Priorities for the Education and Training of Software Engineers

Timothy C. Lethbridge

School of Information Technology and Engineering
150 Louis Pasteur
University of Ottawa
Ontario
Canada K1N 6N5

Email: tcl@site.uottawa.ca

Phone: +1 613-562-5800x6685

Fax: +1 613 562-5187

Running Title: Priorities for the education of software engineers

Priorities for the Education and Training of Software Engineers

Timothy C. Lethbridge[1]

School of Information Technology and Engineering

150 Louis Pasteur, University of Ottawa, Ontario, Canada K1N 6N5

Abstract

We present the complete results of our 1998 survey of software practitioners. In this survey we asked over 200 software developers and managers from around the world what they thought about 75 educational topics. For each topic, we asked them how much they had learned about it in their formal education, how much they know about it now and how important the topic has been in their career. The objective of the survey was to provide data that can be used to improve the education and training of information technology workers. The results suggest that some widely taught topics perhaps should be taught less, while coverage of other topics should be increased.

Keywords: Software Engineering Education; Computing Education; Software Engineering Body of Knowledge

1. Introduction

This paper presents the results of a survey of software practitioners conducted during the summer and autumn of 1998. The survey was designed to discover what knowledge is important to the participants, and to better understand their educational and training needs.

Our motivations for conducting this survey arose from our interactions with software practitioners and managers: During these interactions, it became clear that the knowledge they were taught in their formal computing education did not always match the knowledge they needed to apply to their daily work. At the same time, we were establishing a new academic program in software engineering, and wanted data to help us make curriculum decisions.

A significant transition is currently occurring in computing education, especially in North America: Traditional programs in computer science and computer engineering are being joined by programs specializing in software engineering. It is important for designers and accrediters of such programs, as well as those who will be licensing the resulting software engineers, to have a full understanding about what the profession involves. To gain that understanding normal approaches include reading textbooks and research literature, and also talking to practitioners. From these sources, one can learn about the many ideas and principles that comprise the state of the art. However, it is very hard to get a view of the *relative importance* of the various knowledge areas, and also the extent to which current educational programs meet the needs of industry. To obtain this information, one needs a body of data from a systematic survey of practitioners.

This paper covers our second survey of software practitioners. The first was conducted in 1997 (Lethbridge, 1998a and 1998b) and served as a pilot study. The 1998 survey involved substantially better sampling and improved questions. In particular the 1998 survey was a collaborative effort among researchers at various universities in North America and the UK (for a list of collaborators, see the acknowledgements).

Data in this report can also be found in an on-line spreadsheet (Lethbridge, 1999a) and in an articles submitted to IEEE Software (Lethbridge, 1999b) and an 81-page technical report (Lethbridge, 1999c).

In the next section, we will discuss the survey instrument and how the survey was conducted. We will then look at important data from each of the questions, as well as information that can be obtained by comparing answers to the different questions. We leave discussion of the demographics of the participants to the end.

[1] Tel: +1 613 652-5800x6685; email: tcl@site.uottawa.ca; web: http://www.site.uottawwa.ca/~tcl

Readers interested in further information about software engineering education should consult the proceedings of the IEEE conferences on Software Engineering Education and Training. Another excellent source of information is the monthly FASE newsletter http://www.cs.ttu.edu/fase.

Various organizations are working to produce cohesive documents that try to describe the field of software engineering. These include the Working Group on Software Engineering Education (Hilbert, 1998), the IEEE group investigating accreditation (IEEE, 1998), and the IEEE/ACM Software Engineering Body of Knowledge project (Bourque, 1999).

2. The Survey Instrument

The survey was conducted from May to October 1998. Participants were solicited in two main ways: Members of the research team approached the management of various companies to have their employees participate; also, requests were sent to various email lists, postal mail lists (e.g. university computer science and computer engineering alumni) as well as several Usenet newsgroups.

The four distinct questions listed in Fig. 1 were asked about each of 75 topics listed in Fig. 2. The topics were determined by studying typical university curricula and by including topics that were suggested by participants in the 1997 survey.

Several versions of the questionnaire were prepared, each with the same topics but in a different order (this rearrangement was intended to reduce inter-question bias). In addition, we asked several demographic questions in order to learn about the educational, geographic and work backgrounds of the participants. Most participants completed the survey using the Web (see http://www.site.uottawa.ca/~tcl/edrel), however, a paper version was used by a small percentage of the participants.

The total number of responses was 214. Of these, 212 were complete and contained valid data. We wanted to achieve a balance of participants from various industries, yet the distribution of our 212 participants was somewhat biased towards real-time processing and away from MIS systems or data processing. We thus randomly eliminated a few of the real-time responses, and arrived at a more balanced sample of 181 participants that we used for further analysis.

3. Knowledge Learned in Formal Education: Question 1

In this section, we discuss the answers to question 1: In Tables 1 and 2, we present the topics about which participants said they learned most and the least respectively.

Three separate metrics are presented about each topic: a) The mean score given to each topic in question 1 (Tables 1 and 2 are sorted by this criterion); b) the percentage of participants who gave each topic a score of 4 or 5 (they learned *a lot* about the topic, or learned it *in depth*); and c) the percentage of participants who gave the topic a score of at least 2 (they learned the basics or more).

Some interesting observations about this data are as follows:

Five of the top 8 topics are from the mathematics category. We will see in section 5 that this contrasts starkly with data from the questions about importance of topics.

Participants did not in general think they became experts during their formal education. In fact, Calculus was the only topic for which more than half of the participants thought they had become experts.

There are only 13 topics with which over 66% of the participants had become ‘basically familiar’ during their education. These topics might be considered as the core of current educational programs.

3.1 Differences Among Subsets of Participants in Answers to Question 1

This section looks at differences in responses to question 1 from various demographic subsets. See section 8 for details of the demographic groups themselves.

Table 3 compares the education of those who graduated in the four years prior to 1998 (junior participants) with that of those who graduated prior to 1976 (experts). The junior participants learned more about computing topics, particularly those, such as object oriented technology, that have only become widely known in recent years. On the other hand, the expert participants had learned more about many non-computing topics (Lethbridge, 1999c).

Tables 4 and 5 provide information about how a computing education differs from an engineering education, in terms of how much was learned about various topics. For this analysis, we have grouped together all 49 engineering graduates (including 23 electrical engineers, 12 computer engineers and 17 other engineers). We have also grouped together into the 106-member computing category, the 88 computer science graduates, the 22 software engineering graduates, and the 10 graduates of information systems. We performed the calculations at this level of granularity to ensure we had reasonable sample sizes for the comparison.

Software topics are clearly learned far more by the computing graduates, while the engineers learn more about traditional engineering topics such as electronics and Fourier transforms, as well as physics. Engineers also have more background in such computing-related topics as digital signal processing and control theory.

4. Present Knowledge: Question 2

In this section we move on to examine the answers to question 2 in the survey: The *present* knowledge participants have about the topics.

Tables 6 and 7 give data for the three metrics discussed in the last section: The mean score, the percentage that know the topic very well, and the percentage that know the topic at least minimally. As the table shows, these three metrics provide very similar information about which topics are better known.

We see that all but three of the 25 most-known topics are related to computing, and the remaining three are 'soft' topics: Giving presentations, technical writing, and ethics and professionalism.

Specific programming languages and data structures are far out in front of all other topics in terms of both mean knowledge, and the percentage of those who are experts. These are followed by topics that have to do with design, as well as supporting technologies such as operating systems and databases.

The bottom 25 topics in Table 7 are neither extensively taught in university, nor significantly used on the job. If a company expects to start a project that requires any of these topics, then it would do well to find out if employees have sufficient knowledge and provide training courses as appropriate.

4.1 Differences Among Subsets of Participants in Answers to Question 2

Table 8 examines the current knowledge of managers (those 37 participants who spend 25% or more of their time performing management tasks), comparing it to the participants at large. Not surprisingly, 'management' itself is near the top of the list. Of the top 17 topics, all but two are very clearly management-related.

