 p .<13 (not statistically significant)
MS C++	11	465.55	65.84	
LabVIEW	15	463.53	120.45	

Task 1g. summary

Identifying which function panel would allow a user to capture data with the waveform analyzer was difficult for programmers using all the software. The quickest way to perform this task was to use the Capture & Return ASCII Samples function panel. However many participants were misled by the term “ASCII.”. In addition, the phrase “number of points” in the Capture & Return ASCII Samples function panel seemed to be ambiguous. This seemed to result in elevated task times across all software groups.

Task 1h. Display waveform as array

In this task, the participants were to display the waveform they had captured numerically in the previous task. For most participants, this task proved to be substantially easier than capturing the data as an array, particularly for those coding in HP VEE and LabVIEW.

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	15	6	14	35

TIME-OUT	0	5	1	6
	15	11	15	41

The ANOVA and post-hoc tests showed that both HP VEE and LabVIEW programmers displayed the waveform more quickly than C++ programmers. HP VEE programmers were 79% faster than those using C++. LabVIEW programmers were 41% faster than those using C++.

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	67.07	79.68	F(2,40) = 14.47, p <.0001 (statistically significant)
MS C++	11	322.18	180.51	
LabVIEW	15	113.73	111.59	

Task 1h. summary

The longer task times for C++ appeared to be due to programmers neglecting to specify the correct variable type (i.e., float) in their print statement. This was partly due to their strategy of cutting code from help without always understanding the code before they selected it.

Task 1i. Display waveform as a graph

For this task, participants had to display the waveform they had captured in the previous task as a graph. (We did not have C/C++ programmers perform this task, as it would have been impossible for them to complete the task in the allotted time).

The data in the table below show exactly equal performance between the HP VEE and LabVIEW participants:

	HP VEE	LabVIEW	TOTAL
PASS	13	13	26
TIME-OUT	2	2	4
	15	15	41

Because there were only two group means to compare, we performed a t-test on these data. And, as expected, the performance means and variances between the two groups were essentially the same, albeit with the LabVIEW task performance mean slightly, but not significantly, lower than the HP VEE mean:

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	t-test
HP VEE	15	167.73	156.66	t =.41, p <.7 (not significant)
LabVIEW	15	144.60	152.85	

Task 1i. summary

Any difficulty participants had with this task related to uncertainty about which display option to choose. For HP VEE, the correct choice was either Strip Chart or XY Trace. However, many participants also tried X vs. Y Plot and Waveform (Time). The difference among these options appeared to be unclear.

Task 2a. Limit test: code user prompt

After explaining the limit test to participants, we had them attempt to write code that would prompt a user to input an amplitude value.

Most HP VEE and LabVIEW participants were able to complete the task. However, it proved to be much more difficult for the C++ programmers.

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	13	7	14	34
TIME-OUT	2	4	1	7
	15	11	15	41

However, the analysis of the mean task performance times show a substantial advantage for using HP VEE for this task. HP VEE participants coded the user prompt significantly faster than those using LabVIEW or MS C++, with task times that were 38% and 42% faster, respectively.

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	185.60	117.30	$F(2,40) = 3.41$, $p < .05$* (statistically significant)
MS C++	11	318.27	157.16	
LabVIEW	15	298.60	159.09	

Task 2a. summary

This task was easiest for HP VEE programmers. First, many C++ programmers seemed to be confused by what syntax and function call would let them obtain input from a user. In addition, LabVIEW programmers tried to use the same function (dialog box) to prompt the user and obtain the user input.

Task 2b. Perform limit test

This task required participants to write code to compare an amplitude value to each of the values captured with the waveform analyzer. If any value exceeded the user's input value, the

test failed, and they were to display the word FAIL on the screen, otherwise they were to display the word PASS.

