Prior Experience and New IT Students

Edward Holden and Elissa Weeden
Golisano College of Computing and Information Sciences, Rochester Institute of Technology, Rochester, NY, USA

eph@it.rit.edu **emw@it.rit.edu**

Abstract

The Information Technology (IT) Department of the Golisano College of Computing and Information Sciences (GCCIS) at Rochester Institute of Technology (RIT) offers core courses in four threads. These threads cover various subject areas of Information Technology. Increasingly, students enter the IT program with prior programming experience from high school or college courses, work, or from other activities. Several studies, including the authors' own have indicated that students, who have prior experience, often perform better in the first programming course. Our earlier study verified that using 2002 data and found that after the first course, the experience did not have a significant impact on student performance in subsequent programming courses.

In this study, we reexamined the results of our first study with data from 2003 and looked at beginning courses in other core threads to see if the prior experience had a significant impact on those courses.

Keywords: Computers, Education, Information Technology (IT), Information Science Education, Computer Science Education, Curriculum, Information Systems Education

Introduction

Like most computing departments, the Information Technology (IT) Department of the B. Thomas Golisano College of Computing and Information Sciences (GCCIS) at Rochester Institute of Technology (RIT) offers introductory courses in four subject areas as part of the core requirements for a BS in IT. These are intended to give students a broad understanding of these IT areas.

The core subject areas fall into four threads: programming and application development, web and multimedia development, hardware and networking, and user-centered design and development. The first of these is the most complex thread and the programming courses are prerequisite to courses in other threads.

When students are entering the program, various criteria are used to place students into courses. The advisor asks questions and looks at student records in an attempt to place students in courses where they will be successful. This paper is part of an ongoing study which is trying to determine criteria that will predict success in the IT program and design courses to fit certain student profiles. This study is measuring the correlation between specific prior experience criteria and student performance as measured by the course grade received upon completion of certain courses. The grade is based on the typical four point scale. The assumption is that more experience will allow students to obtain higher grades.

Since much of the information received by the advisors comes from the students themselves, the interest was in the students' perception of their skill levels. For this reason a survey was developed to investigate students' perceptions of their past experience. A 2002 survey was originally used to investigate experience level and comfort level with computers from the students' perspective. This was given to all students entering the first programming course. The survey was narrowed to a target population of new IT majors entering as freshman. The response rate was nearly 100%. In 2003 and 2004, these survey questions were included as part of an enlarged survey that added additional questions concerning student learning styles. This survey is sent to all incoming IT freshman in the summer before their entrance into RIT. In 2003, Eighty-eight percent (88%) responded and allowed us to use their data as part of the study. Processing is not complete on the 2004 data. The learning style results are not part of this particular paper.

In 2003 and 2004, the authors studied the introductory programming sequence using data from 2002 to determine the impact of prior programming experience on that sequence (Holden & Weeden, 2003, 2004). Figure 1 shows an illustration of this sequence. It was found that prior programming experience does have an impact on student performance in the first course in the programming sequence; however it does not have a significant impact on student performance in subsequent courses. By the end of the sequence, students seem to have equivalent performance. Any level of programming experience improves performance in the first course, and students with prior experience are more likely to complete the faster three-course sequence versus the four-course alternate sequence.

The hurdle in learning programming appears to be learning the basic concepts such as sequence, iteration, and decision which are covered in the first course. Once these are learned, students are able to master the more advanced concepts of later courses.

The study further found that students' indication of "comfort level" with computers is not an indication future performance in programming courses. In addition, the programming language used in the prior experience does not seem to indicate future success over other languages. This would indicate that students' perception of their comfort level with computers was not an indicator of future performance, nor was a particular language used in the past and these questions are of little value in determining success or failure. This would also support that learning the basic concepts is the hurdle.

The depth of programming experience does have an impact in the first course although there are diminishing marginal returns for going from minimal experience to higher levels, but these returns are not statistically significant. Once the key hurdles of problem solving and basic programming constructs are achieved, as covered in the first course, other material is more easily learned.

Students exposed to the concepts of inheritance and encapsulation have significantly better performance in the first course. This may be because these students had a more rigorous programming experience. The performance converged as we moved through the sequence. Non-experienced students performed as well when dealing with the advanced concepts as their experienced peers.

The paper further concluded that formal educational experience does improve performance in the first course in the sequence. An added improvement occurs when the formal experience is combined with another type of experience as this may indicate an increased motivation to program.

Building on these results, we decided to examine the 2003 data and see if the results are consistent with the earlier findings for the first course in the programming sequence and for the percentage completing the faster three course sequence. In addition, we decided to look at two of the other three threads in the core to see if prior experience impacts performance in the first course in each thread. The two threads examined were web and multimedia development, and hardware and networking. User-centered design and development was not included since it begins late in the student's course of study, typically at the end of the first year and continuing into the second. In addition, this thread covers some of the important softer skills needed by an IT professional and may be less impacted by prior technical experience.