The table can provide guidance to those who desire to become managers: Among 'soft' topics, they should clearly study negotiation, marketing, psychology and leadership. They should, also, study process standards and software metrics.

Tables 9 and 10 compare the knowledge of programmers with that of the rest of the participants. Programmers are defined as those 102 participants who spend 25% or more of their time programming.

As expected, programmers know more about core programming topics such as algorithm design, programming languages, object technology and parsing. However, they know less about certain other topics that probably should be important to a programmer: Process standards, metrics, cost estimation and requirements.

5. Importance of Topics: Questions 3 and 4

An important objective of this survey was to determine the topics that software practitioners find most important to their work. As presented earlier (see Fig. 1), two distinct questions were asked about each topic: Question 3 asked about the importance of each topics' details, while question 4 asked about how much each topic had influenced the participant. In this section, we present several different analyses of the answers to these questions.

To compute overall importance, we simply take the average of questions 3 and 4. Results of this computation for all 75 topics are presented in Tables 11 through 13. The most interesting data is found near the top of Table 11, the most important topics that every software engineer should presumably know; and near the bottom of Table 13, topics that most software engineers clearly do not need to know.

5.1 Importance of The Details: Question 3

Tables 14 and 15 present the data for the importance of the details (question 3). Table 14 has most of the same topics in the same order as in Table 11 (combined overall importance), including the top five. It is interesting to note, however, several topics near the top that have risen in the rankings: Giving presentations, technical writing, project management and operating systems. Three of these are 'soft skills'. On the other hand human-computer interaction has fallen somewhat in the rankings: Perhaps this area teaches awareness of the issues, more than it teaches details.

Tables 15 shows the topics for which participants believe the details are least important. The presence of continuous mathematics as well as chemistry and electronics is notable in this table.

5.2 Influence of Topics: Question 4

The other question about importance (question 4) asked about how influential each topic has been to the participants, irrespective of whether the details have been important.

Table 16 presents the answers to this question using the three metrics that we have presented in earlier sections: Mean, percent rating 4 or 5, and percent rating at least 2.

Although it is clear that many of the same topics whose details are important, also prove influential, it is interesting to note where differences exist. The following topics have a higher ranking for influence than for details: Ethics and professionalism, and human-computer interaction / user interfaces.

It is interesting to note that 'specific programming languages' is at the top of Table 16: Not only are the details of programming languages important, but learning them has been more influential, on average, than anything else.

5.3 Forced Learning: An Alternative Measure of Importance

Table 17 presents a completely different approach to ascertaining the importance of topics. The premise is that if a participant did not learn much about a topic in his or her formal education, but has since increased his or her knowledge substantially in order to meet the requirements of work, then the topic must be important.

Table 17 lists topics with the highest increase in knowledge since completion of formal education, i.e. the differences between question 2 and question 1 are large. However only those people who had said that they knew almost nothing about the topic at the time they completed their education (scoring question 1 with 0 or 1) are considered. The number of people considered is listed in the last column of the table.

The table contains the same topics that were seen when using the other methods of computing importance, with only slight differences in the order.

Several institutions are in the business of retraining people from other fields so that they can enter the information technology field. The topics listed in Table 17 might be the most suitable material for such retraining.

Increase in knowledge, *irrespective* of the amount learned in education, is discussed in section 6.

5.4 Differences Among Subsets of Participants Regarding Overall Importance

This section examines the differences of opinion about overall importance (average of questions 3 and 4) for various demographic subsets of the participants. Tables 18 through 20 only include topics where there was a significance in overall importance from participants at large and where overall importance high. The tables are ordered by overall importance so as not to give false impressions about which topics are important.

Tables 18 and 19 list the topics that real-time developers and MIS developers, respectively, consider more important than others. Certain topics are on the lists as expected: Databases in the MIS list, and real-time design on the real-time list. However it is notable that technical writing and giving presentations are of particular importance to real-time developers, while entrepreneurship is of particular interest to MIS developers.

Table 20 lists topics that managers consider more important than developers at large. Process standards, marketing, management, project management and cost estimation are the topics with the greatest increase in importance in the opinion of managers. These same topics are also found near the top of Table 8, the topics which managers know most about.

6. On-The-Job Knowledge Change

In this section we examine the difference between responses to question 2 (current knowledge) and responses to question 1 (knowledge after education). This difference represents the amount of on the job-learning when it is positive, or forgetting when it is negative.

If there is much on-the-job learning of a topic, then that topic must, to some extent, be important. We cannot use the difference as a direct measure of importance, however, because if a topic has already been extensively taught then it will naturally have lower on-the-job-learning.

The most important application for this data is to find out cases where formal education should be improved so that the amount of on-the-job learning needed is less. Similarly, if there is net on-the-job forgetting, then this suggests reductions in the amount of formal education that should be devoted to the topic in question.

One factor that was not measured in this survey was the ease with which a given topic can be learned on the job. It seems clear that a topic like configuration management, which appears at the top of Table 21, should be relatively easier to learn on the job than a complex topic like a branch of mathematics, or real-time design. Future researchers might consider studying this issue. This information would help educators know whether in fact it is necessary for them to provide coverage of topics listed here, or whether they can safely rely on the workplace to provide the needed training and experience.

Table 21 shows two different ways of measuring on-the-job learning: The mean difference between questions 2 and 1, as well as the percentage of people reporting a large difference.

It may be considered a waste if students forget significant amounts of what they have learned following graduation. Table 22 shows the topics where this has happened: The table starts with continuous mathematics, and also contains other mathematics as well as basic science and theoretical material.

It might be the case that mathematical and theoretical material is more prone to be forgotten, no matter how important it is. However, when the forgetting data is compared with the data about which topics are unimportant from the last section, we see considerable of correlation (the linear correlation coefficient is 0.99).

7. Needs for Learning and Training

The data presented in sections 5 and 6 provide insights into which topics are important and therefore should be included in education and training programs. However, additional analyses presented in this section can help inform us where the biggest gaps lie between what is taught or known and what is important. Knowing this information can help us to focus the curriculum improvement process on the most needed topics.

Table 23 presents the difference between the importance of details and the current knowledge of practitioners. A large lag suggests that although practitioners recognise the importance of the topic's details, they don't feel that, relative to its importance, they have adequate knowledge. We suggest that topics in this table might be the most suitable topics for focus by corporate training programs.

For Table 23, we have selected only the topics that have an overall importance greater than 3 because trainers probably would not care about a large lag less important topics. We have also ordered the table by importance; ordering it by lag would have placed some topics near the top that are only moderately important (e.g. data transmission), and we thought that this would be misleading. Trainers should pay particular attention to topics that are both higher in the table and have a higher lag: Requirements gathering, technical writing and human-computer interaction / user interfaces.

Table 24 is structured similarly to table 23. This time, however, the comparison is between overall importance of topics and the amount that students learn in their higher education. The purpose of the table is to guide educators about which topics might need more emphasis. It is notable that the topics in both tables are very similar, and in a similar order. Again we see requirements analysis and user interfaces as having both high lags and high importance.

Table 25 presents topics at the other end of the scale, those that are apparently taught far more in universities than their importance might warrant. It is interesting that all these topics are from the mathematics or basic science categories, and most of them are taught as compulsory topics to computing students.

