Issues in Informing Science and Information Technology Volume 3, 2006

What Makes Valuable Pre-experience for Students Entering Programming Courses?

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

For the past several years, the authors have been studying the impact of prior experience on performance in introductory Information Technology (IT) courses. Since 2002, data has been collected on all incoming freshmen and performance has been measured by the grade received in initial courses. The grades are expressed in the traditional four-point scale used at most US colleges and universities.

Prior studies (Holden & Weeden, 2003, 2004, 2005) have used an experience index to determine the level of prior experience possessed by students entering the IT undergraduate program. The index has also been used to place students in appropriate classes. This study looks at the components of the formula used to calculate this index as well as some informal experience information that is collected as part of the survey. It concludes with a revised version of the experience index formula which will be used to place students into cohorts in the future.

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

Introduction

In the Information Technology (IT) department at Rochester Institute of Technology (RIT), the authors started looking at retention in initial programming courses in the IT undergraduate program. It was observed that certain students seemed to be unintentionally intimidated by other students who appeared to have more experience. This caused the students without prior experience to not become fully engaged in the course. A formula was developed to predict performance in the first course in the programming sequence (Holden & Weeden, 2003).

It was then decided to group incoming students into cohorts based on their score using the formula, under the assumption that the students without experience would not be intimidated. This series of studies was used to examine student performance as measured by the grade received in the class (Holden & Weeden, 2003, 2004, 2005). An earlier study by Wilson and Shrock (2001) indicated that students performed better when they are comfortable in class. One way to increase comfort would be to have a more homogeneous group.

Since 2002, data has been collected on all incoming freshmen students beginning the Information Technology program at RIT. This data has shown the types and level of experience possessed by these students. Course performance has been measured by the grade received in initial courses. These grades are expressed in the traditional four-point scale used at most US colleges and universities.

The formula was developed and used in our prior studies (Holden & Weeden, 2003, 2004, 2005) to analyze the experience level of students entering the program. The formula has been used to determine the depth of experience of each incoming freshman. The formula focused on the formal experience that a student can encounter before entering the program. The formula included formal high-school courses, college courses, work experience and exposure to object-oriented concepts only and is documented in Appendix A. It did not include informal activities. This is similar in concept to the formula developed by Wilson and Shrock (2001) to determine experience level except that they included other factors that we did not consider relevant for our study. Other factors are evaluated separately later in this paper. These include other ways to gain experience that are not included in the formula and include programming for enjoyment, self teaching, summer camps or other short programs, and clubs.

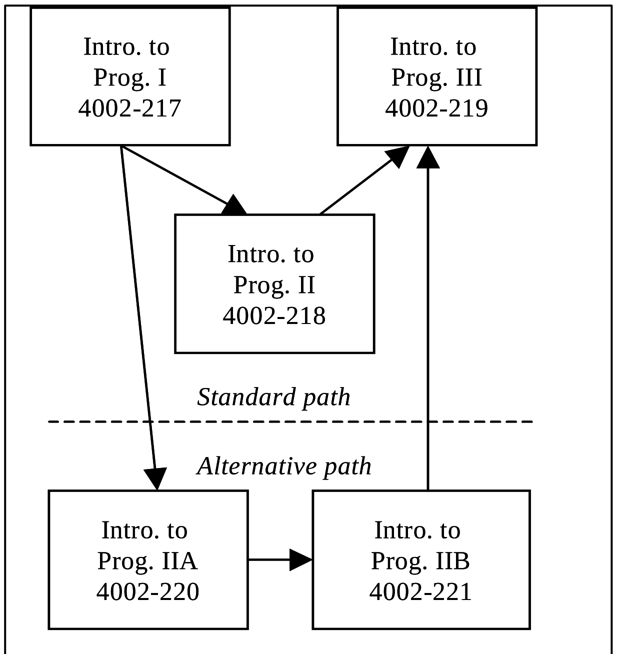
Intro. to Prog. I 4002-217
Intro. to Prog. III 4002-219
Intro. to Prog. II 4002-218
Standard path
Alternative path
Intro. to Prog. IIA 4002-220
Intro. to Prog. IIB 4002-221

Figure 1: Introductory Programming Sequences

This paper will layout the results of prior studies (Holden and Weeden, 2003, 2004, 2005) and report on the sources of prior experience both inside and outside the formula mentioned above, and how this experience impacts the performance, as measured by grade in the first programming course, Introduction to Programming I (4002-217). This course is the first of a three or four course sequence, as documented in prior studies (Holden and Weeden, 2003) and is illustrated in Figure 1.

