Evaluation of a Multilanguage Videocourse in Object-Oriented Systems

Edward F. Gehringer
Department of Computer Science and
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC  27695-7911
efg@ncsu.edu

Abstract

Since 1987 the author has taught a graduate-level course in object-oriented systems on campus and over several video networks. Since 1989, the course has taught object-oriented concepts through two languages, Smalltalk and Eiffel, along with a discussion of object-oriented design. The last one-third of the class used to focus on active research areas such as type systems, type inference, compiler optimizations, memory management, and object-oriented databases. In Fall 1992, these topics were moved to a new course and replaced with an expanded coverage of C++. Between Fall 1989 and Fall 1992, 423 students took the course. The students were surveyed, primarily through e-mail. Responses were received from 140 students so far. When the responses raised interesting issues, followup questions were posed.

The results suggest that the video medium is valuable for performing on-screen demonstrations; however, the demos will benefit students little unless they explore the system on their own. How the students perceive the three languages is heavily influenced by their previous programming background and the programming environment they work in. There is evidence that Smalltalk may be easier to learn than Eiffel. Our students found C++ easiest to learn, but this is probably because most were already familiar with C and the OOP style. They found it more valuable to learn Smalltalk than Eiffel, and those students moving from universities to industry were especially anxious to learn C++. Students who took the course remotely thought their learning experience was just about as good as it would have been on campus; the disadvantage of being isolated from fellow students was counterbalanced by the advantage of being able to fit the course into their work schedule.

Keywords: Object-oriented programming, Smalltalk, C++, Eiffel, videocourse, browser, programming environments.

The Course

The subject matter

The course focuses on two views of object-oriented programming, the late-binding view, as exemplified by Smalltalk, and the strongly typed view, illustrated by C++ and Eiffel. These languages are placed in a framework developed by Wegner [We 87], with important concepts from other languages being presented as the need arises. Before Fall 1992, the latter one-third was devoted to research issues. Topics included compilation of late-binding languages such as Smalltalk and SELF (e.g., type inference, cloning of methods); parallel o-o programming environments (a taped seminar given by Hank Levy); and memory-management issues, such as generation-scavenging garbage collection, and implementation of a very large object space. Object-oriented databases are also introduced, currently during the last two weeks of the semester.

It is a great advantage to be able to use the video medium. Not only can one use prerecorded lectures given at other sites, but on-screen demonstrations can be shown. Two professional Smalltalk developers, Sam Adams and Kyle Brown, have given a five-lecture introduction to the Smalltalk programming environment in which a Macintosh screen is displayed via a video interface. In addition, guest lectures can be taped, and excerpted in later semesters. In 1989, Bertrand Meyer gave a lecture for us on the philosophy behind the Eiffel language. Many of the topics he discussed are now covered elsewhere in the course, but his observations remain apropos, and it is helpful to be able to cut to his presentation at appropriate points during the discussion.

The students

The course has been taught on campus each fall in a television studio belonging to the College of Engineering. However, only 110 of the 423 stu-
students have taken the course on campus. Some of these are full-time graduate students, and others are part-time students from local industry. A few students view the course live from other universities around North Carolina. Many more students see the course via videotape through the in-state Video-Based Engineering Education (VBEE) program, and the National Technological University (NTU), which provides degree programs to industry around the U.S. via satellite. Each summer, the previous fall’s lectures are replayed, giving more students a chance to take the course. About two-thirds of the students (285 during the survey period) watch the lectures on tape.

In the fall of 1990, this course also served as the pilot project in a course-exchange program established by NTU. This program allows one university to offer a course taught at another university by viewing the videotapes from the originating school, while grading all homework and exams locally. The course was picked up by Colorado State University and supervised by Prof. Gerry Johnson. The students were primarily part-time CSU students with full-time jobs in industry. Johnson presented a report on the experience at the AMCEE conference in New Orleans in 1991.