As can be seen, this test proved to be easiest for C/C++ programmers, difficult for LabVIEW programmers, and very difficult for HP VEE coders:

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	1	10	6	17
TIME-OUT	14	1	9	24
	15	11	15	41

The task performance time data reflect the above: Both C++ and LabVIEW participants programmed the limit test significantly faster than those using HP VEE. C++ programmers performed the task 25% faster than HP VEE programmers, and LabVIEW programmers were 14.5% faster than those using HP VEE.

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	508.13	48.43	$F(2,40) = 6.36$, $p < .005$* (statistically significant)
MS C++	11	382.91	123.13	
LabVIEW	15	434.20	94.70	

Task 2b. summary

The higher task completion rates and lower task times of C/C++ programmers relates to one of the positive aspects of textual programming. Graphical software applications require users to manipulate predefined functions to fit the actions they wish their program to perform. In most cases, the list of available functions is extensive enough to allow users to employ this method of programming effectively. However, Task 2b requires several steps: 1) comparing values, 2) determining whether the comparison passes or fails the test, and 3) displaying the pass/fail information.

The simplest solution includes a for loop which enables the program to continually compare the user defined value to each of the captured values. If any of the captured values is greater than the user defined value, a Boolean value is set to false. Once all the values have been compared, the program determines the value of the Boolean and displays “true” or “false” accordingly. This code closely followed participants’ mental model of how the program should work.

We feel that coding these same steps in HP VEE and LabVIEW isn’t as simple because programmers had to manipulate the predefined functions to match their mental model. Many participants had to revise their thoughts in order to fit the tools they were given. For example, some HP VEE programmers had difficulty including the equivalent of a Boolean value, so their programs displayed a pass or fail for each of the 400 captured values. Other HP VEE programmers didn’t know which tool to use to display the information. For this task, the tools

supplied in HP VEE did not conform to the programmers' mental model of how the program should work.

LabVIEW programmers had significantly lower task times than HP VEE programmers. As indicated above, HP VEE programmers had particular difficulty locating a tool that would act as a Boolean value. On the other hand, there is a Boolean option easily visible within the function palette in LabVIEW.

HP VEE programmers also had trouble displaying the result of the limit test. In HP VEE, programmers needed to connect a text object, with the appropriate message, to an alphanumeric display object. Displaying the limit test result is easier in LabVIEW because the message can be written directly on a dialog box.

Task 3. Write data to ASCII text file

The last task required participants to create a file containing the timestamp from the computer clock, the numeric data they had captured with the waveform analyzer, and the PASS or FAIL status of the limit test. We had them create the file using the Windows Notepad so that we could easily open it and check its contents.

We observed that participants had different levels of difficulty with different components of the task, so we coded their PASS/TIME-OUT status separately for each part of the task. As can be seen in the following three tables, HP VEE programmers had the easiest time writing the data to a file, but had a tougher time inserting the timestamp and the PASS/FAIL status into it. More LabVIEW than HP VEE programmers were able to write the timestamp and the PASS/FAIL status into the file, but appear to have had a more difficult time writing the actual data into the file.

Task 3a. Write data to file: put timestamp into file

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	4	1	6	11
TIME-OUT	11	10	9	30
	15	11	15	41

Task 3a. summary

One of the most difficult tasks for HP VEE participants was writing the current date and time from the computer's clock to a file. This required a rather circuitous path from the Device menu => Built-in Functions => Math Functions => Date & Time. As the results above demonstrate, only a single HP VEE programmer was able to do this in the required time. Moreover, the date and time function under Constant in the Data menu does not seem to produce the desired result.

Task 3b. Write data to file: print data to file

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	11	1	3	15
TIME-OUT	4	10	12	26
	15	11	15	41

Task 3c. Write data to file: print PASS/FAIL to file

	HP VEE	MS C++	LabVIEW	TOTAL
PASS	10	1	3	14
TIME-OUT	5	10	12	27
	15	11	15	41

The task time data (which is a combined time to complete all three subtasks listed above) for all three groups were at or exceeded the 8-minute limit, and as expected, we did not find any differences among the three groups:

Software	N	Mean task performance time (seconds)	Std. deviation (seconds)	ANOVA
HP VEE	15	444.93	75.21	$F(2,40) = 1.21$, $p < .2$ (not significant)
MS C++	11	471.27	79.22	
LabVIEW	15	490.80	79.65	

Task 3b. and 3c. summaries

The disadvantage of LabVIEW and C++ programmers in completing this task stems from a their difficulty in writing data to a file. These participants were generally unsure about what function call to use to create a file and write data to it.

