The Relevance of Education to Software Practitioners: Data from the 1998 Survey

University of Ottawa Computer Science Technical Report TR-99-05 (Rev. 2)

July 1999

Timothy C. Lethbridge
School of Information Technology and Engineering
University of Ottawa
Ottawa, Canada, K1N 6N5
tcl@site.uottawa.ca
http://www.site.uottawa.ca/~tcl

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 now about it 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.

Table of Contents

1. Introduction

This report presents the complete 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.

Our studies of the work practices of software practitioners (Lethbridge and Singer 1998) have shown us that they spend a great deal of time performing activities oriented around software design, programming, requirements analysis and testing. They spend considerable time on-the-job learning about software process issues as well as the details of specific types of software architectures – topics they had not learned much about in university. On the other hand, we did not observe practitioners making significant use of certain material they were taught in university such as much of the mathematics. This apparent imbalance suggests that university curricula might be improved in order to better serve the needs of both industry and students. One manager, told us that he would prefer to hire somebody from a good community college, rather than a university, because such people were more likely to have the skills he required.

It is currently a good time to study software practice and curricula because a significant transition is occurring, 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; for this, one needs a body of data from a systematic survey of practitioners.

This report 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 articles submitted to IEEE Software (Lethbridge 1999b) and the Journal of Systems and Software (Lethbridge 1999c). The latter is an abridged version of this report.

Section 2 of this report discusses the survey instrument and how the survey was conducted. Section 3 focuses on the knowledge learned by the participants in their formal education, while section 4 looks at their current knowledge. Section 5 then examines how important the topics were to the participants, while sections 6 and 7 analyse changes in knowledge over time as well as their education and training needs. Section 8 compares the 1998 data to the 1997 data. We leave discussion of the demographics of the participants to the end (Section 9), since it is of secondary importance.

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 participation rate on mailing lists was 5%.

Four distinct questions (see Figure 1) were asked about each of 75 topics listed in Table 1. 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.

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

Figure 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™ / 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

Table 1: The 75 topics in the survey, arranged in categories; alphabetically within each category.

3. Knowledge Learned in Formal Education: Question 1

In this section, we discuss the answers to question 1: In Tables 2 and 3, we present the topics about which participants said they learned most and least respectively. The same data is graphically presented in Figures 2 and 3.

Three separate metrics are presented about each topic:

- The mean score given to each topic in question 1 (Tables 2 and 3 are sorted by this criterion);
- 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*);
- The percentage of participants who gave the topic a score of at least 2 (they learned the basics or more).

Some interesting observations 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 expert.
- There are only 13 topics about 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.

Rank	Topic	Mean of Q1	% rating 4 or 5	% rating > 1	n	Std. Err.	Std. Dev.
1	Specific Programming Languages	3.2	44%	89%	178	0.10	1.36
2	Differential and Integ. Calculus	3.2	52%	89%	176	0.11	1.41
3	Linear Algebra and Matrices	3.1	45%	85%	179	0.11	1.43
4	Probability and Statistics	2.9	34%	87%	179	0.09	1.25
5	Data Structures	2.9	42%	78%	178	0.12	1.55
6	Physics	2.7	31%	82%	173	0.11	1.42
7	Differential Equations	2.7	30%	79%	177	0.11	1.44
8	Set Theory	2.5	29%	75%	175	0.12	1.52
9	Design of Algorithms	2.3	26%	68%	177	0.12	1.54
10	Operating Systems	2.3	20%	71%	178	0.11	1.49
11	Computer System Architecture	2.2	28%	63%	179	0.12	1.65
12	Programming Lang. Theory	2.2	19%	68%	172	0.11	1.5
13	Predicate Logic	2.2	25%	65%	173	0.13	1.64
14	Chemistry	2.2	14%	70%	172	0.11	1.4
15	Comp. Meth. for Numeric Probs.	2.2	18%	67%	177	0.11	1.5
16	Formal Languages	2.2	29%	63%	171	0.13	1.66
17	Dig. Electronics & Dig. Logic	2.1	25%	59%	172	0.13	1.7
18	Comp. Complex. & Alg. Analysis	2.1	25%	59%	175	0.13	1.65
19	Software Architecture	2.0	17%	59%	175	0.12	1.54
20	Microprocessor Architecture	1.9	19%	56%	172	0.12	1.59
21	File Management	1.9	16%	57%	173	0.11	1.48
22	Databases	1.9	18%	56%	179	0.12	1.58
23	Graph Theory	1.9	20%	56%	174	0.12	1.55
24	Analysis and Design Methods	1.9	13%	57%	176	0.12	1.52
25	Second Lang. Other than English	1.8	15%	57%	173	0.12	1.61