The Survey

The author attempted to contact all students who took the class between 1989 and 1992. Surveys were sent via e-mail if possible, and otherwise via fax or U.S. mail. Replies were received from 140 students, approximately 1/3 of all students. Twenty-six questions were asked, in four categories: information about the student, the facilities (programming environments and other software), delivery of the course (quality of various kinds of lectures, for example), and usefulness of the course (to see how students were able to apply what they had learned). The responses make it possible to answer questions like, Do people in industry view C++ more favorably than students do? How helpful is it to teach the Smalltalk programming environment via an on-screen display? and How much trouble do the students have learning each of the three languages?

Use of On-Screen Demonstrations

The video medium offers a potential benefit for teaching Smalltalk: We can give on-screen demonstrations of the Smalltalk programming environment. Students should learn more easily by seeing the environment used than they would by simply hearing a lecture about it. The on-campus students view the demonstration on large-screen monitors in the front of the classroom.

By and large, the students rated the on-screen demonstrations useful, giving them a mean of 3.67 on a scale of 1 to 5 (5=very useful). However, this score was lower than expected, so a followup question was sent to students who had rated this technique 1 or 2 on the scale. The most frequent complaint was that the demos went too fast and were not well organized. The students could not follow the actions completely, because they could not see which keys or mouse buttons were being depressed. Furthermore, in the early years when Sam Adams did the demos, he would take impromptu suggestions from the audience about what functionality should be added. This meant that it was impossible to complete the design in advance. Bugs frequently arose, and were usually corrected so fast that the students could not follow what was being done.

Indeed, any debugging during an on-screen demonstration is likely to be distracting, because it juxtaposes the use of tools such as inspectors with other tools that are being used to write the code. As one student put it,

“Until students understand the tools available, showing how the tools work together is of no use. Consider the following analogy. I’m an auto mechanic and I’m going to show you how to tune up your car. I slowly walk you through a step-by-step process using tools you’ve never seen or used before and discuss causal mechanisms you’ve never reasoned with before. No matter how slow I go, it will be very difficult for you to reproduce a tune-up unless you have a photographic memory—and in either case, you won’t learn anything to generalize with.

“However, if I let you play with the various tools on a junker and try to figure things out for yourself, and then go through the same step-by-step procedure, you’re more likely to learn something useful.”

This hits on what seems to be the key factor determining whether a student benefited from the on-screen demos: whether or not the student practiced or experimented on a Smalltalk system between viewing the lectures. The vast majority of students who thought the on-screen demos were very useful had tried out the concepts between lectures. Only one of the students who found the demos unprofitable reported doing so.

1Jeff Pittges, now at Georgia Tech.
The students who liked the on-screen demonstrations found that they provided insight into parts of the environment they hadn’t explored on their own. They frequently developed their own exercises to reinforce the concepts of the lecture. They suggested that, in the future, the course staff might develop exercises for the other students. In this respect, the off-campus students had an advantage, as they were often able to take the tapes home and play them while working on their computers. This proved invaluable in picking up details that are lost when viewing the tape all at once.

**Evolution of the on-screen demonstrations**

Preprinted lecture notes were provided for most of the lectures in the course but initially, not for the on-screen demonstrations, which were intended to have an element of spontaneity. It soon became apparent that students were having trouble keeping up. In summer 1991, lecture notes were assembled by viewing the videotapes from the previous fall and writing a narration. The videotapes and notes were then used in later semesters. To our surprise, this had a negative impact on how the demonstrations were received (Figure 1). Among all students, their usefulness was perceived to fall from 3.88 to 3.31 (5=very useful), a difference that is significant at the 95% confidence level. For on-campus students, the decline (3.88 to 3.39) was not surprising, since they were no longer able to ask questions to a live instructor. The off-campus students’ ratings of the demos declined even more, from 3.88 to 3.15, suggesting that the deficiencies of the demonstrations could not be cured simply by providing notes.