Methodology

Since 2003, a survey has been sent to all students in the summer before their freshman year. Students were asked to return the survey before they would be registered for their programming courses. This survey investigated five areas.

- General computing background
- Computer programming experience
- RIT and program affiliation
- About yourself
- Learning styles

Only the first two areas are covered in this study. This study was aimed at gaining insight into prior student experience in programming and individual motivation to program. The survey data was collected for later analysis when course grades were available. The questions related to this study are included in Appendix B.

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 gave more insight into the nature of that experience.

Student performance as measured by grade on a four point scale in the first courses in the programming sequence was compared to the data gathered in the survey.

The remainder of the survey is used in other studies.

The 2002 survey was less formal and contained only the first two sections. It was conducted during the first quarter of the freshman year.

Results of Prior Studies

In anecdotal discussion of the impact of prior experience, faculty members have been mixed on their attribution of value of prior experience. Some say that it is a good predictor of success while others have said that it was actually detrimental because students had to unlearn bad practices.

Other studies have looked at the impact of experience on performance in computing courses, but most of theses have been focused on Computer Science courses rather than IT (Byrne & Lyons, 2001; Franklin, 1987; Hagen & Markham, 2000; Taylor & Luegina, 1989).

Several studies have concluded that prior experience does have an impact on student performance (Byrne & Lyons, 2001; Hagan & Markham, 2000), while others have had the opposite result (Bergin & Reilly, 2005; Ventura & Ramamurthy, 2004)

Wilson and Shrock (2001) differentiated between programming experiences from formal course work compared to informal experience. In earlier studies (Holden & Weeden, 2003, 2004, 2005), only formal and work experience was considered, not informal experience. Informal experience will be considered as a separate item in this report. Wilson and Shrock found that programming experience, both formal and informal, as part of their model, did not have a significant impact, but previous programming coursework did.

Prior results found by Holden and Weeden (2003, 2004, 2005) include:

- Prior programming experience has 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.
- Students with prior experience are more likely to complete a faster three course sequence than their inexperienced peers. The alternative is a four course sequence covering the same material.
- 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 covered in 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.
- 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 either too 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.
- Students' indication of "comfort level" with computers is not an indication of future performance in programming courses.
- One programming language used in the prior experience does not seem to indicate future success more than others.
- Formal educational experience does improve performance in the first course in the sequence.

In this paper, the authors examine the components of the formula used in the prior studies to see if one component has more impact than others. It will also be used to modify the method of determining experience as outlined in Appendix A.

Where Do Students Get Prior Experience?

A survey done in 1987 found the 34% of 321 students had prior experience in a high school course (Franklin, 1987). In 2002 Holden and Weeden (2003) found that 53% of 159 students sampled had prior experience through a high school course, an increase of 19% in fifteen years.

In the combined survey from 2002 to 2005, the authors have surveyed 525 students. Many of these students indicated that they had prior experience in programming. It was found that 56% of the incoming freshman gained programming experience in a high school course.

In addition 10% had experience in a college course before entering our first programming course.

The Information Technology department also had 8% of incoming students entering with programming experience from work.

The surveys used also included questions about informal experience that has not until now been analyzed. The questions asked if the incoming students programmed for enjoyment, taught themselves to program, learned to program as part of a club, or learned to program at a summer camp or other short program.

From this data it was found that 38% of our incoming students programmed for enjoyment, 31% taught themselves to program, 6% learned to program as part of a club, and 6% learned to program as part of a summer camp or other short program. These results are summarized in Table 1.

All-in-all a large percentage of the incoming students have some experience before they arrive in the Introduction to Programming I course.

Prior studies by Holden and Weeden (2003, 2004, 2005) have indicated that the formal experience does have a positive impact on performance. We will now isolate the types of experience to see what the impact of each component is.

Table 1: Where students get prior programming experience. Note that students may have more than one type of experience.

Student Category	Number of Students	Percentage of Students
Total students in survey	525	
Formal experience		
High school experience	294	56%
College experience	50	10%
Work experience	41	8%
Informal experience		
Programmed for enjoyment	197	38%
Self taught	165	31%
Clubs	32	6%
Summer camp or other short program	33	6%

Dissecting the Experience Formula

The experience formula documented in Appendix A is broken down into four parts, high school experience, college experience, work experience and exposure to advanced concepts. The analysis began by looking at the first three parts since advanced concepts exposure was a result of the first three. When the formula was dissected, some interesting results were found.