Table 2: Topics about which participants had learned most in their formal education: The topics ranked 1 to 25 according to the means of question 1.

Rank	Topic	Mean of Q1	% rating 4 or 5	% rating > 1	n	Std. Err..	Std. Dev.
51	Simulation	1.2	7%	38%	175	0.11	1.43
52	Parallel and Distributed Proc.	1.2	11%	36%	174	0.11	1.44
53	Project Management	1.2	8%	37%	177	0.11	1.48
54	Management	1.2	6%	37%	172	0.10	1.36
55	HCI / User Interfaces	1.1	4%	35%	178	0.10	1.35
56	Psychology	1.1	3%	36%	171	0.10	1.27
57	SW Reliability & Fault Tolerance	1.0	5%	29%	174	0.10	1.28
58	Digital Signal Processing	1.0	10%	31%	167	0.11	1.42
59	Telephony and Telecom.	1.0	6%	31%	173	0.10	1.33
60	Accounting	1.0	6%	33%	168	0.10	1.33
61	Real-Time System Design	1.0	6%	30%	175	0.10	1.34
62	Pattern Recog. and Image Proc.	0.9	8%	26%	174	0.10	1.38
63	Leadership	0.8	3%	26%	174	0.09	1.22
64	Software Metrics	0.8	5%	24%	177	0.09	1.21
65	Security and Cryptography	0.8	4%	26%	171	0.10	1.29
66	Data Acquisition	0.8	2%	23%	167	0.09	1.17
67	Maint., Reeng. and Rev. Engg.	0.8	3%	21%	177	0.08	1.11
68	Marketing	0.7	4%	20%	171	0.09	1.13
69	VLSI	0.6	4%	17%	166	0.09	1.14
70	Robotics	0.6	5%	19%	167	0.09	1.11
71	Software Cost Estimation	0.6	3%	18%	174	0.08	1.05
72	Config. and Release Mgmt.	0.5	2%	15%	172	0.07	0.96
73	Entrepreneurship	0.5	3%	13%	175	0.08	1.07
74	Process Stds. CMM / ISO 9000	0.5	1%	14%	168	0.07	0.93
75	Negotiation	0.5	1%	14%	173	0.07	0.94

Table 3: Topics about which participants had learned least in their formal education: The topics ranked 51 to 75 according to the means of question 1.