We used a different method in fall 1992. A set of lectures was designed and pretaped. The notes were developed concurrently with the lectures. When something unexpected happened, the taping session could be restarted. Ultimately, we had to limit this approach due to lack of studio time, but it did help us smooth over some of the rough spots. It was evidently successful, in that the overall rating in fall 1992 rose to 3.83, an increase that is significant at the 90% confidence level.

**Conclusions:** On-screen demos can be a useful method of introducing the Smalltalk programming environment. To be effective, they should be carefully planned, and rehearsed or pretaped. Students should be given weekly exercises to reinforce the concepts and should be strongly encouraged to attempt them before viewing the next lecture.

**Comparison of the Three Languages**

**Ease of installation**

Ease of installation is not an issue in many courses, where students work on university-supplied equipment. Remote students, however, often need to install software on their own machines. The students judged Smalltalk much easier to install than Eiffel (they were not questioned about installing C++). Only 11% had trouble installing Smalltalk, while 40% had trouble with Eiffel (Figure 2). Eiffel has been a persistent source of installation headaches (in the last four fall semesters, 40%, 25%, 47%, and 38%, respectively, have had trouble). Smalltalk, however, may have improved; no students have reported problems with a current version of either Objectworks or V.

ParcPlace and Digitalk products seem to be equally easy to install, though there were too few students reporting to draw any firm conclusions. Only one of ten students who installed ParcPlace...
software mentioned any difficulty; he said that he did not have enough free disk space for v2.5 (10–12 MB required), and that he never got all the fonts to work, despite following the instructions carefully. Eight out of 74 students had trouble installing Smalltalk V. Two of these used the long-outdated Smalltalk V 1.0 back in 1989. The most common complaint was that V/286 1.1 required 1.5 MB, contrary to the advertised 1.0 MB.

The statistics slightly overstate installation difficulties with the Eiffel compiler, as three students reported problems accessing NCSU’s Eiffel over the Internet, which clearly had nothing to do with installation. Of the remaining 23 students reporting trouble, most of them were using the ill-fated Eiffel/S for PCs and compatibles, which has since been withdrawn from the US market. The most frequently mentioned problem with Interactive Software Engineering’s Eiffel was lack of write permission to certain directories. Eiffel 2.3 is set up to compile most of the system classes lazily, so compiling a user program may trigger a compilation of part of the system, which will cause a write error unless the user has write access to some Eiffel directories.

**Ease of learning Smalltalk vs. Eiffel**

The ease of learning a programming language depends on many factors, including the programming environment and speed and accessibility of the computer. In general, our students found Smalltalk slightly easier to learn than Eiffel (3.13 to 2.87, on a scale of 1 to 5, significant at the 95% confidence level). However, several of them said that getting a superficial knowledge of Smalltalk was easy, while learning it well was very difficult. They really seemed to be referring to the effort required to become proficient with the class library. A few students said that their knowledge of other non-object-oriented languages hindered their learning of Smalltalk. This is consistent with results reported by Detienne [90]. On the other hand, three students counted knowledge of Lisp as an advantage in learning Smalltalk, due to the similarity of their concepts of dynamic binding and the use of implicit pointers.

Those students who found Eiffel easy to learn tended to cite its similarity in syntax and run-time model to familiar languages like Pascal and C. Its clean and consistent design was also considered a plus. So was the fact that it is less dependent on its integrated environment than Smalltalk. One adds and compiles code using tools that are more familiar.

To get a better feel for why one might consider one language easier to learn than the other, I sent a followup question to students who rated one language much easier (two or more points) than the other. Most responses cited peripheral factors, such as the textbook or programming environment. The students who found Smalltalk much easier mentioned its extensive support for graphics and run-time debugging. Smalltalk made rapid prototyping easy; Eiffel’s strong typing made it difficult. Students found it hard to create a deferred parent class which two other classes could inherit. Also, recompiling a class is much faster in Smalltalk than in Eiffel, which generates C code that must then be compiled and linked.