Summary

Two main findings emerged from the test. The first is that overall, more participants completed tasks using HP VEE than those using MS C++ or LabVIEW. Overall, an average of 75% of HP VEE participants completed each task, compared with 62% of LabVIEW participants, and only 49% of those using MS C++.

More importantly, the average time to complete tasks was substantially faster for those programming with HP VEE than with either of the other software packages. The average time to complete a task using HP VEE was 222.29 seconds, compared with 307.63 seconds using LabVIEW, and 342.74 seconds using MS C++. Across all tasks, programming with HP VEE was about 27% faster than programming the same tasks with LabVIEW, and 35% faster than programming them with MS C++.

Analysis of individual task times confirmed HP VEE's speed advantage. Analyses of variance showed that programmers using HP VEE were significantly faster than those using LabVIEW and/or MS C++ when programming many typical test and measurement tasks.

These results suggest that programming with HP VEE can provide productivity advantages, as measured by speed, over programming either in LabVIEW or in MS C++.

Participant comments

In the remaining section, we list the comments we obtained from participants regarding their likes and dislikes about the software package they evaluated.

What do LabVIEW programmers like about LabVIEW?

General
Generates small tasks very easily
Easy to get around in once you're used to it
Good engineering software so hardware people don't need to learn a lot of software
High productivity for coding once have experienced programmers
Can put stuff together very quickly
Nothing out there that's more intuitive

Built-in VI's
Like function libraries - large amount of functions developed for you for data analysis
Like libraries of functions - many capabilities, wide variety of functions
Don't have to be hard-core programmer - automates parts of program

Graphical environment
Don't have to define data types
Like graphical sense - shows flow like flow diagram or schematic
Like visual manipulation
Easier than text-based to see flow
Don't have to relearn new syntax for new language

Help
Like all different types of on-line help
Context-sensitive help screen very helpful
Like nationwide technical support

Debugging
Easy to debug
Like highlight execution - see what's happening step by step
Helps you to see right away what the problems are

Capabilities
Like instrument driver support for most instruments
Versatile - has features that full programming language would have
Powerful - never had problem with it
Not many things you can't do

Miscellaneous

Structures & loops are easy to understand
Can understand other's code
Can change tool by hitting Tab
Easier than I expected to talk to instruments
Like color coding
Can select more than one object
Relatively inexpensive given features & power

What do LabVIEW programmers dislike about LabVIEW?

General
Could use more sophistication, i.e., better user interface
Can get complicated quickly
Kind of frustrating - need to get oriented

VI function list
Don't like finding things - lacks search tool to find VI's
Selecting icons in palettes is difficult so select wrong one
Difficult to look through each VI in function call window
Function list should be organized better
The slowest part is finding the correct VI
Awful lot of icons in the function list to sort through - a lot look like duplicates
Not immediately apparent what the icons in the function list do
Its frustrating trying to find the right function
Can't read function call icon names
Wants to know how to find VI textual list
Difficult to move through VI's and watch the title bar to catch their label
Want more descriptive names for function call icons
There's 100 config icons so I'm going to assume they're all the same thing

Wiring
Wiring tools could be better
Diagrams are hard to read because wires are all confused
Should automatically line up lines to VI inputs, i.e., "snap to"
Lines should move with VI's
LabVIEW's shortcoming - wires can be all over the place

Help
Don't like that help is in black & white
Help doesn't autoscale
Help doesn't disappear when don't need it
Help is vague - should have wizard point to manual
Documentation more tricky than for text-based code
Wants help to say where to find functions