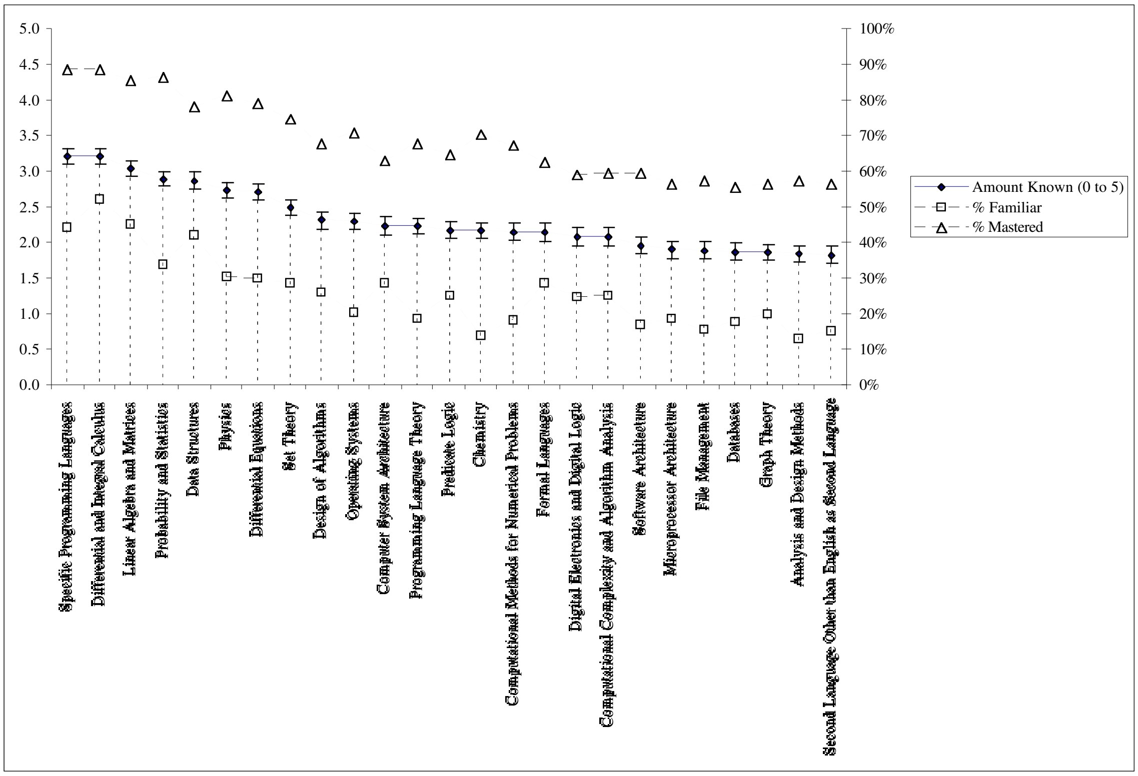

Figure 2: Graph of the three learning metrics for the top 25 topics.

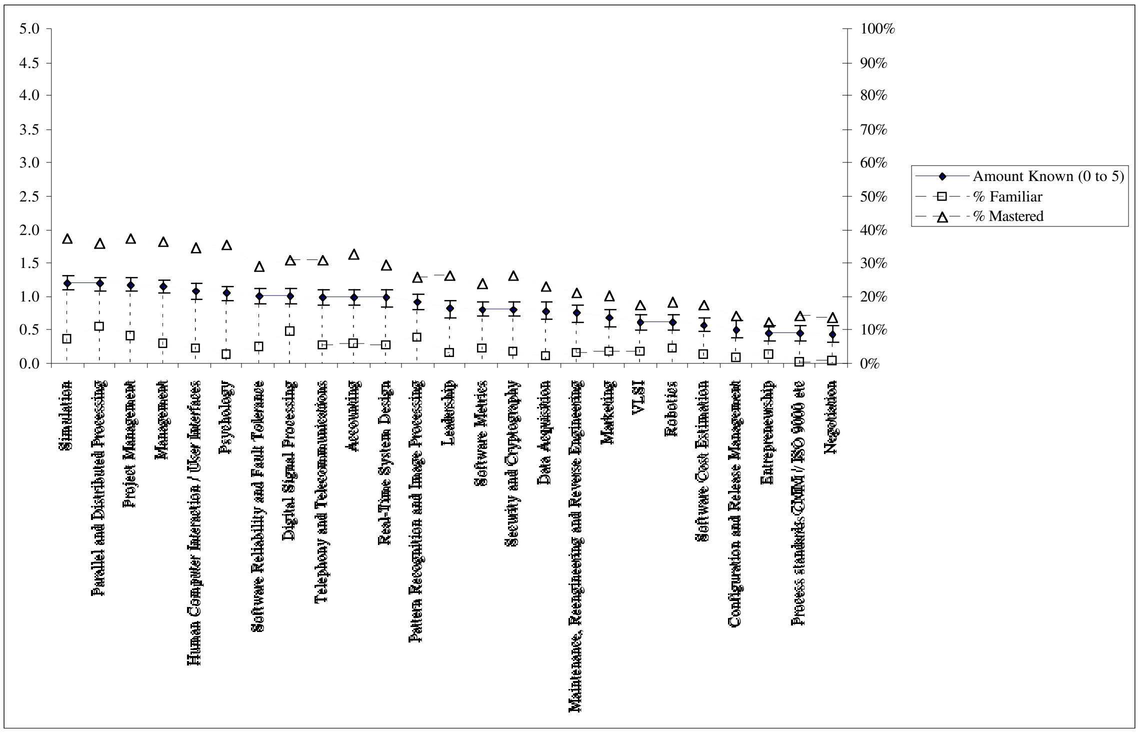

Figure 3: Graph of the three learning metrics for the bottom 25 topics.