Method

A study done in 1987 indicated that approximately 34% of the 321 students sampled had prior experience through a computer science course in high school (Franklin, 1987). The authors found that 71% of the 159 incoming students sampled had received prior programming experience through high school coursework or some other experience in 2002. In 2003 this figure was a comparable 62%. As more and more students are arriving at college with programming experience, it is important to have an understanding of the impact this experience has on student success in an information technology program.

Other studies have indicated that students who had prior programming experience performed better in an introductory computer science course than their non-experienced classmates (Byrne & Lyons, 2001; Franklin, 1987; Hagan &Markham, 2000; Taylor & Mounfield, 1989). Through our study the authors hope to determine how much prior experience impacts student success throughout an entire IT programming sequence, rather than just an initial course and what impact this experience has on other initial core courses.

This study was initiated by conducting a survey of over 200 IT students each year in 2002 and 2003. The study was narrowed to 159 new first-year students who entered the Programming for IT sequence for the first time in the fall quarter, 2002 and a similar group of 151 students in 2003. The survey was aimed at gaining insight into prior student experience in programming and individual motivation to program. The survey data was collected for analysis when course grades were available.

Part I of the survey was designed to determine the students' comfort using computers and asked if they had any prior programming experience, either academically, in a work environment or for fun. Students with no prior experience did not complete part II of the survey. Students with experience completed part II, which provided more insight into the nature of that experience. Appendix A shows part II of the survey.

Student performance in the courses in the programming sequence was shown to be statistically significant when compared to the data gathered in the survey assuming a 95% confidence interval (alpha = 0.05).

The authors examined the performance in the first course in each of the three threads based on the experience level of the students. The three course completion rate of the programming sequence for experienced students vs. non-experienced students was also examined.

As stated in the prior paper (Holden & Weeden, 2003), people with prior programming experience filled in part II of the survey that examined the depth of their experience. The following questions on the survey tried to assess how deep the experience was.

- Did you learn about the concepts of encapsulation, inheritance and polymorphism? An assumption was made that if these concepts were taught, the experience was more rigorous than other experiences.
- High School – Did you learn to program as part of a high school class (either formal or informal)? If yes, how many semesters did you take courses involving programming?

Was this programming experience part of another course (e.g. Math, Business, Science…)?

- College – Did you learn to program as part of a college course? If yes, how many semesters (quarters) did you take courses involving programming? Was this programming experience part of another course (e.g. Math, Business, Science…)?
- Work – Did you program as part of your work? If yes, was your work programming experience full time, part time or less that part time? Roughly how long did you have this work programming experience?

Again, as described in the paper, with this data, the authors developed a formula that assessed the experience level of each individual in the sample.

- Students received a maximum of two points for their high school experience. If they had a one semester course, they received one point. For two or more semesters they received two points. If the programming was only part of another course, the score was multiplied by 0.5.
- A similar approach was used for college experience. The students could score a maximum of three points for college coursework. For one semester they received 1.5, for two or more, they received three points. Again, if the programming was only part of another course, the score was multiplied by 0.5.
- Work experience was handled a little differently. First, a factor was assigned based on the type of position: Full time = 1 point, part time = 0.5 point, and less that part time = 0.25. This was multiplied by a factor reflecting the number of years they held a job: Less than one year = 1.5, one year = 3, Greater than one year but less than or equal to two years = 4, Greater than two years = 6.
- Finally, students who had learned about the object-oriented programming concepts of encapsulation, inheritance and polymorphism received an additional point.
- The sum of these scores was used as the experience index for each individual, with a possible total of 12.

The formula was somewhat arbitrary, but was designed to reflect the assumptions that work experience (application) was worth more than educational experience alone and that there were diminishing marginal returns for additional experience over a certain level. The extra point for having studied encapsulation, inheritance and polymorphism reflected the assumption that courses covering that material would be more rigorous than other courses.

In 2002, the experience levels for students in the sample ranged from 0 to 7.5 out of the maximum possible index of 12 points. The range was 0 to 6.0 in 2003.

We defined student experience levels in terms of the calculated experience index as follows:

• No experience:	experience index = 0
• Minimal experience:	0 < experience index <=2
• Medium experience:	2 < experience index <=4
• Very experienced:	experience index > 4

Core Thread Descriptions and Results

Programming and Application Development

The programming and application development thread is illustrated in Figure 1. It begins with a three or four course programming sequence which is designed so that students with varying backgrounds in programming can obtain consistent programming skills by the end of the sequence. Students typically spend their first three quarters completing the sequence that uses Java as a programming language and progresses from basic programming constructs to object-oriented design and implementation, network programming and threads.