Despite being less important in the eyes of practiioners, the topics on this table are taught in university programs for a variety of reasons, some of which are:

- Some of the topics are considered by many educators to be truly useful to computing professionals in their work, even if, according to the survey, software practitioners do not recognize it. This is would seem to be particularly true for set theory as well as probability and statistics. Both of these topics might come to be seen as more important if formal methods and statistical analysis of metrics come into more widespread use. It might therefore be wise for educators to keep these topics in curricula, or even to increase coverage, so future professionals are adequately prepared.
- The topics perhaps should be taught in case students need to work in application domains that require the knowledge – especially since such knowledge is not the kind that can be readily picked up on the job.
- The topics are useful to other scientists and engineers, and hence many believe that they must be taught to computing professionals so they can communicate with colleagues and be considered *real* scientists and engineers. This reasoning is widely used to make curriculum decisions in other disciplines: For example, certain medical specialists such as psychiatrists have to learn a full spectrum of medical knowledge even though they will only need a part of it: They are doctors, so they must learn a common core that all doctors know.
- Learning the topics perhaps helps create a well-rounded person with a broader educational background.
- Learning the topics might help discipline the mind.

We do not conclude absolutely whether the above reasons provide sufficient justification for keeping the topics in educational programs. However, given that our survey results show that certain topics might need greater emphasis, it seems logical to look near the top of this list for topics where emphasis can be reduced (e.g. differential equations, calculus and chemistry).

8. Demographics

In this section we describe the participants with respect to their educational and work backgrounds.

Table 26 indicates the fields in which participants had obtained degrees, while Fig. 3 describes the highest level of education they reached.

Table 27 and Fig. 4 provide information about the geographical origin of the participants. The survey can be said to be North-America biased since 76% of the participants came from this continent. It is interesting to note that all but 16 of the participants were working in the same countries where they had obtained their highest degree. Of the 16 who had emigrated, 11 had emigrated to the USA.

Table 28 describes the types of software developed by the participants. Participants were given four categories from which they could choose one or more. 72% of the participants chose more than one category.

Table 29 presents a breakdown of the participants by industry. It appears from this data that no one sector had an unduly large representation, although the ‘business’ category was probably somewhat under-represented.

Table 30 describes the sizes of the teams in which participants worked. We originally included a ‘very large’ category (Over 45 people working on an extremely large and complex system; typically more than 1.5 million lines of code); however, nobody in the final sample selected this category.

Fig. 5 and Table 31 describe how the participants say they spend their time. Clearly they believe they spend most of their time programming. The 'Maintenance' category found in Table 31 contains time that overlaps with some of the other categories (e.g. maintenance programming, or maintenance design).

Table 32 and Fig. 6 characterize the experience of participants in terms of the number of years since their highest degree, and the number of years working in the software industry. The data shows that there was a very wide range of participants.

The mean number of years of software experience was 12.4 (maximum 25; standard deviation 9.4). The mean number of years since receiving the highest degree was 10.2 (maximum 38; standard deviation 9.3).

9. Conclusions

The data from the 1998 survey of software practitioners provide concrete evidence that can be used for decision-making on the part of educators and trainers, as well as accreditation agencies. The following are some of the important observations and conclusions that can be drawn from the data.

Although there is far more to software development than programming, the participants in this survey put programming topics, specifically languages and data structures, at the very top. They did this both when asked what topics were the most important, and what they know most about. This might be a symptom of the tendency in much of software development to jump into coding without much requirements analysis or design. On the other hand it might just reflect the observation that whereas most software developers do some programming (and spend an average of 40% of their time at it), other tasks may be distributed to various members of the team. It is also possible that our data are biased towards programmers to some extent.

Behind programming in the amount-known and importance questions come a cluster of topics that have to do with software design, and then a variety of topics that have to do with other activities such as requirements, user interface design and testing. Supporting knowledge such as databases and operating systems also ranks quite highly.

Mathematics, especially calculus, is extensively taught in computing programs. Participants believe that following their education they had a more thorough grasp of calculus than practically anything else, except perhaps programming languages. On the other hand, relatively little mathematics turns out to be important for software engineers in practice and it tends to be forgotten. If we are to continue to teach the amount and type of mathematics, we must justify it by other means than by saying it is important to a software developers work: Our data show that it is normally not. Other justifications could include the need to communicate with other scientists and engineers, or the occasional need to work on problems that do indeed require it. At the very least, educators should look at the mathematics elements of computing programs and examine how they can be made more relevant – at least so the investment in mathematics education is not wasted by it being subsequently forgotten.

Many ‘skills’ or ‘soft’ topics were placed very high in this survey. These include giving presentations, technical writing, as well as ethics and professionalism. Educators should therefore consider increasing their emphasis on these topics.

At the time they graduate, there are considerable differences between engineering students and computing students in terms of the amount of computing knowledge. This is particularly the case for parsing, databases, systems programming and algorithmics topics. Since many engineering students find their way into computing jobs, this might be of concern: Engineering educators perhaps should boost the computing content in engineering curricula, while corporate trainers might target their engineering new-hires for additional computing training. At the same time, engineering graduates typically have greater knowledge than computing graduates in important areas such as digital signal processing, control systems, and electronics.

Both universities and corporate training departments could improve their offerings by adding more material about software design, architecture, user interfaces and project management as well as giving presentations to an audience and technical writing. Training departments ought additionally to focus on configuration management, while universities ought additionally to focus on object-oriented concepts as well as ethics and professionalism.

We would be happy to provide additional derivative data and analyses to readers on request, although of course we cannot release the individual responses.

Acknowledgements

This research was supported by the Consortium for Software Engineering Research (CSER). I thank all the participants and participating companies (who must remain anonymous). I also thank Anatol Kark at the National Research Council of Canada for assistance with data gathering, as well as the following researchers who provided much essential feedback and helped solicit participants: Nancy Mead at the Software Engineering Institute of Carnegie Mellon University; W. Michael McCracken at Georgia Tech; Laurie Werth at the University of Texas, Austin; Lawrence West at Columbia College; Lesley Beddie at Napier University, UK; Michael Lutz at the Rochester Institute of Technology, and Pearl Brereton at Keele University, UK. Finally, I thank K. Teresa Khidir for editing the paper.

References

Bourque, P., Dupuis, R., Abran, A., Moore, J. W., Tripp, L., Shyne, K., Pflug, B., Maya, M. and Tremblay, G. , 1999. Guide to the Software Engineering Body of Knowledge - A Straw Man Version", http://www.lrgl.uqam.ca/publications/pdf/365.pdf.

Hilburn, T.B., Bayert, D.J., Mengol, S., Oexman, D., 1998. Software engineering across computing Curricula, 3rd Annual Conference on Integrating Technology into Computer Science Education, 4.

IEEE, 1998. Draft Software Engineering Accreditation Criteria, *Computer*, April 1998, 73-77.

Lethbridge, T.C., 1998a. A survey of the relevance of computer science and software engineering education, 11th IEEE Conference on Software Engineering Education and Training, Atlanta, 56-66.

Lethbridge, T.C., 1998b. The relevance of software education: A survey and some recommendations" Annals of Software Engineering, 6, 91-110.

Lethbridge, T.C, 1999a. Online Excel spreadsheet with the data for the survey. http://www.site.uottawa.ca/~tcl/edrel/EdrelData1998.xls.

Lethbridge, T.C., 1999b. What knowledge is important to a software engineer?" Submitted January 1999, IEEE Software.

Lethbridge, T.C., 1999c. The relevance of education to software practitioners: Data from the 1998 survey, Computer Science Technical Report TR-99-06, University of Ottawa, Canada. http://www.site.uottawa.ca/~tcl/edrel/EdrelTechReport/

Timothy C. Lethbridge is an assistant professor of at the University of Ottawa. He teaches courses in the new software engineering program which he helped establish. His research interests include user interfaces, software engineering tools, software engineering education and knowledge representation. He is one of the principal investigators in the Consortium for Software Engineering Research (CSER) which is studying how to better understand and evolve large legacy software systems. In this capacity he works with Mitel and other Canadian corporations. Lethbridge received his Ph.D. from the University of Ottawa, and his M.Sc. and B.Sc. from the University of New Brunswick, Canada. He is a member of the IEEE, the IEEE Computer Society and the ACM.