Students found the Smalltalk browser handy for learning about the system classes; it encouraged them to “play around” and grow comfortable with the language. This gave them enough patience to overcome the bias they might otherwise have felt toward the more familiar command-line interface of Eiffel. It should be noted, though, that in several semesters, we lacked the graphical environment to run the most recent version of the Eiffel browser.

Students who found Eiffel much easier most frequently mentioned its extensive compile-time diagnostics. Their usefulness can be attributed, at least in part, to Eiffel’s well developed type system which makes early diagnosis of errors possible. Once a program compiles, it frequently runs to conclusion the first time. This observation should be tempered with a statement that this applies only to diagnostics in early passes. Eiffel errors encountered on later compilation passes (e.g., Pass 4), or by the back-end C compiler, are very difficult for the novice to track down.

However, not all of the reasons Eiffel was rated easier can be attributed to the language. Having learned Smalltalk, students had already wet their feet on o-o concepts before delving into Eiffel. Also, our Eiffel programming assignments frequently made less use of the class library than did our Smalltalk assignments. A followup question asked whether students would have found Eiffel easier even if browsers were equally good and they had to look up the same number of classes in each. Most students had trouble answering this question, but one said that Smalltalk would have been easier because of its more rapid feedback when an error was made.

**Conclusion: **Our experience suggests that Smalltalk is a somewhat easier language to learn than Eiffel. An important reason is Smalltalk’s better developed programming environment, rather than any inherent feature of the language.
Ease of learning Smalltalk vs. C++

In fall 1992, we taught C++ along with Smalltalk and Eiffel. Students found C++ significantly easier than Smalltalk (3.77 to 3.13, significant at the 95% confidence level). This seemed to be heavily influenced by previous knowledge of C. Eleven students responded to a followup question asking how much C experience they had had before studying C++. Nine of these eleven had at least two years of C experience. These nine gave C++ a mean score of 4.11 for ease of learning. Both of the less experienced students rated it a 3, which was lower than they rated Smalltalk. Of course, these results are hardly definitive. In a larger sample, it would be interesting to compare novices’ difficulty in learning each of the two languages, controlling for the fact that one of the two languages is taught first.

Some students found C++ easier because it was not the first object-oriented language they tried to learn (having just studied Smalltalk). They also found the extensive Smalltalk class library hard to get used to. Although its richness would be an advantage in the long run, beginners needed a lot of time to discover how to perform simple tasks.

Students who found Smalltalk easier alluded to some of the well known rough edges of C++, including its much larger syntax, less readable code, and the lack of a standard class library. The complexity of C++’s inheritance, with friends, templates, multiple inheritance from virtual or duplicated base classes, and various protection modes makes it difficult to learn. One also needs to write header files, and match function declarations with their definitions.

An artifact of teaching C++ seemed to be that it made it easier to learn Eiffel later. When only Smalltalk and Eiffel were taught, Smalltalk was rated easier by a margin of 3.15 to 2.78, significant at the 95% level. When Smalltalk was taught, then C++, then Eiffel, the languages were rated 3.00, 3.77, and 3.27, respectively, with the difference between C++ and the other two languages being significant at the 95% level. Thus Eiffel, instead of being more difficult, became “easier” to learn than Smalltalk.

**Conclusion:** Our students found it easier to learn C++, probably because most of them already knew C. But many students preferred Smalltalk because of its perceived elegance of syntax and environment.

Student Perceptions of Languages’ Importance

Ease of learning is one prominent attribute of a programming language. Another is the importance of the language to industry. One might thus expect students to disdain Eiffel and esteem knowledge of C++. The picture that actually emerges is considerably more complex.