A total of 525 students who entered the Introduction to Programming I course, using Java, from 2002 through 2005 were included in the study. Of these students, 318 fell into the experienced categories while 207 had no experience. The average grade of students in the experienced group was 2.86 while those with no experience only averaged 2.33, 0.53 points lower than the experienced. An independent sample *t*-test showed a significant difference (p=0.000) between the non-experienced group and the experienced group in the first course and is consistent with the results of the prior research.

To isolate the components, the experienced group was further broken down into the type of experience used in the experience formula, high school only, college only, work only and a mixture of high school, college or work. An ANOVA comparing the grades earned in the Introduction to Programming I course showed no significant difference between the experience types used in the experience formula (p=.336).

A total of 233 students had only high school experience. These students averaged 2.91, which was also consistent with the overall experienced students. Similarly, the 64 students with a mixture of experience had a 2.75 average, again consistent with the experienced group.

The 12 students in the college only group only scored an average of 2.25. These results are not significant due to the small sample size. The evaluation of experience was done from the students' perspectives. The sample size is low here due to two factors. First, most of the students who enter this course are first-time college students who have not taken a college programming course. Second, students who have had a college experience equivalent to this course, as determined by the department’s undergraduate coordinator, are allowed to waive this course. This would cut the number of students even further. The lower average is explained by the fact that the undergraduate coordinator determined that students taking this course had a less rigorous experience than the other students who had a college-only experience.

The nine students who had work-only experience scored higher than the overall experienced group, with an average grade of 3.11. This result however is not significant due to the small sample size. Again, the sample is small because most entering students come from high school, and have not entered the work force as programmers. These students often represent non-traditional students who have entered our program. These results are shown in Table 2.

Table 2: Average grade in Introduction To Programming I by type of experience

Type of Experience	**Number of Students**	**Average Grade in Introduction to Programming I (4.0 scale)**
High School Only	233	2.91
College-Only	12	2.25
Work-Only	9	3.11
Mixture	64	2.75
Total Experienced	***318***	***2.86***
No Experience	207	2.33
All Students	***525***	***2.65***

The fourth factor in the experience formula was exposure to the advanced, object-oriented programming (OOP), concepts of polymorphism, encapsulation and inheritance. The 2002 survey had a slightly different question than the 2003 though 2005 surveys. The 2002 survey only asked if the students had "learned" the concepts (Yes or No). The 2003 through 2005 survey asked students to describe their level of understanding: not at all, weak, moderate, or strong. The 2002 students were considered to have had exposure to OOP if they selected "Yes", while the 2003 through 2005 students were considered to have had exposure to OOP if they selected a moderate or strong understanding.

The results showed that all experience categories had higher average grades if they had exposure to OOP concepts than those who did not. Of the 233 students who had high school only experience, 42 had exposure to OOP concepts. Those 42 had an average grade of 3.23, 0.32 above their peers without the OOP exposure. A 1-tailed independent sample *t*-test indicated that a students that had programming experience in high school and were exposed to OOP concepts performed significantly higher in Introduction to Programming I than students that had programming experience in high school but were not exposed to OOP concepts (p=.04).

The students with college-only experience who had exposure to OOP concepts averaged 2.33, 0.08 above their inexperienced peers. The sample was small (3/12) and the difference between the groups was not significant (p=.46) given a 1-tailed independent sample t-test. Likewise, those with only work experience and exposure to OOP concepts finished with a 4.00, 0.89 above their peers without OOP exposure. But again, the sample is small (2 / 9) and the results using a 1-tailed independent sample *t*-test indicated that there was no significant difference (p=.051). College-only experience will no longer be included because of the introduction of the undergraduate coordinator's process mentioned earlier, which causes a downward bias on the grades.

The 16 students who had a mixture of types of experience, who also had exposure to OOP concepts had a significant improvement over their 64 peers without exposure to OOP concepts. The former averaged 3.31, 0.56 above the latter. The results where shown to be significant through a 1-tailed independent sample t-test (p=.038).

In general, the 318 experienced students averaged 2.86. The 63 students who had exposure to OOP concepts demonstrated better performance (3.24) than the 255 who did not (2.76). This difference is shown to be significant through a 1-tailed independent sample *t*-test (.005). These results are summarized in Table 3.

Table 3: Average grade in Introduction To Programming I for experienced students with and without exposure to advanced OOP concepts.