Table 4 gives the mean scores given to question 1 for each of the *categories* of topics (see figure 1 for a list of categories).

These figures are only marginally useful because, for example, a few highly-learned software topics are averaged with many software topics that are normally taught as electives, if at all. Nevertheless it is interesting to note that mathematics appears to be the most taught category, while software engineering process is tied with business as the least taught category.

Subject	**Amount learned**
All 75 topics	1.6
General mathematics (14 topics)	2.1
Computer science theory (10 software & math)	1.9
Software (all 36 topics)	1.5
Software design (10 topics)	2.0
Software engineering process (12 topics)	1.0
Specialized software techniques (10 topics)	1.4
Computer engineering (10 topics)	1.7
Hardware (10 topics)	1.3
Non-computing, non-math (15 topics)	1.3
Business (5 topics)	1.0

Table 4: Average learning for the categories of topics. Mean answers to question 1 (see Figure 1).

Tables 5 and 6 provide top-10 rankings of those topics in which people became experts, or minimally familiar, respectively during their education. These tables highlight some of the data found in columns 4 and 5 of Table 2.

Rank	Topic	% Expert
1	Differential and Integral Calculus	52%
2	Linear Algebra and Matrices	45%
3	Specific Programming Languages	44%
4	Data Structures	42%
5	Probability and Statistics	34%
6	Physics	31%
7	Differential Equations	30%
8	Formal Languages	29%
9	Set Theory	29%
10	Programming Language Theory	28%

Table 5: Top 10 topics in terms of the percentage of people who became experts (scored question 1 with 4 or 5) during their education. See Figure 1 for the test of question 1.

Rank	Topic	% Familiar
1	Specific Programming Languages	89%
2	Differential and Integral Calculus	89%
3	Probability and Statistics	87%
4	Linear Algebra and Matrices	85%
5	Physics	82%
6	Differential Equations	79%
7	Data Structures	78%
8	Set Theory	75%
9	Operating Systems	71%
10	Chemistry	70%

Table 6: Top 10 topics in terms of the percentage of people who became minimally familiar with each topic (scored question 1 with 2 or above) during their education. See Figure 1 for the test of question 1.

Variability in the Responses to Question 1

Tables 7 and 8 provide information about the variability of the responses to question 1.

Table 7 shows bipolarity, which is the tendency for there to be two distinct peaks in the data. Bipolarity is a measure of the depth and width of valleys between any two peaks in the histogram for a topic; a monotonically increasing or decreasing histogram would have a bipolarity of 0, as would a histogram with just one peak. Bipolarity reaches a maximum value of 100 if all the responses are evenly distributed between the two extreme buckets of the histogram (buckets 0 and 5 in our case). The formula for bipolarity is:

$$\text{bipolarity} = 100\left(\frac{\displaystyle\sum_{i=1}^{h-2}\sum_{j=i+2}^{h} 2^{j-i-1}\,\text{valley}(i,j)}{2^{h-3}n}\right) \qquad \text{(eq. 1)}$$

where:

n is the total number of participants responding to the question (the sum of all the buckets in the histogram)

h is the number of buckets in the histogram (in our case 6)

$$\begin{aligned}&\text{valley}(i,j) = \\ &\quad \text{if} \quad n_i > \text{intermax}(i,j) \wedge n_j > \text{intermax}(i,j) \\ &\quad \text{then} \quad \left\lfloor n_i - \text{intermin}(i,j), n_j - \text{intermin}(i,j) \right\rfloor \\ &\quad \text{else} \quad 0\end{aligned} \qquad \text{(eq.2)}$$

$$\text{intermax}(i,j) = \left\lceil n_k \right\rceil_{k=i+1}^{j-1} \qquad \text{(eq.3)}$$

$$\text{inter}\min(i,j) = \left\lfloor n_k \right\rfloor_{k=i+1}^{j-1} \qquad \text{(eq.4)}$$

n_i is the number of responses in the i^{th} bucket of the histogram.