Capabilities
No undo so have to revert but can't if its not saved
Don't like that not able to save as bitmap
Want snap to or cross hairs over whole screen to size things

Graphical environment
Prefer to execute code fragments & execute line by line
Data flow makes doing some things more difficult than in C
Couldn't create executable code that is standalone
Windows, palettes are overwhelming
Too many menus on top of each other
Palettes can take up a lot of room

Textual information
Don't know what the "error code" is
Word "ASCII" in function label Capture & Return ASCII Samples threw me off - should be "data" or "waveform" samples
Is there a book to see what that error message is?
Phrase "resource name" in Initialize Waveform Analyzer misleading since was called "instr desc" for function generator
Not too confident that the voltage setup function call will default to peak to peak

Miscellaneous
Not sure where to find tools for dealing with I/O
Some of the drivers weren't in order
Array functions not straightforward
Doesn't document well in terms of generating reports
(Version 3.1) memory issues
Problem changing tools, particularly text to other tools
Don't know how to build array
Has backward compatibility issues
Wants to be able to move field separately from its label
Can't use on all the machines we have
Compiling slower than text-based

What do HP VEE programmers like about VEE?

General
Intuitive
If you want to make a small program - its fairly easy to do
Easy to translate to new tasks
Don't have to be a programmer
It's economical & efficient

Compared to what's out for Labtech, LabVIEW, & Testpoint, it's by far the easiest to use
More intuitive than text-based applications
Lots of intuitive things - easy to accomplish tasks
Easy to learn
Immediately useful

Communicating with instruments
Likes ease of connection between objects, and talking to objects
Instrument calls worked w/ defaults the first time
Easy to use Plug & Play drivers
First time using Plug & Play, took him a little time, but very easy to use to connect to instruments
Detects instruments so don't have to figure out if they're there or write drivers
Like direct I/O

Programming speed
Speed of programming/program development is pretty strong - one of the quicker ones
Fast environment to code up stuff
Fairly obvious - makes it fast to try things
Functions in easily accessible boxes
Easy to write - dropping boxes, quick interconnecting, visual
Relatively fast in terms of acquiring data
Quicker than text-based applications
Saves a ton of time

Help
On-line help is good
User friendly once you learn it because of the user manual & any user panels that have been set up
Like bubble help
Like help at bottom of Select Function Panel window
There are consultants, a well-written manual, excellent user groups

Graphical environment
Like that it's object-oriented - therefore more intuitive
When your done, it's all flowcharted - can follow the flow of your program at a glance
Like the visual debugging flow
Don't have to worry about syntax too much as long as have user manuals
Like the graphical environment
Very sequential, like flow chart - easy to develop quick program for test & development
Programming development environment makes it easy for simple tasks
Programming development environment leads to short learning curve

Capabilities
Like the display capabilities, e.g., display waveform

Like the data collection features
Many capabilities - data processing, input, data acquisition, output (even in special formats)
Can control 3rd party boards
Can link to other programs written in different languages
Data execution, probing outputs make it easier to debug
Debugging is good - animating the process is nice

Miscellaneous
Things are in the right place
Usability is pretty good - files on side, tree format
Likes different colored lines
Can control when things happen
Reuse code all the time
Error messages more intuitive than most
Like free run-time
Like that files are set up like in Windows
Like how objects show what inputs you need
Likes can click on left top of box to delete - like Windows

What do HP VEE programmers dislike about VEE?

Screen real estate
Takes a lot of workspace
Run out of room quickly - have to scroll
Can't scale the environment, e.g., zoom in & out
Hard to manipulate if don't have real big monitor
Impossible to connect boxes that are too far apart - should be zoom
Can be cumbersome - so many boxes
Don't like annoyingly big objects
No zoom out - have to have 21-inch monitors

Capabilities
If he writes a program in VEE, has to have a copy of VEE on every machine - not the case with Visual Basic
Still no undo button, darn it - that would be really great
Functionality is confined somewhat
Surprised by lack of Clear Device function call
Wants to select both Date & Time at once from Built-In Functions list
If # of points is too big - freezes if don't put in time-out
Can't take screen from network analyzer & print on VEE
Would like to see more quick keys
Would be nice to have way of demonstrating that was able to reset instrument, i.e., visual trigger
There's just so much functionality you hardly remember what it all does