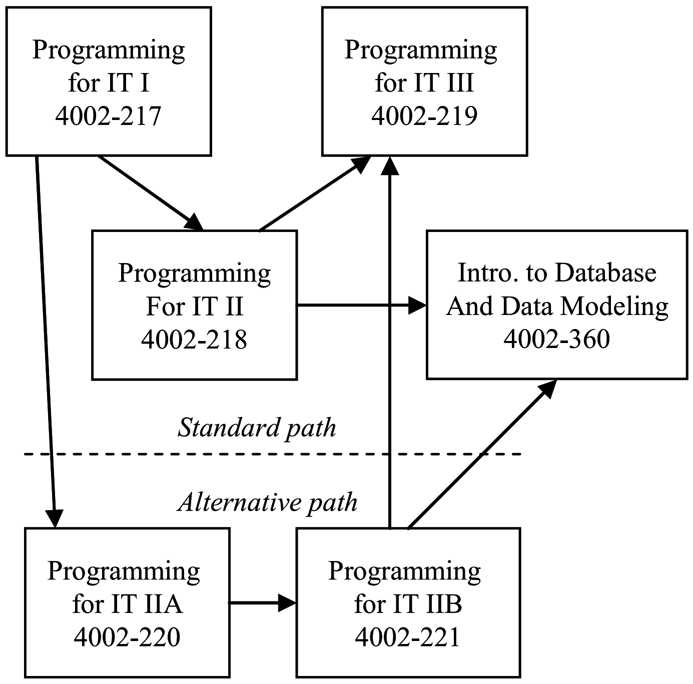

Programming for IT I 4002-217
Programming for IT III 4002-219
Programming For IT II 4002-218
Intro. to Database And Data Modeling 4002-360
Standard path
Alternative path
Programming for IT IIA 4002-220
Programming for IT IIB 4002-221

Figure 1: Programming and Application Development Core Thread

There is an alternative path through this sequence designed for students who desire or need additional time on some of the more difficult concepts at the intermediate level, such as inheritance, GUI and event driven development. Students who follow this alternate path take a four course sequence that begins and ends with the same courses as the three course sequence. Course descriptions are included in Appendix B.

Traditionally we have required all students to take the first programming course (217). This course has been used as a gatekeeper course because it covers the basic programming concepts. If necessary, students are encouraged to get tutoring or other assistance if they have difficulty with this course. After this course students may take alternate paths thru the programming sequence.

Students are not allowed to test-out of the programming sequence, but the first course (217) is used to place students in future courses. Students in 217 must receive a C average on in-class programming exams to pass, regardless of other grades they receive in the course. Students who receive an A or B in 217 could continue with the second course (218). Student who receive a C in 217 are encouraged to take 220 and 221 (alternate path). These two courses cover the same material covered in 218, but at a slower pace. Students who receive below a C in 217 are strongly encouraged to take 217 again. All students have a choice of the path they will take.

This thread culminates with a course in Database and Data Modeling which is normally taken in the sophomore year. This course covers the basics of database technology with an emphasis on Relational databases and SQL. They also learn the fundamentals of Data Modeling with an emphasis on translating data models into relational databases.

Table 1: The impact of prior experience on grades in the first programming course

StudentBack-ground	2002 217 #Stu-dents	% of Stu-dents	Avg. course GPA	2003 217 #Stu-dents	% of Students	Avg. course GPA
w/o experience	46	29%	1.74	57	38%	2.54
With experience	113	71%	2.79	94	62%	3.11
All	159	100%	2.49	151	100%	2.89

Table 1 shows the grades for the first course in the programming sequence based on the students' prior experience. In 2002, Forty-six (46) of the 159 students (29%) who started the first course (217) indicated that they had no prior programming experience. In 2003, fifty-seven (57) of the 151 students (38%) indicated that they had no prior programming experience. The 217 grades indicate the first attempt at 217 in cases where the course was repeated. In addition, students who withdrew from the first course with a grade of W were treated as if they received an F.

In our normal grading, a W grade does not go into the calculation of the student's GPA, although it does appear on the transcript. Few students withdraw from these courses and those who do normally do so because they are performing poorly. In this analysis, we counted them as an F because it was an unsuccessful completion.

Students with prior programming experience completed the first course a full letter grade higher (1.05 points) than students with no experience (2.79 vs. 1.74 on a 4.0 scale) in 2002. In 2003 they competed more that half (0.6) of a letter grade higher 3.11 vs. 2.54). An independent sample *t*-test showed a significant difference (p=0.0005 and p=0.007) between the non-experienced group and the experienced group in the first course for both 2002 and 2003, respectively.