The data show that the strongest bipolarity occurs in formal languages. This indicates that students either learned a considerable amount about this topic or else almost nothing – there were few people who responded with intermediate scores. In general the bipolar topics tend to be ones that are normally elective (e.g. second languages), or else are compulsory in certain programs but not others (e.g. formal languages, predicate logic, combinatorics, etc.).

Knowing that a topic has high bipolarity can be useful:

- To those hiring people needing certain knowledge, so they do not assume that all applicants have the knowledge.
- To curriculum designers, so that they can be aware of strong differences of opinion about what might be included in a program.

The standard deviation figures given in Table 8 provide further information about the variability of the data. However, unlike bipolarity, a high standard deviation does not necessarily point to systematic and sharp differences in the types of education received. For example, there was a wide variability in the amount participants learned about automata, digital electronics and parsing; however, none of these topics showed bipolar distributions.

Rank	Topic	Bipolarity
1	Formal Languages	20.6
2	Second Language Other than English	10.1
3	Predicate Logic	9.1
4	Combinatorics	9.0
5	Technical Writing	8.4
6	Analysis and Design Methods	8.2
7	Computational Methods for Numerical Problems	7.8
8	Chemistry	7.5
9	Operating Systems	6.6
10	Network Architecture & Data Transmission	6.5

Table 7: The 10 topics with the most pronounced bipolar distributions, suggesting the presence of distinctly different types of education.

Rank	Topic	Standard Deviation
1	Automata theory	1.70
2	Digital Electronics & Digital Logic	1.70
3	Parsing and Compiler Design	1.67
4	Laplace and Fourier Transforms	1.67
5	Formal Languages	1.66
6	Computational Complexity & Algorithm Analysis	1.65
7	Programming Language Theory	1.65
8	Predicate Logic	1.64
9	Combinatorics	1.63
10	Second Language Other than English	1.61

Table 8: The 10 topics with the most pronounced standard deviations of amount learned, indicating wide variability in education.

Differences Among Subsets of Participants in Answers to Question 1

Tables 9 and 10 compare 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 expert participants had learned more about non-computing topics, while the junior participants learned more about computing topics, particularly those, such as object oriented technology, that have only become widely known in recent years.

Rank	Topic	% Increase for juniors	Junior learning	Expert learning
1	Object Oriented Concepts & Tech.	256%	2.4	0.7
2	Parallel and Distributed Proc.	161%	2.0	0.8
3	SW 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	HCI / User Interfaces	141%	1.6	0.7
7	Software Metrics	135%	1.3	0.5
8	Pattern Recognition and Image Proc.	131%	1.4	0.6
9	Digital Signal Processing	121%	1.4	0.6
10	Config. and Release Management	108%	0.8	0.4
11	Artificial Intelligence	100%	1.7	0.8
12	Data Transmission and Networks	98%	2.6	1.3
13	Process Stds. CMM / ISO 9000	94%	0.6	0.3
14	Network Arch. & Data Trans.	94%	2.4	1.2
15	Project Management	92%	1.7	0.9

Table 9: 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	% Decrease for juniors	Junior learning	Expert learning
1	Psychology	-42%	0.7	1.2
2	Marketing	-41%	0.4	0.8
3	Accounting	-38%	0.7	1.1
4	Economics	-30%	1.4	1.9
5	Philosophy	-29%	1.0	1.4
6	Second Lang. Other than English	-19%	1.5	1.8
7	Analog Electronics	-15%	1.1	1.3
8	Set Theory	-7%	2.4	2.6

Table 10: Topics which junior participants learned less about in their formal education than did expert participants

Tables 11 and 12 provide information about how a computer science 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 'CS/SE' 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.

There are few surprises in the data: Software topics are clearly learned far more by the CS/SE graduates, while the engineers learn more about traditional engineering topics such as electronics and Fourier transforms, as well as physics and chemistry. Engineers also have more background in such computing-related topics as digital signal processing, control theory, telecommunications and simulation. It is interesting to note that engineers have more background in entrepreneurship and also ethics and professionalism than computer scientists.