Type of Experience	Number of Students w/Advanced Concepts	Average Grade in Intro to Prog I (4.0 scale)	Number of Students w/o Advanced Concepts	Average Grade in Intro. to Prog. I (4.0 scale)
High School Only	42	3.24	191	2.83
College-Only	3	2.33	9	2.22
Work-Only	2	4.00	7	2.86
Mixture	16	3.31	48	2.56
All Types	***63***	***3.24***	***255***	***2.76***

Other Factors Outside the Experience Formula

The experience formula detailed in Appendix A does not include informal experience like self training, clubs, summer camps or other activities. The surveys did collect information on these areas.

When looking at these informal factors, some interesting results were found. Of the 318 experienced students, 217 students (68%) were involved in one or more of the informal activities. The experienced students had an overall average of 2.86. Of these students, 174 said that they wrote programs for enjoyment. These students had an overall average of 2.97, 0.11 above the average for the experienced group. Similarly the 142 experienced students who taught themselves to program had an average of 2.99, 0.13 above the experienced group.

The opposite results were seen when the non-experienced group was examined. Only 27 (13%) of the 207 students took part in these informal activities. The 207 students who had no prior experience had an average of 2.33. The 23 students who programmed for enjoyment had an average of 2.48, 0.15 above the non-experienced group. The 23 students who were self-taught had an average of 2.65, 0.32 above the non-experienced group.

This may have indicated that these students had more motivation than the others to learn to program.

The 31 experienced students who belonged to a club fared no better in the first programming course that their experienced group as a whole. They averaged 2.87 compared to 2.86 for the larger group.

The questions concerning learning to program at a summer camp or other short term program were not included in the 2002 survey, so these results are from the later surveys.

The experienced students who learned to program as part of a summer camp or other short program actually fared worse than the whole experienced group. These 30 students averaged 2.47, 0.39 less than the larger group.

The sample was too small for the non-experienced students who belonged to a club or attended a summer camp or other short program to make judgments about the results. Only one of these students belonged to a club and three went to a summer camp or other short program. Table 4 summarizes these results.

Table 4: Average grade in Introduction To Programming I by type of informal experience

Experience Type	Number of Students	Average Grade in Intro. To Prog. I
All Experienced	318	2.86
Experience - Program for Enjoyment	174	2.98
Experience - Self Taught	142	2.99
Experience - Club	31	2.87
Experience - Summer Camp	30	2.47
No Experience - Program for Enjoyment	23	2.48
No Experience - Self taught	23	2.65
No Experience - Club	1	2.00
No Experience - Summer Camp	3	3.67
All No Experience	207	2.33

Restating the Experience Formula

When the components of the experience formula documented in Appendix A were examined more closely for possible refinement, it was noted that various weights based on the type of experience that the student had were not always appropriate. For example, more weight was given to college courses than high school and much more weight was give to work experience over high school. The formula was based on some assumptions derived from discussions with faculty and students. The assumptions were then added together. One assumption was that these factors were additive, implying that the more, different types, of experience the better. The data suggest that for placement in an introductory programming course, these factors are not always be additive as can be seen in Table 2.

As shown in Table 2, all 318 students that indicated they had programming experience received an average grade of 2.86, 0.53 above 2.33 that the non-experienced students received. The average for the 233 students who had only high school experience was 2.91, slightly above the total experienced group, but consistent with the overall average for the whole experienced group. The group of nine students who only had work experience had an even higher average (3.11). The 64 students who had a mixture of experience averaged 2.75, slightly below the average for the total experienced group, but consistent with the experienced group.

One conclusion that can be drawn from this is that experience is not additive since the average grade for students who had only one type of experience was consistent with those with a mixture of experience.

Students with only college experience were left out of this discussion. This was because of the way transfer credit is handled within the department by the undergraduate coordinator, as described earlier.

Based on this analysis, either high school or work experience will be counted as determining an experienced student, but not both, since they do not appear to be additive. College experience will no longer be used in this situation, as it does not apply for the reasons mentioned earlier. This exclusion of college experience may not apply at other institutions.

The old formula added one point if the student had experience with the advanced concepts of polymorphism, encapsulation and inheritance. The new formula will continue this. This is based on the results shown in Table 3. In all types of experience the students averaged higher that those without the concepts.

The old formula did not include the informal activities shown in Table 4. Experienced students that programmed for enjoyment or taught themselves to program averaged 0.12 and 0.13 higher than the overall average for experienced students, respectively. These additional factors will be included in the new formula.

The New Formula

With this new information, a new formula has been developed to assess the experience level of each individual in the sample.