In addition, experienced students complete the programming sequence in the programming and application development core at a much higher rate that the non-experiences students. In 2002, only 24% of the students with no experience completed the standard three course sequence in three quarters, compared to 45% of the experienced students. The remainder took the four course sequence, or did not finish. In 2003, only 18% completed the sequence in three quarters compared to 38% of the experienced students.

Table 2: The impact of different levels of prior experience on grades in the first programming course

Student Background	2002 217 #Stu-dents	Avg. course GPA	2003 217 #Stu-dents	Avg. course GPA
No experience	46	1.74	57	2.54
Minimal Experience	64	2.63	62	3.06
Medium Experience	32	2.94	29	3.14
Very Experienced	17	3.18	3	3.67
All	159	2.49	151	2.89

It is interesting to note that a lower percentage of students with experience withdrew from the first programming course. In 2002, Seven (6%) out of the 113 students with experience withdrew while five (11%) of the 46 students without experience withdrew from the course. In 2003, three students (3%) of the 94 students with experience withdrew compared to four (7%) of the 57 non-experienced students.

This indicates that the prior experience does have a positive impact on the grade in 217.

Table 2 looks at Student performance at the different experience levels. As described before, there was a significant increase in performance among students who had prior programming experience, but there were decreasing marginal returns from increasing levels of prior experience. Students with minimal experience earned almost a full letter grade (0.89) higher than students with no experience in 2003 and half a letter grade (0.52) in 2003, while there was only a 0.31 grade increase from minimal experience to medium experience and only 0.24 from medium to very experienced in 2002 and 0.08 and 0.53 in 2003. In all cases, students with any level of experience exceeded the average for all students (2.49 in 2002 and 2.89 in 2003).

This may be an indication that going from no experience to minimal experience overcomes some key hurdles for new programmers like problem solving and dealing with syntax, basic concepts and logic. In fact, for 2002 data ANOVA post-hoc tests showed that the non-experienced group was significantly different from the groups that had prior experience (minimal (p=0.025), medium (0.007) and very experienced (0.009)). The 2002 data also showed that there was no statistical significance among the experienced groups. An ANOVA for the 2003 data showed that the groups were not statistically different.

One measure that was looked at was the rigor of the prior experience. The authors made an assumption that students who had been exposed to the concepts of inheritance, encapsulation and polymorphism had a more rigorous experience that those who did not.

On the survey, students rated themselves on their knowledge of inheritance and encapsulation as not at all, weak, moderate or strong. The authors considered those with moderate or strong knowledge of these topics as having a more rigorous experience. The average grades were calculated for the experienced students who had been exposed to these advanced concepts vs. those who had not. This is illustrated in Table 3. Students with prior experience in these advanced concepts scored almost three-quarter (0.74) of a letter grade higher on average than those who did not. This is a significant difference.

Table 3: The impact on experienced students of learning advanced concepts on grades in the first programming course

StudentBackground	2003 217 #Students	% of Students	Avg. course GPA
w/o inheritance and encapsulation	79	84%	2.99
With inheritance and encapsulation	15	16%	3.73
All	94	100%	3.11

In addition, all of the three experienced students who withdrew from the first programming course did not have experience with these advanced concepts. However, this is not statistically significant.

This indicates that learning more advanced concepts in the prior experience does increase performance in the first course and indicates a more rigorous prior experience.

Web and Multimedia Development Thread

The core also includes a thread on web and multimedia development composed of two courses. The first, Introduction to Multimedia: Internet and Web (4002-320), provides a basic introduction to Internet technologies and web development. The Internet technology topics (UNIX, FTP, Telnet, email, and protocols) provide a foundation for a variety of IT core courses. The web development and imaging topics provide an introduction to the multimedia and web development topic area within the department, and are a prerequisite for concentration level courses in the computer-mediated experience area of the curriculum. Again, course descriptions are in Appendix B. Like the first programming course, this course has no prerequisites and is usually take by freshmen in the fall quarter. Unlike the programming course there is no significant programming, in the traditional sense. Programming is done in later courses.

In the second course, Interactive Digital Media (4002-330), students will create highly interactive documents making use of time-based and traditional digital media. They will storyboard and implement strategies for navigation in various multimedia environments, edit and synchronize audio, video and animation in multimedia presentations, and write event-driven handlers to support highly interactive interfaces. This course is taken in the fall of the sophomore year and does require programming. It is designed to follow the programming sequence described earlier.

The same evaluations were done on the first course in this thread as was done on the first programming course in the prior section. In this case only the data from 2003 was used, since there was no prior study of this sequence. Table 4 shows the performance of students with and without prior programming experience. The difference is only 0.23. This is not a significant difference (p=.137) as shown through an independent sample *t*-test.