Rank	**Topic**	**% CS/SE Learned Less**	**Engineering Learning**	**CS/SE Learning**
1	Digital Signal Processing	-59%	2.1	0.9
2	Analog Electronics	-54%	2.5	1.1
3	Control Theory	-50%	2.3	1.2
4	Laplace and Fourier Transforms	-47%	3.1	1.6
5	VLSI	-43%	1.1	0.6
6	Telephony and Telecom.	-36%	1.7	1.1
7	Entrepreneurship	-35%	0.7	0.4
8	Simulation	-31%	1.8	1.2
9	Physics	-25%	3.5	2.6
10	Differential Equations	-24%	3.4	2.6
11	Dig. Electronics & Dig. Logic	-22%	3.0	2.3
12	Ethics and Professionalism	-21%	1.4	1.1
13	Data Acquisition	-20%	1.1	0.9
14	Chemistry	-18%	2.7	2.2
15	Robotics	-17%	0.9	0.7

Table 11: Topics where computer science graduates learned considerably *less* than engineering graduates.

Rank	Topic	% CS/SE Learned More	Engineering Learning	CS/SE Learning
1	Parsing and Compiler Design	90%	1.3	2.4
2	Process Stds. CMM / ISO 9000	90%	0.3	0.7
3	File Management	80%	1.4	2.5
4	Information Retrieval	79%	0.9	1.7
5	Security and Cryptography	73%	0.6	1.1
6	Programming Language Theory	70%	1.7	2.9
7	Databases	69%	1.5	2.5
8	Systems Programming	65%	1.4	2.2
9	Software Cost Estimation	65%	0.5	0.8
10	Software Architecture	61%	1.6	2.6
11	Operating Systems	49%	2.0	3.0
12	Automata theory	44%	1.5	2.1
13	Design of Algorithms	43%	2.0	2.9
14	Information Theory	39%	1.4	1.9
15	Comput. Complexity & Algor. Analysis	38%	1.8	2.5
16	Artificial Intelligence	37%	1.2	1.7
17	Software Design and Patterns	36%	1.6	2.2
18	Requirements Gathering and Analysis	35%	1.2	1.6
19	Formal Languages	35%	2.0	2.7
20	Analysis and Design Methods	34%	1.7	2.3
21	Data Structures	33%	2.7	3.5
22	Graph Theory	31%	1.7	2.3
23	Project Management	29%	1.1	1.4
24	Parallel and Distributed Processing	28%	1.2	1.5
25	Computer Graphics	28%	1.3	1.6

Table 12: Topics where computer science graduates learned considerably *more* than engineering graduates

Table 13 lists the topics for which students with masters or PhD degrees in computer science had learned considerably more during their education compared with those having only undergraduate degrees in computer science.

This data might be useful for managers who are attempting to decide whether to look for applicants with a postgraduate degree, or whether a bachelors degree is sufficient. It is interesting that Table 13 contains topics such as management and marketing that are not normally taught as part of CS graduate programs: Perhaps those who are interested in pursuing graduate degrees were also motivated to learn about these business topics in at some other time.

Rank	**Topic**	**% Increase**
1	Simulation	31%
2	Pattern Recognition and Image Proc.	25%
3	Second Language Other than English	23%
4	Management	22%
5	Marketing	21%
6	Formal Specification Methods	18%
7	Real-Time System Design	18%
8	Configuration and Release Management	18%
9	Robotics	18%
10	Project Management	18%

Table 13: Topics where graduate students in computer science had learned considerably more in their formal education than undergraduates.

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 14 and 15, as well as Figures 4 and 5, give data for the three metrics discussed in the last section: The mean score, the percent that know the topic very well, and the percent that know the topic at least minimally. As the figures show, 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 know them minimally. 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 are courses that were both not extensively taught in university, and also not 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.