Question 1. How much did you learn about this in your formal education (e.g. **University or College**)? **0**=Learned nothing at all **1**=Became vaguely familiar **2**=Leaned the basics **3**=Became functional (moderate working knowledge) **4**=Learned a lot **5**=Learned in depth; became expert (Learned almost everything)	*Question 2.* What is your **current knowledge** about this, considering what you have learned on the job as well as forgotten? **0**=Know nothing **1**=Am vaguely familiar **2**=Know the basics **3**=Am functional (moderate working knowledge) **4**=Know a lot **5**=Know in depth / am expert (Know almost everything)
Question 3. How useful have the details of this **specific material** been to you in your career as a software developer or software manager? Please leave blank if you know little about the material. **0**=Completely Useless **1**=Almost never useful **2**=Occasionally useful **3**=Moderately useful, but perhaps only in certain activities **4**=Very useful **5**=Essential	*Question 4.* How much **influence** has learning the material had **on your thinking** (i.e. your approach to problems and your general intellectual maturity), **whether or not you have directly used the details** of the material? Please consider influence on **both your career and other aspects of your life**. Please leave blank if you know little about the material. **0**=No influence at all **1**=Almost no influence **2**=Occasional influence **3**=Moderate influence in some activities **4**=Significant influence in many activities **5**=Profound influence on almost everything I do

Fig. 1. The four questions that were asked about each of the 75 topics in the survey. Bold or underlined words and phrases were designed to ensure that participants would understand key aspects of each question even if they read it quickly.

Software Engineering Process
- Analysis and Design Methods
- Configuration and Release Management
- Formal Specification Methods
- Maintenance, Reengineering and Reverse Engineering
- Performance Measurement and Analysis
- Process standards CMM™[2] / ISO 9000 etc
- Project Management
- Requirements Gathering and Analysis
- Software Cost Estimation
- Software Metrics
- Software Reliability and Fault Tolerance
- Testing, Verification and Quality Assurance

Software Design Core
- Data Structures
- Design of Algorithms
- Human Computer Interaction / User Interfaces
- Object Oriented Concepts and Technology
- Software Architecture
- Software Design and Patterns
- Specific Programming Languages

Software Subsystem Design
- Databases
- File Management
- Parsing and Compiler Design

Other Software
- Artificial Intelligence
- Computational Methods for Numerical Problems
- Computer Graphics
- Information Retrieval
- Pattern Recognition and Image Processing
- Security and Cryptography
- Simulation

Computer Engineering Software Topics
- Data Transmission and Networks
- Operating Systems
- Parallel and Distributed Processing
- Real-Time System Design
- Systems Programming

Computer Science Theoretical Topics
- Computational Complexity and Algorithm Analysis
- Programming Language Theory

Mathematical Topics Widely Used in Computer Science
- Automata theory
- Control Theory
- Formal Languages
- Graph Theory
- Information Theory
- Predicate Logic
- Queuing Theory
- Set Theory

Other Mathematics
- Combinatorics
- Differential and Integral Calculus
- Differential Equations
- Laplace and Fourier Transforms
- Linear Algebra and Matrices
- Probability and Statistics

Computer Engineering Hardware Topics
- Computer System Architecture
- Digital Electronics and Digital Logic
- Microprocessor Architecture
- Network Architecture and Data Transmission
- Telephony and Telecommunications

Other Hardware
- Analog Electronics
- Data Acquisition
- Digital Signal Processing
- Robotics
- VLSI

Basic Science
- Chemistry
- Physics

Business
- Accounting
- Economics
- Entrepreneurship
- Management
- Marketing

Humanities & Skills
- Ethics and Professionalism
- Giving Presentations to an Audience
- Leadership
- Negotiation
- Philosophy
- Psychology
- Second Language Other than English as Second Language
- Technical Writing

Fig. 2: The 75 topics in the survey, arranged in categories; alphabetically within each category.

---

[2] CMM is a trademark of Carnegie Mellon University.

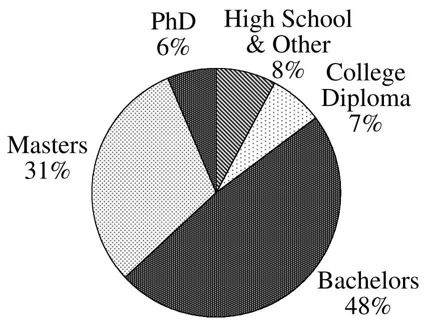

Fig. 3. Highest educational level reached by the participants (170 responses).

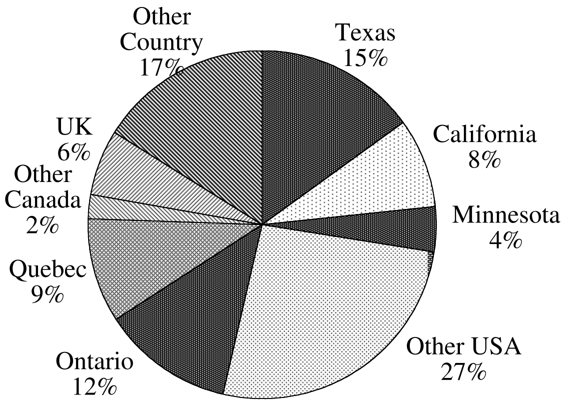

Fig. 4. Work locations of the participants. Participants work in a total of 24 countries.

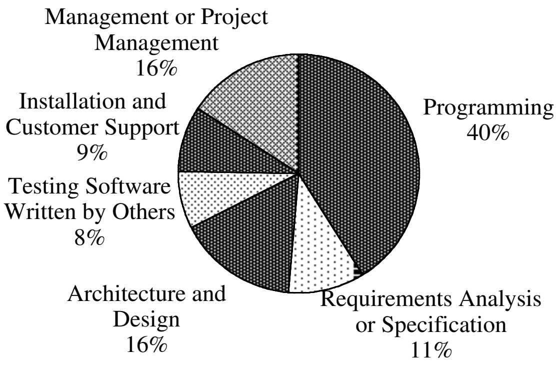

Fig. 5. How the average participant distributes his or her time. This data should be considered highly subjective, and accounts for no idle time, meetings, phone calls etc.

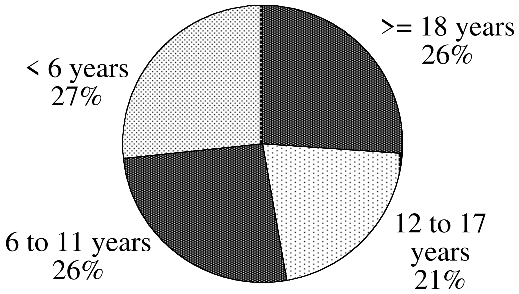

Fig. 6. Experience of participants developing software: the number of years working in the software industry.

Table 1
Topics about which participants had learned most in their formal education: The topics ranked 1 to 25 according to the means of question 1. The text for question 1 is found in Fig. 1.

Rank	Topic	Mean of Q1	Std. dev.	% rating 4 or 5	% rating > 1
1	Specific Programming Languages	3.2	1.36	44%	89%
2	Differential and Integral Calculus	3.2	1.41	52%	89%
3	Linear Algebra and Matrices	3.1	1.43	45%	85%
4	Probability and Statistics	2.9	1.25	34%	87%
5	Data Structures	2.9	1.55	42%	78%
6	Physics	2.7	1.42	31%	82%
7	Differential Equations	2.7	1.44	30%	79%
8	Set Theory	2.5	1.52	29%	75%
9	Design of Algorithms	2.3	1.54	26%	68%
10	Operating Systems	2.3	1.49	20%	71%
11	Computer System Architecture	2.2	1.65	28%	63%
12	Programming Language Theory	2.2	1.50	19%	68%
13	Predicate Logic	2.2	1.64	25%	65%
14	Chemistry	2.2	1.40	14%	70%
15	Computational Methods for Numeric Problems	2.2	1.50	18%	67%
16	Formal Languages	2.2	1.66	29%	63%
17	Digital Electronics & Digital Logic	2.1	1.70	25%	59%
18	Computational Complexity & Algorithm Analysis	2.1	1.65	25%	59%
19	Software Architecture	2.0	1.54	17%	59%
20	Microprocessor Architecture	1.9	1.59	19%	56%
21	File Management	1.9	1.48	16%	57%
22	Databases	1.9	1.58	18%	56%
23	Graph Theory	1.9	1.55	20%	56%
24	Analysis and Design Methods	1.9	1.52	13%	57%
25	Second Language Other than English	1.8	1.61	15%	57%

Table 2
Topics about which participants had learned least in their formal education: The topics ranked 51 to 75 according to the means of question 1. The text for question 1 is found in Fig. 1.