Academic stereotypes

Students from universities might be expected to appreciate the “purist” perspective of Eiffel more than students from industry. When asked whether it was valuable to learn Eiffel, the university students ranked it 3.45 on the 1-to-5 scale compared to 3.06 for the industrial students. However, this difference is not significant at the 90% confidence level. Conversely, one might expect students from industry to be more enthusiastic about C++. But when asked whether it would have been more valuable to cover C++ in more depth, the university students gave an average score of 3.87, compared to their industry counterparts’ 3.59, a difference which is not significant at the 90% confidence level. Interestingly, the group rating C++ most important was those who had moved into industry since taking the course; they rated it 4.25, with 7 of 10 such students agreeing strongly. This suggests that they may have found C++ valuable in finding a job, although not enough such students responded for the results to be significant.

![Figure 3: Students’ ratings of the three languages](image)

The growing popularity of C++

Most of the important concepts of object-oriented programming can be adequately conveyed through Smalltalk or Eiffel. At the same time, popular demand for C++ has been increasing. Among our own students, this can be seen from
the rising scores (Figure 4) given to the question, “This course would have been more useful if C++ had been covered in more depth.” Before fall 1992, only one week was devoted to C++, and no programs were assigned. In fall 1992, coverage was expanded to five weeks, with programming required. At this point, the mean score accorded the C++ question fell to 3.67. However, the difference between the fall 1992 scores and the scores from the previous year is not significant at the 90% level.

Ironically, the decline in the C++ score was matched by a drop in the rating given the statement, “It was valuable to learn Smalltalk.” This score, which had averaged more than 4 for five consecutive semesters, suddenly fell to 3.67. The difference between the fall 1992 scores and the earlier scores taken as a whole is not significant at the 90% confidence level, however. In contrast, the Eiffel scores, which are lower than the scores for Smalltalk or C++, have remained relatively constant over time.

![Figure 4: Perceived value of studying the various languages](image)

### Conclusion:
Students at the university did not appreciate Eiffel significantly more than their counterparts in industry. Students at the university seemed to be interested in learning C++, especially those students near graduation. Increasing demand for C++ finally led to its inclusion in the course in fall 1992.

### Live Classroom vs. Remote Video

The on-campus students found it easier to learn all three languages than their off-campus counterparts, but only by a small margin (3.23/4.00/2.93 for Smalltalk/C++/Eiffel, vs. 3.69/3.76/2.84). This probably reflects the off-campus students’ greater isolation. In the future, this can be combated by setting up a listserv for the students to broadcast e-mail messages to each other. By contrast, the on-campus students were somewhat more likely to say the pace of the course was too fast. In most semesters, more off-campus students received A’s.

On the surface, it appears that off-campus students were non-committal (3.06 on a scale of 1–5) when asked whether their educational experience would have been better on campus. Upon further investigation, this reflects several tradeoffs.

### Time considerations

For local students, the most obvious advantage of taking courses by video is that it saves commuting time, including the time needed to look for parking. Also, there is an infrastructure to provide students with textbooks, software, and course notes, eliminating trips to the bookstore. Finally, off-campus students frequently have computers right at their desks, while on-campus students must usually work in labs.

These time savings, however, are partially offset by the need to install one’s own software, often in a different version from that used by the rest of the class. If this is not or cannot be done, the only alternative is to use software over the Internet, which can be slower, especially if the remote host is heavily loaded. Moreover, if the version is different from the one used by the instructor, the in-class examples may not work. Many of our students, for example, use Smalltalk V/286 rather than Objectworks or V/Windows as used in class. Finally, there is the psychological effect of watching tapes during off-hours, a few days later than the on-campus students; this made at least one student feel as though he was always behind.

Time-shifting is also an advantage of video-courses. Students can watch tapes when it fits their work schedule. They can work around business trips, or, sometimes even take the tapes with them.