Table 4: The impact of prior experience on grades in the first multimedia course

StudentBack-ground	2003 320 #Stu-dents	% of Students	Avg. course GPA
w/o experience	58	36%	2.90
With experience	103	64%	3.13
All	161	100%	3.04

Table 5: The impact of different levels of prior experience on grades in the first multimedia course

Student Background	2003 320 #Stu-dents	Avg. course GPA
No experience	58	2.90
Minimal Experience	67	3.09
Medium Experience	33	3.21
Very Experienced	3	3.00
All	161	3.04

This is probably because the prior experience surveyed emphasizes programming which is not part of this course and programming skills do not necessarily carry over into the non-programming areas.

Similar results can be seen if the various levels of experience are examined. This is shown in Table 5. Again, the results at each of the levels are very close and vary up and down. These differences are not statistically significant as shown through an ANOVA.

Again, the authors looked at the impact of learning advanced topics on experienced students. This is shown in Table 6. An increase (0.48) can be seen in the average grade for students who viewed themselves as having a moderate or strong knowledge of inheritance, encapsulation and polymorphism according to the survey. An independent sample *t*-test showed a significant difference in achievement (p=.034) between students who were exposed to these concepts and students who were not.

Table 6: The impact on experienced students of learning advanced concepts on grades in the first multimedia course

StudentBack-ground	2003 320 #Stu-dents	% of Students	Avg. course GPA
w/o inheritance and encapsulation	86	83%	3.05
With inheritance and encapsulation	17	17%	3.53
All	103	100%	3.13

Again, exposure to these concepts is assumed to indicate more rigor in the prior experience. This increased rigor may have included a more well-rounded IT background. Further study would be needed to make this determination.

The only two experienced students who withdrew from the class did not have this advanced experience. This is not statistically significant.

Hardware and Networking Thread

A third thread in the core requirements is hardware and networking. This thread includes three courses: Computer Concepts and Software Systems (4002-340), Data Communications and Computer Networks (4002-341) and Internetworking Lab (4002-342). Course descriptions are in Appendix B.

This sequence does not start until the spring of the freshman year, so students will have had a significant amount of college course experience before they take this course. The first course also has the Programming for IT I course (217) as a prerequisite and includes some programming assignments. Based on the results of the authors' prior study, the big impact of prior programming experience was on the first programming course, it was expected that there would be no impact of the prior experience on this course. As you can see in Tables 7 and 8, the average grades for the students are varied. These differences are not statistically significant as indicated through and independent sample *t*-test (p=.134) for Table 7 and an ANOVA for Table 8.

Table 7: The impact of prior experience on grades in the first hardware course

Student Background	2003 340 #Students	% of Students	Avg. course GPA
w/o experience	25	40%	3.52
With experience	38	60%	3.24
All	63	100%	3.35

When the impact of learning advanced concepts on experienced students was explored (Table 9), a reverse effect can be seen. The average grade is almost half a letter grade lower (–0.48). The only experienced person who withdrew from the class did not have this experience. This does not appear to be statistically significant as indicated through an independent sample *t*-test (p=.163).

Since these courses start later in the students' course of studies, it can be assumed that the students who had little experience now have gained experience through their coursework. It was shown in the programming sequence that prior experience proved beneficial in the first course.

Table 8: The impact of different levels of prior experience on grades in the first hardware course

Student Background	2003 340 #Students	Avg. course GPA
No experience	25	3.52
Minimal Experience	24	3.50
Medium Experience	11	2.64
Very Experienced	3	3.33
All	63	3.35

Table 9: The impact on experienced students of learning advanced concepts on grades in the first hardware course

StudentBackground	2003 340 #Students	% of Students	Avg. course GPA
w/o inheritance and encapsulation	33	87%	3.30
With inheritance and encapsulation	5	13%	2.80
All	38	100%	3.24

User-Centered Design and Deployment Thread

The final thread in our core is the User-Centered Design and Deployment thread. This thread covers the softer skills needed by the IT professional. These courses do not begin until the spring of the sophomore year, and continue until through the junior and senior years. Therefore they are not included in this study.

Conclusion

The current research supports the earlier findings (Holden & Weeden, 2003, 2004) that prior programming experience does have an impact on student performance in the first course in the programming sequence; however it does not have a significant impact on student performance in subsequent programming courses. By the end of the sequence, students have equivalent performance. In addition, students with prior experience are more likely to complete the faster three-course sequence than their inexperienced peers.

The hurdle in learning programming appears to be learning the basic concepts such as sequence, iteration, and decision. Once these are learned, students are able to master the more advanced concepts of later courses.