Rank	Topic	Mean of Q1	Std. dev.	% rating 4 or 5	% rating > 1
51	Simulation	1.2	1.43	7%	38%
52	Parallel and Distributed Processing	1.2	1.44	11%	36%
53	Project Management	1.2	1.48	8%	37%
54	Management	1.2	1.36	6%	37%
55	Human Computer Interaction / User Interfaces	1.1	1.35	4%	35%
56	Psychology	1.1	1.27	3%	36%
57	Software Reliability & Fault Tolerance	1.0	1.28	5%	29%
58	Digital Signal Processing	1.0	1.42	10%	31%
59	Telephony and Telecommunications	1.0	1.33	6%	31%
60	Accounting	1.0	1.33	6%	33%
61	Real-Time System Design	1.0	1.34	6%	30%
62	Pattern Recognition and Image Processing	0.9	1.38	8%	26%
63	Leadership	0.8	1.22	3%	26%
64	Software Metrics	0.8	1.21	5%	24%
65	Security and Cryptography	0.8	1.29	4%	26%
66	Data Acquisition	0.8	1.17	2%	23%
67	Maintenance, Reengineering & Reverse Engineering	0.8	1.11	3%	21%
68	Marketing	0.7	1.13	4%	20%
69	VLSI	0.6	1.14	4%	17%
70	Robotics	0.6	1.11	5%	19%
71	Software Cost Estimation	0.6	1.05	3%	18%
72	Configuration and Release Management	0.5	0.96	2%	15%
73	Entrepreneurship	0.5	1.07	3%	13%
74	Process Standards such as CMM and ISO 9000	0.5	0.93	1%	14%
75	Negotiation	0.5	0.94	1%	14%

Table 3
Topics which those graduating in the last four years (junior participants) learned more thoroughly in their formal education as opposed to those graduating 12 or more years ago (expert participants).

Rank	Topic	% that juniors learned more	Junior learning	Expert learning
1	Object Oriented Concepts & Technology	256%	2.4	0.7
2	Parallel and Distributed Processing	161%	2.0	0.8
3	Software Reliability & Fault Tolerance	154%	1.5	0.6
4	Software Cost Estimation	154%	1.0	0.4
5	Computer Graphics	141%	2.0	0.8
6	Human Computer Interaction / User Interfaces	141%	1.6	0.7
7	Software Metrics	135%	1.3	0.5
8	Pattern Recognition and Image Processing	131%	1.4	0.6
9	Digital Signal Processing	121%	1.4	0.6
10	Configuration and Release Management	108%	0.8	0.4
11	Artificial Intelligence	100%	1.7	0.8

Table 4
Topics where engineering graduates learned considerably more than computing graduates in their formal education. This includes only cases where significant learning occurred (Q1 > 2).

Rank	Topic	% Engineers learned more	Engineering learning	Computing learning
1	Digital Signal Processing	142%	2.1	0.9
2	Analog Electronics	119%	2.5	1.1
3	Control Theory	98%	2.3	1.2
4	Laplace and Fourier Transforms	89%	3.1	1.6
5	Physics	32%	3.5	2.6
6	Differential Equations	32%	3.4	2.6
7	Digital Electronics & Digital Logic	27%	3.0	2.3

Table 5
Topics where computing graduates learned considerably *more* than engineering graduates in their formal education. This includes only cases where significant learning occurred (Q1 > 2).

Rank	Topic	% that CS/SE learned more	Computing learning	Engineering learning
1	Parsing and Compiler Design	90%	2.4	1.3
2	File Management	80%	2.5	1.4
3	Programming Language Theory	70%	2.9	1.7
4	Databases	69%	2.5	1.5
5	Systems Programming	65%	2.2	1.4
6	Software Architecture	61%	2.6	1.6
7	Operating Systems	49%	3.0	2.0
8	Automata theory	44%	2.1	1.5
9	Design of Algorithms	43%	2.9	2.0
10	Computational Complexity & Alg. Analysis	38%	2.5	1.8

Table 6
Topics about which participants currently know most. The topics ranked 1 to 25 according to the means of question 2. The text for question 2 is found in Fig. 1.

Rank	Topic	Mean of Q2	Std. dev.	% rating 4 or 5	% rating > 1
1	Specific Programming Languages	4.1	1.03	78%	97%
2	Data Structures	3.7	1.12	65%	96%
3	Operating Systems	3.4	1.11	46%	96%
4	Software Design and Patterns	3.4	1.30	52%	90%
5	Software Architecture	3.3	1.36	49%	87%
6	Giving Presentations to an Audience	3.3	1.34	50%	89%
7	Databases	3.2	1.22	45%	89%
8	Object Oriented Concepts & Technology	3.2	1.41	45%	85%
9	Testing, Verification & Quality Assurance	3.2	1.18	41%	91%
10	Analysis and Design Methods	3.2	1.33	50%	85%
11	Requirements Gathering & Analysis	3.1	1.37	48%	87%
12	Project Management	3.1	1.25	39%	91%
13	File Management	3.1	1.46	45%	84%
14	Ethics and Professionalism	3.0	1.47	38%	84%
15	Design of Algorithms	3.0	1.46	44%	83%
16	Technical Writing	3.0	1.43	41%	85%
17	Data Transmission and Networks	3.0	1.29	39%	87%
18	Configuration and Release Management	2.8	1.50	34%	79%
19	Human Computer Interaction / User Interfaces	2.8	1.39	30%	81%
20	Programming Language Theory	2.7	1.48	33%	77%
21	Computer System Architecture	2.7	1.45	31%	79%
22	Performance Measurement & Analysis	2.6	1.43	28%	79%
23	Maintenance, Reengineering and Rev. Engg.	2.6	1.42	33%	79%
24	Network Architecture & Data Transmission	2.6	1.41	27%	77%
25	Systems Programming	2.5	1.67	36%	68%

Table 7
Topics about which participants currently know least. The topics ranked 51 to 75 according to means of question 2. The text for question 2 is found in Fig. 1.

Rank	Topic	Mean of Q2	Std. dev.	% rating 4 or 5	% rating > 1
51	Security and Cryptography	2.0	1.45	12%	62%
52	Predicate Logic	1.9	1.55	18%	55%
53	Telephony and Telecommunications	1.9	1.53	14%	57%
54	Second Language Other than English	1.8	1.56	14%	54%
55	Philosophy	1.7	1.42	10%	52%
56	Graph Theory	1.7	1.43	11%	51%
57	Psychology	1.6	1.38	9%	50%
58	Entrepreneurship	1.6	1.53	13%	50%
59	Information Theory	1.6	1.48	13%	49%
60	Queuing Theory	1.6	1.43	9%	47%
61	Differential Equations	1.6	1.30	8%	47%
62	Accounting	1.5	1.34	7%	49%
63	Data Acquisition	1.5	1.60	13%	42%
64	Automata theory	1.5	1.56	13%	44%
65	Chemistry	1.4	1.18	4%	47%
66	Marketing	1.4	1.42	9%	42%
67	Combinatorics	1.4	1.42	8%	43%
68	Artificial Intelligence	1.4	1.29	6%	43%
69	Digital Signal Processing	1.3	1.44	10%	36%
70	Pattern Recognition and Image Processing	1.3	1.41	7%	36%
71	Laplace and Fourier Transforms	1.2	1.36	6%	37%
72	Analog Electronics	1.2	1.39	7%	36%
73	Control Theory	1.1	1.32	5%	34%
74	Robotics	0.9	1.19	4%	27%
75	VLSI	0.7	1.19	4%	20%

Table 8
Topics that managers know better than the participants at large. This includes only cases where knowledge of managers is significant (Q2 > 2).