### Interpersonal contact

Direct contact with the instructor and the teaching assistants is beneficial to learning; without it, students spend more of their time discovering the obvious. There may be no local experts on a particular software package. E-mail and sometimes fax can alleviate this problem somewhat. Students watching via videotape, though, have no opportunity to ask questions in class. Finally, since they often have no fellow students at their site, they also miss interacting...
with other students. Not only does this make it harder to get help with their problems, but they also don’t have the opportunity to learn by helping someone else.

To study the importance of having other students to interact with, the off-campus students were divided into two groups: those alone at their site \( n = 34 \), and those with at least one other student at their site \( n = 60 \). As expected, the lone students more strongly believed that their experience would have been better on campus (mean score 3.29) than did those who had colleagues at their site (2.95). The magnitude of this difference is not large; it is not significant at the 90% level, and it is less than the difference in the two groups’ responses to several other questions (e.g., usefulness of live guest lectures). Therefore, for remote students, the advantage of having colleagues at one’s site appears to be small.

Again, there are offsetting advantages to using video. Instead of asking questions in class, a student can rewind the tape and repeat a difficult section. Note-taking is also facilitated by the ability to freeze-frame. And there is not always a dearth of potential helpers; on some topics, the off-campus student may be able to consult with many experienced co-workers.

**Conclusion:** For off-campus students, the advantages of taking a videocourse balanced the disadvantages. Greater access to the instructor and other students was offset by the need to leave work and use a different computing environment than they were familiar with.

**Summary**

Students appreciate the use of current technology to deliver the course, but are less enamored of special techniques than one might expect. Full-time students find it easier to pick up a new programming environment, due to the greater support staff and large number of peers, but are less adept at using it than full-time programmers are. Smalltalk is very well received, as is C++, though for different reasons. Students do understand the reasons for preferring a purer language like Eiffel over C++, but do not believe it will be so useful in finding employment.

**Recommendations**

The choice of what programming languages to cover in a curriculum is always a difficult one, fraught with pedagogical and political considerations. However, a careful study of the findings of this paper suggests several guidelines for how to improve delivery of a course, regardless of what language is used.

- A video display of a programming environment is useful, but the presentation must be carefully rehearsed to be sure that the students do not get lost. Things that are not obvious (e.g., which mouse button is being depressed) should be explained.
- An on-screen demonstration should concentrate on one thing at a time, either the program or the programming environment. Interspersing coding with use of debugging tools is likely to prove distracting.
- Students need practice exercises to reinforce the lessons of each demonstration.
- The availability of a browser was perhaps the most important factor affecting how hard it was to learn a particular language; therefore, browsers should be obtained and installed whenever possible.
- Devices such as listservs or bulletin boards should be used to facilitate communication among students; this will improve the learning process, especially for students isolated from their classmates.

**Acknowledgments**

The author would like to thank Blake Stewart of the National Technological University for setting up an alias and forwarding mechanism so students could respond to the survey anonymously. He is also grateful to Faye Childers and Dr. Dennis Boos of the NCSU Statistics Department for help with statistical analysis using SAS. Finally, this paper would not have been possible without the help of the 140 students who responded to the survey. Many also answered several follow-up questions.

**References**


### Appendix: Tabulated Results of the Survey

*Note: Part I (Questions 1 to 5) involved identifying information (e.g., did you take the course for credit, were you in industry, etc.).*