The depth of programming experience does have an impact in the first course although there are diminishing marginal returns for going from minimal experience to higher levels, but these returns are not statistically significant. Once the key hurdles mentioned above are overcome, other material is more easily learned.

Students exposed to the concepts of inheritance, encapsulation and polymorphism have significantly better performance in the first course. This may be because these students had a more rigorous prior programming experience. The performance converged as we moved through the sequence. Non-experienced students performed as well when dealing with the advanced concepts as their experienced peers.

When the first course in web design was examined it was found that prior programming experience does not have an impact. Learning markup languages like HTML is not dependent on knowing traditional programming language concepts.

Likewise, the first hardware course also is not impacted by programming experience prior to entering the program. This may be from two factors. First, students will have had significant experience in the program before they take this first course which may override any lack of prior experience. Second, the skills learned in the hardware course may not be dependent on programming experience prior to entering the program. Since some programming is required in the hardware course, the authors lean toward the first reason.

This study has shown that there is a positive difference in performance for students' who have prior experience, particularly if the experience covered more advanced concepts. This has some broad implications for IT programs. Institutions should consider designing their early curriculum around the experience level of their entering students. They may consider designing more intense courses for experienced students or transitional courses for those who do not have experience. They may also want to organize students into cohorts based on experience. This could eliminate the frustration of early students who complain that a course is to fast or too slow. It may also eliminate the frustration felt by students who believe that they are the only ones who have difficulty with the material. This latter point is currently being investigated.

Future Study

The findings of the earlier study are being used track students in the first course in our initial programming sequence. The thought is that inexperienced students may be inadvertently intimidated by experienced students who appear to know more than the inexperienced students do. The authors think that cohorting these students into separate classes will help eliminate this issue. Since these students are just completing this course, a report on the results of this experiment is not yet available. The results of this study will also include the students' assessment of their comfort level with the course.

Appendix A: The Experience Survey

Students answered a survey prior to entering the program. The survey had five sections: general computing background, computer programming experience, RIT and program affiliation, about yourself, and learning styles. The section below shows the first and second sections of the survey. The second section was answered only if students indicated they had prior programming experience.

Choose the phrase that best describes your level of experience with each of the following languages:

Part I – General Computing Background

1. Choose the statement that best describes your level of comfort with using computers.
   a) I am very comfortable and have used computers extensively.
   b) I am comfortable but have not used them extensively.
   c) I am moderately comfortable with computers.
   d) I am a little uncomfortable using computers.
   e) I am very uncomfortable using computers.

Enter your level of experience on each of the following computer platforms:

2.	Windows (any version)	a) none	b) a little	c) some	d) a lot
3.	Macintosh	a) none	b) a little	c) some	d) a lot
4.	Linux/Unix	a) none	b) a little	c) some	d) a lot
5.	Other	a) none	b) a little	c) some	d) a lot

6. Do you have programming experience (excluding HTML)? a) YES b) NO

If you answered NO to question #6, please skip to Part III of the survey now.

If you answered YES to question #6, please continue with Part II of this survey.

Part II – Computer Programming Experience

Do not answer these questions unless you answered YES to question #6 in Part I above. Choose the phrase that best describes your level of experience with each of the following languages:

7.	C Language	a) none	b) a little	c) some	d) a lot
8.	C++	a) none	b) a little	c) some	d) a lot
9.	Java	a) none	b) a little	c) some	d) a lot
10.	Visual Basic	a) none	b) a little	c) some	d) a lot
11.	Pascal	a) none	b) a little	c) some	d) a lot
12.	Fortran	a) none	b) a little	c) some	d) a lot
13.	Other (excluding HTML)	a) none	b) a little	c) some	d) a lot

Choose the phrase that best describes your understanding of the programming language topics below.

14.	variables, constants and data types	a) not at all	b) weak	c) moderate	d) strong
15.	logic structures: sequence	a) not at all	b) weak	c) moderate	d) strong
16.	logic structures: decision (if)	a) not at all	b) weak	c) moderate	d) strong
17.	logic structures: iteration (loop)	a) not at all	b) weak	c) moderate	d) strong
18.	methods or procedures	a) not at all	b) weak	c) moderate	d) strong
19.	arrays	a) not at all	b) weak	c) moderate	d) strong
20.	encapsulation, inheritance & polymorphism	a) not at all	b) weak	c) moderate	d) strong

High School

21. Did you learn to program as part of a high school class (either formally or informally)? (circle)

    a) YES b) NO

If you answered YES to question # 21, please answer the following three questions about your high school classes.

22. How many courses did you take that involved some computer programming?

    a) one
    b) two or three
    c) more than three

23. Did you have at least one course that focused primarily on computer programming?

    a) YES b) NO

24. Did you take an Advanced Placement (AP) Computer Science course?

    a) YES b) NO

College

25. Did you learn to program as part of a college course?

    a) YES b) NO

If you answered YES to question # 25, please answer the following two questions.