Rank	Topic	% that managers know more	Knowledge of managers	Knowledge of participants at large
1	Negotiation	50%	3.0	2.0
2	Management	48%	3.6	2.5
3	Process Standards CMM / ISO 9000	40%	2.9	2.1
4	Psychology	38%	2.2	1.6
5	Software Metrics	37%	2.9	2.1
6	Leadership	35%	3.4	2.5
7	Project Management	32%	4.1	3.1
8	Software Cost Estimation	31%	2.8	2.2
9	Accounting	29%	2.0	1.5
10	Entrepreneurship	27%	2.1	1.6
11	Ethics and Professionalism	21%	3.6	3.0
12	Requirements Gathering & Analysis	20%	3.8	3.1

Table 9
Topics that programmers know more about than participants at large. This includes only cases where knowledge or programmers is significant (Q2 > 2).

Rank	Topic	% that programmers know more	Knowledge of programmers	Knowledge of participants at large
1	Design of Algorithms	10%	3.3	3.0
2	Object Oriented Concepts & Technology	9%	3.5	3.2
3	Specific Programming Languages	8%	4.4	4.1
4	Parsing and Compiler Design	8%	2.1	2.0

Table 10
Topics that programmers know less about than the participants at large. This includes only cases where knowledge of participants at large is significant (Q2 > 2).

Rank	Topic	% that programmers know less	Knowledge of programmers	Knowledge of participants at large
1	Process Standards CMM / ISO 9000	-23%	1.6	2.1
2	Management	-22%	1.9	2.5
3	Leadership	-19%	2.1	2.5
4	Negotiation	-17%	1.7	2.0
5	Software Metrics	-16%	1.8	2.1
6	Software Cost Estimation	-15%	1.8	2.2
7	Project Management	-13%	2.7	3.1
8	Giving Presentations to Audience	-12%	2.9	3.3
9	Requirements Gathering & Analysis	-10%	2.8	3.1

Table 11
The most important 25 topics – based on the average of both importance of details and influence.

Rank	Topic	Overall Importance	Details (Q3)	Influence (Q4)
1	Specific Programming Languages	3.8	4.1	3.5
2	Data Structures	3.6	3.7	3.4
3	Software Design and Patterns	3.5	3.6	3.5
4	Software Architecture	3.4	3.5	3.3
5	Requirements Gathering & Analysis	3.4	3.5	3.3
6	Human Computer Interaction / User Interfaces	3.3	3.3	3.3
7	Object Oriented Concepts & Technology	3.3	3.3	3.3
8	Ethics and Professionalism	3.3	3.2	3.4
9	Analysis and Design Methods	3.3	3.3	3.3
10	Giving Presentations to an Audience	3.3	3.5	3.1
11	Project Management	3.3	3.4	3.2
12	Testing, Verification & Quality Assurance	3.2	3.3	3.1
13	Design of Algorithms	3.2	3.3	3.1
14	Technical Writing	3.1	3.4	2.9
15	Operating Systems	3.1	3.3	3.0
16	Databases	3.1	3.3	2.8
17	Leadership	3.0	3.1	3.0
18	Configuration and Release Management	3.0	3.3	2.8
19	Data Transmission and Networks	3.0	3.1	2.8
20	Management	2.9	2.9	2.9
21	File Management	2.8	3.2	2.4
22	Software Reliability & Fault Tolerance	2.8	2.9	2.7
23	Systems Programming	2.8	2.9	2.7
24	Network Architecture & Data Transmission	2.8	2.8	2.7
25	Negotiation	2.8	2.9	2.6

Table 12
The middle 25 topics in terms of importance – based on the average of both importance of details and influence. Continuation of Table 11.

Rank	Topic	Overall Importance	Details (Q3)	Influence (Q4)
26	Performance Measurement & Analysis	2.7	2.8	2.6
27	Maintenance, Reengineering and Rev. Engg.	2.7	2.8	2.6
28	Programming Language Theory	2.7	2.7	2.7
29	Computer System Architecture	2.7	2.7	2.6
30	Computational Complexity & Algor. Analysis	2.6	2.6	2.6
31	Probability and Statistics	2.6	2.4	2.7
32	Software Cost Estimation	2.6	2.7	2.4
33	Real-Time System Design	2.5	2.6	2.5
34	Information Retrieval	2.5	2.7	2.3
35	Software Metrics	2.5	2.6	2.4
36	Formal Languages	2.4	2.4	2.4
37	Formal Specification Methods	2.4	2.4	2.4
38	Process Standards CMM / ISO 9000	2.4	2.4	2.4
39	Predicate Logic	2.4	2.2	2.5
40	Entrepreneurship	2.4	2.2	2.5
41	Simulation	2.3	2.4	2.3
42	Security and Cryptography	2.3	2.2	2.4
43	Telephony and Telecommunications	2.3	2.3	2.3
44	Parsing and Compiler Design	2.3	2.3	2.3
45	Parallel and Distributed Processing	2.3	2.3	2.3
46	Microprocessor Architecture	2.2	2.2	2.3
47	Digital Electronics & Digital Logic	2.2	2.1	2.3
48	Set Theory	2.2	2.2	2.2
49	Automata theory	2.1	2.0	2.3
50	Data Acquisition	2.1	2.2	2.0

Table 13
The least important 25 topics – based on the average of both importance of details and influence. Continuation of Table 12.

Rank	Topic	Overall Importance	Details (Q3)	Influence (Q4)
51	Marketing	2.1	2.0	2.3
52	Computational Methods for Numeric Problems	2.1	2.2	2.0
53	Psychology	2.1	2.0	2.2
54	Accounting	2.1	2.1	2.1
55	Economics	2.1	1.8	2.3
56	Linear Algebra and Matrices	2.0	2.0	2.1
57	Philosophy	2.0	1.5	2.5
58	Second Language Other than English	2.0	1.9	2.1
59	Physics	2.0	1.6	2.3
60	Information Theory	2.0	1.9	2.0
61	Graph Theory	2.0	2.0	1.9
62	Queuing Theory	1.9	2.1	1.8
63	Computer Graphics	1.9	1.9	1.8
64	Digital Signal Processing	1.8	1.7	1.8
65	Control Theory	1.7	1.6	1.8
66	Pattern Recognition and Image Processing	1.6	1.6	1.7
67	Differential and Integral Calculus	1.6	1.3	1.9
68	Combinatorics	1.6	1.5	1.6
69	Artificial Intelligence	1.5	1.3	1.8
70	Analog Electronics	1.5	1.4	1.7
71	Laplace and Fourier Transforms	1.3	1.3	1.4
72	Differential Equations	1.3	1.1	1.4
73	Chemistry	1.3	0.9	1.6
74	Robotics	1.3	1.2	1.4
75	VLSI	1.2	1.1	1.3

Table 14
Topics whose details are perceived to be most important. Mean responses to question 3. The text for question 3 is found in Fig. 1. See Table 15 for the bottom scores.