<table>
<thead>
<tr>
<th>Part II. Facilities</th>
<th>All students</th>
<th>Took class on campus</th>
<th>Took class via tape</th>
<th>Took class as full-time student</th>
<th>Took class in industry</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Rate on a scale of 1 (= very hard) to 5 (= very easy): How easy was it for you to get access to the software you needed for the course?</td>
<td>3.38</td>
<td>3.56</td>
<td>3.31</td>
<td>3.38</td>
<td>3.42</td>
</tr>
<tr>
<td>7. Did you find the course unusually expensive, either because of the need to purchase two textbooks, or because you needed to buy software?</td>
<td>29%</td>
<td>33%</td>
<td>27%</td>
<td>41%</td>
<td>25%</td>
</tr>
<tr>
<td>8. What version of Smalltalk did you use? [Many different versions; results not tabulated here.]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. Did you have problems installing Smalltalk if applicable)? % answering yes:</td>
<td>11%</td>
<td>14%</td>
<td>9%</td>
<td>32%</td>
<td>7%</td>
</tr>
<tr>
<td>10. Did you have problems installing Eiffel if applicable)?</td>
<td>40%</td>
<td>20%</td>
<td>44%</td>
<td>30%</td>
<td>40%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part III. Delivery (1=strongly disagree to 5=strongly agree.)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11. Rate on a scale of 1 (= very hard) to 5 (= very easy): How easy was it to learn to use Smalltalk?</td>
<td>3.13</td>
<td>3.23</td>
<td>3.11</td>
<td>3.12</td>
<td>3.13</td>
</tr>
<tr>
<td>11a. How easy was it to learn C++? (Asked in Fall 1992 only.)</td>
<td>3.77</td>
<td>4.00</td>
<td>3.80</td>
<td>3.60</td>
<td>3.82</td>
</tr>
<tr>
<td>12. How easy was it to learn to use Eiffel?</td>
<td>2.87</td>
<td>2.93</td>
<td>2.74</td>
<td>3.19</td>
<td>2.77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part IV. Usefulness (Same scale as Part III.)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13. The lectures by the regular instructor were helpful in understanding the material.</td>
<td>4.00</td>
<td>4.07</td>
<td>4.03</td>
<td>4.00</td>
<td>4.01</td>
</tr>
<tr>
<td>14. Showing an on-screen Smalltalk display during lectures was helpful in understanding the material.</td>
<td>3.67</td>
<td>3.52</td>
<td>3.81</td>
<td>3.51</td>
<td>3.75</td>
</tr>
<tr>
<td>15. The live guest lectures were helpful in understanding the material.</td>
<td>3.78</td>
<td>3.74</td>
<td>3.80</td>
<td>3.60</td>
<td>3.83</td>
</tr>
<tr>
<td>16. The videotaped guest lectures were helpful in understanding the material.</td>
<td>3.56</td>
<td>3.63</td>
<td>3.56</td>
<td>3.59</td>
<td>3.54</td>
</tr>
<tr>
<td>17. It was helpful for the instructor to include in his lecture excerpts from the pre-taped Meyer lecture on Eiffel.</td>
<td>3.97</td>
<td>3.75</td>
<td>4.07</td>
<td>3.77</td>
<td>4.04</td>
</tr>
<tr>
<td>18. It was (or would have been [whichever applies]) helpful to be working on a Smalltalk system at the same time you were viewing the lectures.</td>
<td>3.84</td>
<td>4.00</td>
<td>3.76</td>
<td>4.06</td>
<td>3.76</td>
</tr>
<tr>
<td>19. This question for video students only: My educational experience would have been better if I had been able to take this course on campus.</td>
<td>3.07</td>
<td>3.00</td>
<td>3.08</td>
<td>3.00</td>
<td>3.03</td>
</tr>
<tr>
<td>20. Was the pace of the course too fast or too slow? (1 = way too slow, 2 = too slow, 3 = about right, 4 = too fast, 5 = way too fast).</td>
<td>3.23</td>
<td>3.43</td>
<td>3.11</td>
<td>3.49</td>
<td>3.14</td>
</tr>
</tbody>
</table>

**Survey Results:**

- **Part I:**
  - Questions 1 to 5 involved identifying information.
  - Many students indicated they found the course unusually expensive.

- **Part II:**
  - Questions 6 and 7 focused on the ease of accessing software.
  - Students rated the course difficulty and software access.

- **Part III:**
  - Delivery questions rated on a scale from 1 to 5.
  - Lectures, guest lectures, and material excerpts were helpful.

- **Part IV:**
  - Usefulness questions focused on instructor lectures and course pace.

*Results tabulated.*