- The high school experience will continue to be assessed the same way. 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.
- For the reasons mentioned above, college experience will not be included. For other universities, this may be a factor that should be included if they do not do the same screening that is done by the undergraduate coordinator.
- Work experience will continue to be assessed the same way but with the weights assigned for time worked reduced. First, a factor was assigned based on the type of position: Full time = 1 point, part time = 0.5 point, and less than part time = 0.25. This was multiplied by a factor reflecting the number of years they held a job: Less than six months = 0.5, 6 months or more but less than 18 months = 1, 18 months or more = 2.
- The experience index will be the maximum of the high school score or the work score, plus one point if the student had learned about the object-oriented programming concepts of polymorphism, encapsulation and inheritance.
- One-half point will be added if the student has informal experience from programming for enjoyment or being self-taught. This was not included in the original formula.
- This index will be used as the experience index for each individual, with a maximum possible total of 3.5.

This formula is closer to the reality that different types of experience are not as radically different as indicated in the old formula. It recognizes that students with college experience are removed from the sample by the undergraduate coordinator if their experience is equivalent to the introductory course. It also recognizes that the different types of experience are not necessarily additive as shown through the ANOVA performed across the experience types shown in Table 3.

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

- No experience: experience index = 0
- Average experience: 0 < experience index <=2
- High experience: 2 < experience index

Using this new formula, the experience levels for students in the sample ranged from 0 to 3.5 out of the maximum possible index of 3 points. The average is 1.1. Out of the 404 students, 142 had no experience, 178 had an average level of experience and 84 had a high level of experience

Conclusion

Based on the new evaluation of this analysis, IT departments may want to consider cohorting students into groups based on the new experience formula, which gives a more realistic view of the students' prior experience. This will allow tailoring of course material and pace to individual student needs.

This, of course, will depend on the number of course sections being offered. In some situations it may be necessary to combine groups for logistical reasons.

Future Study

In future studies we will continue examine and refine the experience formula. In addition, other experiences that may impact student performance should be examined and considered. This includes the items in Table 3 as well as other factors that are included in the survey.

Also, a separate study is examining the cohort data to determine if cohorted students actually perform better than non-cohorted students.

References

Bergin, S., & Reilly, R. (2005). Programming: Factors that influence success. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, 37*(1), 411-415. Retrieved February 9, 2006, from ACM Digital Library database.

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, 33*(3), 49-52. Retrieved February 9, 2006, from ACM Digital Library database.

Franklin, R. E., Jr. (1987). What academic impact are high school computing courses having on the entry-level college computer science curriculum? *Proceedings of the Eighteenth SIGCSE Technical Symposium on Computer Science Education, 19*(1), 253-256. Retrieved February 9, 2006, from ACM Digital Library database.

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 ITiCSE Conference on Innovation and Technology in Computer Science Education, 32(3), 25-28. Retrieved February 9, 2006, from ACM Digital Library database.

Holden, E., & Weeden, E. (2003). The impact of prior ecperience in an information technology programming course sequence. In *Conference on Information Technology Education. Proceedings of the 4th conference on information technology education* (pp. 41-46). New York: ACM Press. Retrieved February 9, 2006, from ACM Digital Library database.

Holden, E., & Weeden, E. (2004). The experience factor in early programming education. In *Conference on Information Technology Education. Proceedings of the 5th conference on information technology education* (pp. 211-218). New York: ACM Press. Retrieved February 9, 2006, from ACM Digital Library database.

Holden, E., & Weeden, E. (2005). Prior experience and new IT students. *The Journal of Issues in Informing Science and Information Technology, 2*(1), 189-204. Retrieved February 9, 2006, from http://iisit.org/IssuesVol2v2.htm

Taylor, H. G., & 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, 21*(1), 195-198. Retrieved February 9, 2006, from ACM Digital Library database.

Ventura, P., & Ramamurthy, B. (2004). Wanted: CS1 students, no experience required. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, 36*(1), 240-244. Retrieved February 9, 2006, from ACM Digital Library database.

Wilson, B. C., & Shrock, S. (2001). Contributing to the success in an introductory computer science course: A study of twelve factors. *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education, 33*(1), 184-188. Retrieved February 9, 2006, from ACM Digital Library database.

Appendix A - The Current Experience Formula Used in this Study

People with prior programming experience filled in part II of the survey included in Appendix B. This data was used to examine the depth of their programming experience. The following questions on the survey were used to assess how deep the experience was.

- Did you learn about the concepts of encapsulation and inheritance?
- 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?

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

The experience levels for students in the sample ranged from 0 to 7.0 out of the maximum possible index of 12 points.

Student experience levels were defined 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

Appendix B - Survey Questions

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)?
    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

<u>Other Educational Experiences</u>

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

<u>Job Experiences</u>

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?
    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

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 Mathematics 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.