Rank	Topic	Mean of Q3	Std. dev.	% rating 4 or 5	% rating > 1
1	Specific Programming Languages	4.1	1.19	75%	95%
2	Data Structures	3.7	1.22	60%	96%
3	Software Design and Patterns	3.6	1.34	58%	91%
4	Software Architecture	3.5	1.29	53%	92%
5	Requirements Gathering & Analysis	3.5	1.22	53%	93%
6	Giving Presentations to an Audience	3.5	1.45	57%	88%
7	Technical Writing	3.4	1.41	53%	90%
8	Project Management	3.4	1.32	45%	92%
9	Analysis and Design Methods	3.3	1.38	48%	87%
10	Object Oriented Concepts & Technology	3.3	1.45	48%	88%
11	Operating Systems	3.3	1.31	44%	89%
12	Human Computer Interaction / User Interfaces	3.3	1.23	44%	91%
13	Testing, Verification & Quality Assurance	3.3	1.30	44%	90%
14	Databases	3.3	1.51	48%	87%
15	Configuration and Release Management	3.3	1.35	48%	88%
16	Design of Algorithms	3.3	1.30	47%	90%
17	File Management	3.2	1.34	42%	87%
18	Ethics and Professionalism	3.2	1.50	44%	85%
19	Data Transmission and Networks	3.1	1.30	44%	88%
20	Leadership	3.1	1.46	41%	84%

Table 15
Topics whose details are perceived to be least important: Mean responses to question 3 (see Table 14 for the top scores).

Rank	Topic	Mean of Q3	Std. dev.	% rating 0 or 1
1	Chemistry	0.9	1.15	71%
2	Differential Equations	1.1	1.29	71%
3	VLSI	1.1	1.43	70%
4	Robotics	1.2	1.28	68%
5	Laplace and Fourier Transforms	1.3	1.46	65%
6	Artificial Intelligence	1.3	1.27	61%
7	Differential and Integral Calculus	1.3	1.33	66%
8	Analog Electronics	1.4	1.34	55%
9	Philosophy	1.5	1.38	53%
10	Combinatorics	1.5	1.36	47%

Table 16
Topics that have had the most influence. Mean responses to question 4.

Rank	Topic	Mean of Q4	Std. dev.	% rating 4 or 5	% rating > 1
1	Specific Programming Languages	3.5	1.48	55%	86%
2	Software Design and Patterns	3.5	1.48	57%	86%
3	Ethics and Professionalism	3.4	1.49	56%	86%
4	Data Structures	3.4	1.30	52%	91%
5	Human Computer Interact. / User Interfaces	3.3	1.32	51%	90%
6	Requirements Gathering & Analysis	3.3	1.34	49%	88%
7	Object Oriented Concepts & Technology	3.3	1.50	49%	86%
8	Software Architecture	3.3	1.46	49%	87%
9	Analysis and Design Methods	3.3	1.42	45%	87%
10	Project Management	3.2	1.47	44%	84%
11	Testing, Verification & Quality Assurance	3.1	1.40	42%	85%

Table 17
Topics which people needed to learn most about on the job when they knew almost nothing to start with – indirectly indicating importance.

Rank	Topic	Forced Learning (Q2 - Q1)	Std. Dev.	n
1	Specific Programming Languages	3.1	1.45	20
2	Data Structures	2.7	1.61	38
3	Operating Systems	2.7	1.24	52
4	Testing, Verification & Quality Assurance	2.7	1.23	106
5	Project Management	2.7	1.35	111
6	Object Oriented Concepts & Technology	2.6	1.54	103
7	Databases	2.6	1.30	79
8	Configuration and Release Management	2.6	1.58	146
9	Requirements Gathering & Analysis	2.5	1.47	100
10	Ethics and Professionalism	2.5	1.65	104
11	Software Design and Patterns	2.5	1.43	83
12	Giving Presentations to an Audience	2.4	1.42	88

Table 18
Topics that real-time developers consider more important than other developers. This considers only topics for which overall importance was greater than 2.5.

Rank	Topic	% increased importance for real-time	Importance for real time developers	Importance for participants at large
1	Technical Writing	12%	3.5	3.1
2	Giving Presentations to an Audience	6%	3.5	3.3
3	Real-Time System Design	29%	3.3	2.5
4	Software Reliability & Fault Tolerance	10%	3.1	2.8
5	Systems Programming	5%	2.9	2.8
6	Negotiation	6%	2.9	2.8
7	Computer System Architecture	9%	2.9	2.7
8	Digital Electronics & Digital Logic	24%	2.8	2.2
9	Process Standards. CMM / ISO 9000	9%	2.6	2.4
10	Simulation	11%	2.6	2.3

Table 19
Topics that MIS developers consider more important than developers at large. This considers only topics for which overall importance was greater than 2.5.

Rank	Topic	% increased importance for MIS	Importance for MIS	Importance for participants at large
1	Databases	10%	3.4	3.1
2	Information Retrieval	6%	2.6	2.5
3	Entrepreneurship	12%	2.6	2.4

Table 20
Topics that managers consider more important than developers at large. This considers only topics for which overall importance was greater than 3.

Rank	Topic	% increased importance for managers	Importance for managers	Importance for participants at large
1	Project Management	25%	4.1	3.3
2	Requirements Gathering & Analysis	14%	3.9	3.4
3	Giving Presentations to an Audience	14%	3.7	3.3
4	Management	26%	3.7	2.9
5	Ethics and Professionalism	11%	3.7	3.3
6	Analysis and Design Methods	8%	3.6	3.3
7	Software Architecture	5%	3.6	3.4
8	Leadership	17%	3.5	3.0
9	Testing, Verification & Qual. Assurance	7%	3.4	3.2
10	Technical Writing	9%	3.4	3.1
11	Negotiation	20%	3.3	2.8
12	Network Architecture & Data Transmission	17%	3.2	2.8
13	Data Transmission and Networks	7%	3.2	3.0
14	Software Cost Estimation	25%	3.2	2.6
15	Process Standards CMM / ISO 9000	32%	3.2	2.4
16	Software Metrics	24%	3.1	2.5

Table 21
Topics for which on-the-job learning was greatest: Large positive differences between questions 2 and 1.

Rank	Topic	On-the-job learning (Q2-Q1)	Q1: Knowledge after education	Q2: Knowledge now	% where Q2 -Q1 > 2
1	Configuration and Release Management	2.3	0.5	2.8	54%
2	Project Management	1.9	1.2	3.1	41%
3	Testing, Verification & Qual. Assurance	1.9	1.3	3.2	39%
4	Maintenance, Reengineering and Rev. Engg.	1.9	0.8	2.6	34%
5	Object Oriented Concepts & Technology	1.8	1.4	3.2	36%
6	Requirements Gathering & Analysis	1.8	1.4	3.1	36%
7	Ethics and Professionalism	1.8	1.2	3.0	35%
8	Leadership	1.7	0.8	2.5	33%
9	Human Computer Interact. / User Interfaces	1.7	1.1	2.8	35%
10	Giving Presentations to Audience	1.6	1.6	3.3	31%
11	Process Standards CMM / ISO 9000	1.6	0.5	2.1	32%
12	Software Cost Estimation	1.6	0.6	2.2	32%

Table 22
Topics for which there was the most forgetting of material following completion of education. Negative differences between questions 2 and 1.

Rank	Topic	Forgetting since education (Q2-Q1)	Q1: Knowledge after education	Q2: Knowledge Now	% where Q2 -Q1 <= -2
1	Differential Equations	-1.2	2.7	1.6	38%
2	Differential and Integral Calculus	-1.1	3.2	2.1	38%
3	Linear Algebra and Matrices	-0.8	3.1	2.3	24%
4	Chemistry	-0.8	2.2	1.4	21%
5	Physics	-0.7	2.7	2.1	21%
6	Laplace and Fourier Transforms	-0.6	1.8	1.2	21%
7	Probability and Statistics	-0.5	2.9	2.4	17%
8	Combinatorics	-0.4	1.7	1.4	12%
9	Set Theory	-0.4	2.5	2.1	12%
10	Predicate Logic	-0.3	2.2	1.9	12%
11	Graph Theory	-0.2	1.9	1.7	7%
12	Control Theory	-0.2	1.2	1.1	7%
13	Analog Electronics	-0.2	1.3	1.2	9%

Table 23
Topics for which there may be the most need for training. Important (Q3>3) topics where current knowledge lags most behind the importance of details (ordered by importance).