26. How many courses did you take involving programming?

    a) one
    b) two to three
    c) more than three

27. Was this programming experience part of another course (e.g. Math, Business, Science, etc.)

    a) YES b) NO

Other Educational Experiences

28. Did you learn to program as part of a summer camp or other short program?

a) YES b) NO

If you answered YES to question # 28, please answer the following question about your experience.

29. Estimate how many days you spent programming.
    a) less than 3 days
    b) 3 to 5 days
    c) 7 to 14 days
    d) more than 14 days

30. Did you teach yourself to program?	a) YES	b) NO
31. Did you program as part of a computer club?	a) YES	b) NO
32. Do you program for enjoyment?	a) YES	b) NO

Job Experiences

33. Did you have a job that involved programming?	a) YES	b) NO

If you answered YES to question #33, please answer the following two questions about your work.

34. How many hours a week did you program? (Circle one)
    a) full time (roughly 30 hours or more per week)
    b) half time (roughly 20 to 30 hours per week)
    c) less than half time (less than 20 hours per week)

35. How long did you program at work?
    a) less than 3 months
    b) 3 to less than 6 months
    c) 6 to less than 12 months
    d) 12 to less than 18 months
    e) 18 months or more

Appendix B: Course Descriptions for Core IT Threads

Programming and Application Development Thread

4002-217 PROGRAMMING FOR IT I

This is the first course in the introductory programming sequence required for all Information Technology students. Topics include elementary data types, arithmetic and logical operations, control structures and error handling, methods and functions, and an introduction to object-oriented programming design and implementation. Emphasis is placed on the development of problem-solving skills. Programming projects are required. (Prerequisite: computer literacy)

4002-218 PROGRAMMING FOR IT II

This is the second course in the introductory programming sequence required for all students majoring in Information Technology. Topics include further exploration of classes and objects, programming through composition and inheritance, reusability, input/output, and object oriented design. Emphasis is placed on the development of problem-solving skills. Moderately large programming assignments are required. (Prerequisite: 4002-217)

4002-219 PROGRAMMING FOR IT III

This is the third course in the introductory programming sequence required for all students majoring in Information Technology. Topics include advanced interface concepts, traditional programming data structures, programming utilities and reusability, introductory project design and management concepts and other concepts as time permits. Emphasis is placed on the development of problem-solving skills. Large programming assignments are required. (Prerequisite: 4002-218 or 4002-221)

4002-220 PROGRAMMING FOR IT IIA

This is the first of two courses that is equivalent to 4002-218. 4002-218 is the second course in the introductory programming sequence required for all students majoring in Information Technology. This course and the subsequent one (4002-221) are designed to cover the same materials covered in 4002-218. These two courses are designed for students that find programming difficult and would like to have more time to learn OOP concepts and programming techniques. Topics include further exploration of classes and objects, programming through composition and inheritance, reusability, and object-oriented design. Emphasis is placed on the development of problem-solving skills. Moderately large programming assignments are required. (Prerequisite: 4002-217)

4002-221 PROGRAMMING FOR IT IIB

This is the second of two courses that is equivalent to 4002-218. 4002-218 is the second course in the introductory programming sequence required for all students majoring in Information Technology. This course and the previous one (4002-220) are designed to cover the same materials covered in 4002-218. These two courses are designed to help those students that find programming difficult and would like to have more time to learn OOP concepts and programming techniques. Topics include further exploration of classes and objects, programming through composition and inheritance, reusability, and object-oriented design. Emphasis is placed on the development of problem solving skills. Moderately large programming assignments are required. (Prerequisite: 4002-220)

4002-360 INTRODUCTION TO DATABASE & DATA MODELING

A presentation of the data modeling process and database implementation fundamentals. Data modeling, fundamental relational concepts, the process of normalization, relational algebra, SQL, and guidelines for mapping a data model into a relational database will be covered. Students will model a multimedia or text-only information problem and implement it with a commercially available database package. (Prerequisite: 4002-218 and Discrete Math)

Web and Multimedia Development

4002-320 INTRODUCTION TO MULTIMEDIA

This class provides an introduction to key Internet, web, and multimedia technologies, as well as familiarity with the Macintosh computer platform. Topics covered include computer-mediated communication, basic Internet applications such as telenet, FTP, and the WWW, basic digital image, audio, and video techniques, and web page development and publishing. (Prerequisite: computer literacy)