Feature richness is too rich at times & sometimes not rich enough - might prefer fewer features & more depth
Can't control XY trace scaling
Can't do standard 1 dB compression test
Should automatically generate input pin when add transaction to the object for writing data to a file

Textual information
Want more information when errors occur - messages. are too cryptic
Headings & selections don't correspond, i.e., not descriptive
Hard to find out what each function call does
Some functions don't seem to do what they imply
Expects Arbitrary Waveform Setup function call to contain selection for frequency, waveform type, etc.
Confused by phrase "Write text @ eol" in object for writing data to a file
Nothing tells me if the voltage is peak to peak in the Output Voltage Setup panel
Waveform (Time) display never works with my programs

Help
Not consistent in how to get to help, i.e., right hand button or F1
Help is a "3" on a scale from 1 to 5 in terms of helpfulness
Help not helpful - help gives me information on panel display not VXI display, want help for function generator

Miscellaneous
(Version 3.1) As you'd open nested objects then collapsed lower level, would collapse all lower windows
Some functions are buried
Too many options in menus
Lines overlap so can't tell where they're coming from
I/O learning curve
Programming instruments is difficult if don't have built-in functions
Panel drivers too slow
Propagation of some objects works differently than others
Doesn't like when exponent disappears when type # into field
Execution seemed to be slow
Some data manipulation is a bit weak - has run into problems where pulling one data point at a time takes a long time
Not intuitive where to pick up Start box
Have to select tiny edge of Start box to move
Not intuitive what to do with To/From Waveform Analyzer object input
Can do a lot of nice things but looks like spaghetti mess after awhile
Need better way to copy transactions

What do C/C++ programmers like about C/C++?

Capabilities
Like available data types - types of variables
Allows character arrays
Development environment is well-integrated - lots of functionality
Can do whatever program I want - doesn't limit me like other software
Lots of low-level control - never been in a situation where you couldn't do something
A lot of available tools without cluttering
Flexibility of interfacing with other Windows programs

Help
Help is on-line
Convenient to get into C++ help
Dominates market so there are help books with examples
Help at my fingertips
That's what people use so easy to find books with examples

Miscellaneous
Like compiling in project area
Workspace environment nice - flips between different windows, shows standard I/O window, compile window, etc. so there's little window management
Easy to input & output data to waveform analyzer & function generator
Easy to pick up
Can make applications
Can add on to other people's software
Everybody can use it

What do C/C++ programmers dislike about C/C++?

General
Always a run-it-and-see-what-happens sort of thing
Difficult language - high learning curve
Not very intuitive - have to study and learn it

Help
Want help on a particular word
Want coding examples
Difficult to find in help the appropriate variables to use
Need to dig into help
How do I go back to workspace from C++ Help?
Wants help within C for the function generator
Why do they have a separate "How Do I?" section in Help?
There's nobody to support you doing C except the manual

Editor

Editor could be improved for keyboard entrees with brief commands
Wants to jump to close bracket
Would prefer editor to automatically insert parentheses at For loops
Editor should automatically insert brackets
Editor should change color for functions so you know if spelled something wrong
Editor should automatically type in format after user types function call

Miscellaneous

No array dimension trapping (at end of array)
Doesn't lend itself toward Windows programs
Don't like taking two steps to build & execute
So many features - hard to master them all & keep track
Versions are very different
Easy to make mistakes about data type
What a pain in the butt capturing the waveform is
Took way too long to figure out how to prompt a user
CTRL-C should paste item if mistakenly pressed instead of CTRL-V
Wants other Build menu items to gray out when building
Doesn't ask if want to save after selecting Close Workspace
No line number in error message
Haven't used most intuitive names for functions
Stupid to refer to amplitude as voltage