Rank	Topic	% Lag	Current knowledge (Q2)	Importance of details (Q3)
1	Software Design and Patterns	6%	3.4	3.6
2	Software Architecture	7%	3.3	3.5
3	Requirements Gathering & Analysis	10%	3.1	3.5
4	Giving Presentations to an Audience	6%	3.3	3.5
5	Technical Writing	13%	3.0	3.4
6	Project Management	8%	3.1	3.4
7	Analysis and Design Methods	5%	3.2	3.3
8	Human Computer Interaction / User Interfaces	17%	2.8	3.3
9	Configuration and Release Management	13%	2.8	3.3
10	Design of Algorithms	9%	3.0	3.3
11	Ethics and Professionalism	5%	3.0	3.2
12	Data Transmission and Networks	6%	3.0	3.1
13	Leadership	17%	2.5	3.1

Table 24
Topics for which there may be the most need for improvement in university courses. Important (>3) topics where learning in educational programs lagged most behind overall importance (ordered by importance).

Rank	Topic	% Lag	Knowledge after education (Q1)	Overall importance (Q3+Q4)/2
1	Software Design and Patterns	48%	1.8	3.5
2	Requirements Gathering & Analysis	60%	1.4	3.4
3	Software Architecture	43%	2.0	3.4
4	Human Computer Interaction / User Interfaces	67%	1.1	3.3
5	Object Oriented Concepts & Technology	58%	1.4	3.3
6	Ethics and Professionalism	63%	1.2	3.3
7	Analysis and Design Methods	44%	1.9	3.3
8	Giving Presentations to an Audience	52%	1.6	3.3
9	Project Management	63%	1.2	3.3
10	Testing, Verification & Quality Assurance	59%	1.3	3.2
11	Technical Writing	48%	1.6	3.1

Table 25
Topics taught relatively more than their importance might warrant.

Rank	Topic	% Excess	Knowledge after education (Q1)	Overall importance(Q3+Q4)/2
1	Differential Equations	115%	2.7	1.3
2	Differential and Integral Calculus	97%	3.2	1.6
3	Chemistry	73%	2.2	1.3
4	Linear Algebra and Matrices	50%	3.1	2.0
5	Laplace and Fourier Transforms	39%	1.8	1.3
6	Physics	37%	2.7	2.0
7	Set Theory	15%	2.5	2.2
8	Probability and Statistics	13%	2.9	2.6
9	Combinatorics	11%	1.7	1.6
10	Computational Methods for Numeric Problems	3%	2.2	2.1

Table 26
Degree subjects of the 169 participants (out of 181) who responded to this question. Note that many participants had more than one degree.

Type of education	Number of participants	Percentage
Computing degree	106	62.0%
Computer Science	88	51.5%
Software Engineering (mostly masters degrees, but also somewhat inflated by participants who really meant computer engineering)	22	12.9%
Computing Information Systems	10	5.8%
Engineering Degree	49	28.7%
Computer Engineering	12	7.0%
Electrical Engineering	23	13.5%
Other Engineering	17	9.9%
Math / Science Degree	37	21.6%
Math	29	17.0%
Other Science	12	7.0%
No Scientific Degree	33	19.3%
Business Degree	13	7.6%
Other including no degree	22	12.9%

Table 27
Country where each of the 174 (out of 181) responding participants had obtained their highest degree.

Country of highest degree	# participants	Percentage
USA	92	52.9%
Canada	40	23.0%
Non North America (20 countries)	42	24.1%
UK	13	7.5%
Israel	4	2.3%
India	3	1.7%
Sweden	2	1.1%
Norway	2	1.1%
Netherlands	2	1.1%
Italy	2	1.1%
Germany	2	1.1%
1 each: Yugoslavia, Ukraine, South Africa, Portugal, Poland, France, Finland, Denmark, Chile, Bulgaria, Belgium, Australia	12	6.9%

Table 28
The type of software developed by each of the 169 participants who answered this question. Participants were given the list of four categories and were asked to check all that apply.

Type of software developed	# participants	Percentage
Management Information Systems (MIS) or other software for running the business (e.g. accounting, inventory etc.) that is being developed or tailored largely for in-house use	78	45.1%
MIS Software ONLY	29	16.8%
Only NON MIS software	95	54.9%
Consumer or mass-market software (typically sold on the open-market in shrink-wrapped packages)	20	11.6%
Consumer Software ONLY	3	1.7%
Application software produced for specialized markets that does not fit into the MIS, Consumer or Real Time categories	95	54.9%
Specialized Application software ONLY	27	15.6%
Real-time, embedded, systems or telecommunications software (in general, software that is developed as part of a larger system)	82	47.4%
Real Time software ONLY	33	19.1%
Only NON Real Time software	91	52.6%

Table 29
Industries in which the participants worked. Note that some categories (e.g. 'telecommunications software', and 'military') have been counted in more than one higher-level category.

Industry	# participants	Percentage
Software Industry	74	42.0%
Telecommunications or networking software development	19	10.8%
Development of other application software for specialized markets	20	11.4%
Software consulting	13	7.4%
Financial software development	10	5.7%
Engineering or scientific software development	8	4.5%
Consumer software development	4	2.3%
Industries Whose Main Focus is Other than Software	102	58.0%
Public Sector	36	20.5%
Military (also included in military / aerospace below)	16	9.1%
Other government including agencies, law enforcement, local	20	11.4%
Military / Aerospace	27	15.3%
Aeronautics, space, defense contracting	13	7.4%
Engineering (includes also military/aerospace and engineering software)	53	30.1%
Telecommunications or networking equipment manufacturing	2	1.1%
Manufacturing other computer hardware, computer engineering	10	5.7%
Other engineering	6	3.4%
Business (includes also financial software)	35	19.9%
Banking, Finance, Insurance, Financial Services or Consulting etc.	13	7.4%
Health Care	5	2.8%
Retailing, Marketing, Distributing	1	0.6%
Broadcasting, Publishing, Media	2	1.1%
Law, Legal Services	1	0.6%
Other Service Industry	3	1.7%
Telecommunications (includes also telecom software and telecom equipment manufacturing)	24	13.6%
Telecommunication or Networking Service	3	1.7%
Education	8	4.5%
Food Processing	1	0.6%
Non-Profit	1	0.6%

Table 30
The sizes of teams in which participants worked.

Team size	Number of participants	Percent
Small team: 1-7 people working on a small system (typically less than 10000 lines of code)	78	46.4%
Medium team: 5-20 people working on a medium sized system(typically 8000 to 150000 lines of code)	68	40.5%
Large team: 15-60 people working on a large and complex system (typically between 100000 and 2 million lines of code)	22	13.1%

Table 31
Details of percentage of time spent by participants

	No time spent on this	0-5% of time	5-10% of time	10-25% of time	25-50% of time	50-75% of time	75-100% of time
Programming: Working with Source Code (writing code, understanding code etc.)	7%	9%	10%	14%	35%	15%	9%
Requirements Analysis or Specification	5%	22%	37%	27%	7%	1%	1%
Software Architecture and Design	8%	17%	24%	23%	22%	3%	2%
Testing Software Written by Others	21%	34%	25%	11%	6%	1%	1%
Installation and Customer Support	28%	29%	18%	15%	5%	5%	1%
Maintenance: Understanding or Modifying Software or Documents Written by Others	7%	18%	24%	21%	14%	11%	5%
Management or Project Management	13%	27%	18%	20%	8%	10%	4%

Table 32
Graduation decades of participants in the survey.

Graduation decade	Number of participants	Percent
50s graduates	0	0.0%
60s graduates	7	4.3%
70s graduates	29	17.8%
80s graduates	38	23.3%
90s graduates	89	54.6%