4002-330 INTERACTIVE DIGITAL MEDIA

Students will create interactive multimedia content for CD-ROM and the World Wide Web. They will capture, combine control and synchronize video, audio, text and images using authoring environments such as Macromedia Director. Students will write event handlers to control interactive applications. Programming will be required. (Prerequisite: 4002-320 and 4002-217)

Hardware and Networking Thread

4002-340 COMPUTER CONCEPTS & SOFTWARE SYSTEMS

A study of the concepts of computer hardware design and organization needed for effective system implementation. Topics include: computer peripherals and interfacing techniques, Boolean algebra, digital logic design, integrated circuit families, central processing unit design, buses and addressing, interrupts and direct memory access, hierarchical memories, system performance evaluation and a survey of commercially available computers. (Prerequisite: Discrete Math and 4002-217)

4002-341 DATA COMMUNICATIONS & COMPUTER NETWORKS

This course provides an introduction to data communications hardware and software, and use of these components in computer networks. Topics include, but are not limited to, communication e system components, communications software, packet switching, common carrier issues, wide area networks vs. local area networks, and performance considerations over different media. (Prerequisite: 4002-340)

4002-342 INTERNETWORKING LAB

This course is a laboratory-based course on the interconnection of digital devices for the purpose of enabling data communication. The focus is on the hardware, software, and protocols for peripheral and network communication, supported with a substantial laboratory component. Accessing computers and networks from a remote site will also be studied. Students will be required to construct cables, install network cards, configure modems and establish a variety of working connections between various digital devices. Problems may be introduced into working systems and students will be required to use diagnostic tools (both software and hardware) to determine and repair the problem. (Prerequisite: 4002-340 and 4002-341; co-requisite 4002-342 lab)

References

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education* (pp. 49-52). New York, NY: ACM Press.

Franklin, R. (1987). What academic impact are high school computing courses having on the entry-level computer science curriculum? *Proceedings of the Eighteenth SIGCSE Technical Symposium on Computer Science Education* (pp 253-256). New York, NY: ACM Press.

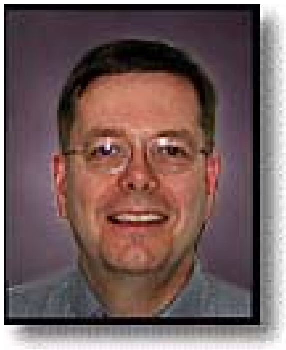
Hagan, D., & Markham, S. (2000). Does it help to have some programming experience before beginning a computing degree program? *Proceedings of the 5th annual SIGCSE/SIGCUE ITiCS Econference on Innovation and Technology in Computer Science Education* (pp 25-28). New York, NY:ACM Press.

Holden, E., & Weeden, E. (2003). The impact of prior experience in an information technology programming course sequence. *Proceedings of the 4th Conference on Information Technology Education* (pp 41 – 46). New York, NY:ACM Press

Holden, E., & Weeden, E. (2004). The experience factor in early programmer education. *Proceedings of the 5th Conference on Information Technology Education* (pp 211-218). New York, NY: ACM Press

Taylor, H., & Mounfield, L. C. (1989). The effect of high school computer science, gender, and work on success in college computer science. *Proceedings of the Twentieth SIGCSE Technical Symposium on Computer Science Education* (pp 195-198). New York, NY: ACM Press.

Biographies



Ed Holden is an Assistant Professor in the Information Technology Department of the Golisano College of Computing and Information Sciences at Rochester Institute of Technology (RIT). He teaches courses in programming, database management, technology transfer, needs assessment, e-commerce and process management. Prior to joining RIT full-time, Ed spent 28 years in the Information Systems business at a major corporation. In his last positions he served as the manager of Global Infrastructure Development for Messaging and Groupware and worked on a special assignment to integrate digital business efforts with back office systems and operations. Prior to this he was the manager of Messaging and Groupware for US and Canada where he supervised the outsourcing of email and groupware and the migration of 30,000 users to a new email and groupware system. He also worked on the preliminary design and requirements definition for the company's business-to-business e-commerce initiative.

Ed has performed all phases of the systems life cycle from proposal through continued support and the negotiation and management of outsourcing contracts. His clients have included finance, marketing, sales, research and development, and supply chain management.

Ed holds a BA in Math from SUNY Oswego (1972) and an MBA, Finance, from RIT (1995).

Elissa Weeden is an Assistant Professor in the Information Technology Department of the Golisano College of Computing and Information Sciences at Rochester Institute of Technology (RIT). Her areas of expertise are in database design and implementation as well as applications programming. She consults professionally and regularly within those areas. Her research and teaching interests include: data modeling, database implementation and administration, active learning, and assessment methods.

Elissa holds a BS in Information Technology and a MS in Software Development and Management, both from RIT. She is currently working towards her Ph.D. in Computing Technology in Education from Nova Southeastern University.