Rank	Topic	Mean of Q2	% rating 4 or 5	% rating > 1	n	Std. Err.	Std. Dev.
1	Specific Programming Languages	4.1	78%	97%	179	0.08	1.03
2	Data Structures	3.7	65%	96%	178	0.08	1.12
3	Operating Systems	3.4	46%	96%	179	0.08	1.11
4	Software Design and Patterns	3.4	52%	90%	179	0.10	1.30
5	Software Architecture	3.3	49%	87%	175	0.10	1.36
6	Giving Presentations to an Audience	3.3	50%	89%	172	0.10	1.34
7	Databases	3.2	45%	89%	180	0.09	1.22
8	Object Oriented Concepts & Tech.	3.2	45%	85%	178	0.11	1.41
9	Testing, Verif. & Quality Assurance	3.2	41%	91%	179	0.09	1.18
10	Analysis and Design Methods	3.2	50%	85%	179	0.10	1.33
11	Requirements Gath. & Analysis	3.1	48%	87%	180	0.10	1.37
12	Project Management	3.1	39%	91%	180	0.09	1.25
13	File Management	3.1	45%	84%	176	0.11	1.46
14	Ethics and Professionalism	3.0	38%	84%	173	0.11	1.47
15	Design of Algorithms	3.0	44%	83%	178	0.11	1.46
16	Technical Writing	3.0	41%	85%	172	0.11	1.43
17	Data Transmission and Networks	3.0	39%	87%	179	0.10	1.29
18	Configuration and Release Mgmt.	2.8	34%	79%	174	0.11	1.50
19	HCI / User Interfaces	2.8	30%	81%	179	0.10	1.39
20	Programming Language Theory	2.7	33%	77%	180	0.11	1.48
21	Computer System Architecture	2.7	31%	79%	173	0.11	1.45
22	Performance Meas. & Analysis	2.6	28%	79%	180	0.11	1.43
23	Maintenance, Reeng. and Rev. Engg.	2.6	33%	79%	180	0.11	1.42
24	Network Architecture & Data Trans.	2.6	27%	77%	173	0.11	1.41
25	Systems Programming	2.5	36%	68%	174	0.13	1.67

Table 14: Topics about which participants currently know most. The topics ranked 1 to 25 according to the means of question 2. See Figure 1 for the text of question 2.

Rank	Topic	Mean of Q2	% rating 4 or 5	% rating > 1	n	Std. Err.	Std. Dev.
51	Security and Cryptography	2.0	12%	62%	178	0.11	1.45
52	Predicate Logic	1.9	18%	55%	173	0.12	1.55
53	Telephony and Telecommunications	1.9	14%	57%	168	0.12	1.53
54	Second Language Other than English	1.8	14%	54%	171	0.12	1.56
55	Philosophy	1.7	10%	52%	169	0.11	1.42
56	Graph Theory	1.7	11%	51%	176	0.11	1.43
57	Psychology	1.6	9%	50%	170	0.11	1.38
58	Entrepreneurship	1.6	13%	50%	167	0.12	1.53
59	Information Theory	1.6	13%	49%	169	0.11	1.48
60	Queuing Theory	1.6	9%	47%	176	0.11	1.43
61	Differential Equations	1.6	8%	47%	179	0.10	1.30
62	Accounting	1.5	7%	49%	169	0.10	1.34
63	Data Acquisition	1.5	13%	42%	166	0.12	1.60
64	Automata theory	1.5	13%	44%	169	0.12	1.56
65	Chemistry	1.4	4%	47%	171	0.09	1.18
66	Marketing	1.4	9%	42%	168	0.11	1.42
67	Combinatorics	1.4	8%	43%	164	0.11	1.42
68	Artificial Intelligence	1.4	6%	43%	175	0.10	1.29
69	Digital Signal Processing	1.3	10%	36%	165	0.11	1.44
70	Pattern Recognition and Image Proc.	1.3	7%	36%	174	0.11	1.41
71	Laplace and Fourier Transforms	1.2	6%	37%	172	0.10	1.36
72	Analog Electronics	1.2	7%	36%	169	0.11	1.39
73	Control Theory	1.1	5%	34%	173	0.10	1.32
74	Robotics	0.9	4%	27%	167	0.09	1.19
75	VLSI	0.7	4%	20%	165	0.09	1.19

Table 15: Topics about which participants currently know least. The topics ranked 51 to 75 according to means of question 2. See Figure 1 for the text of question 2.

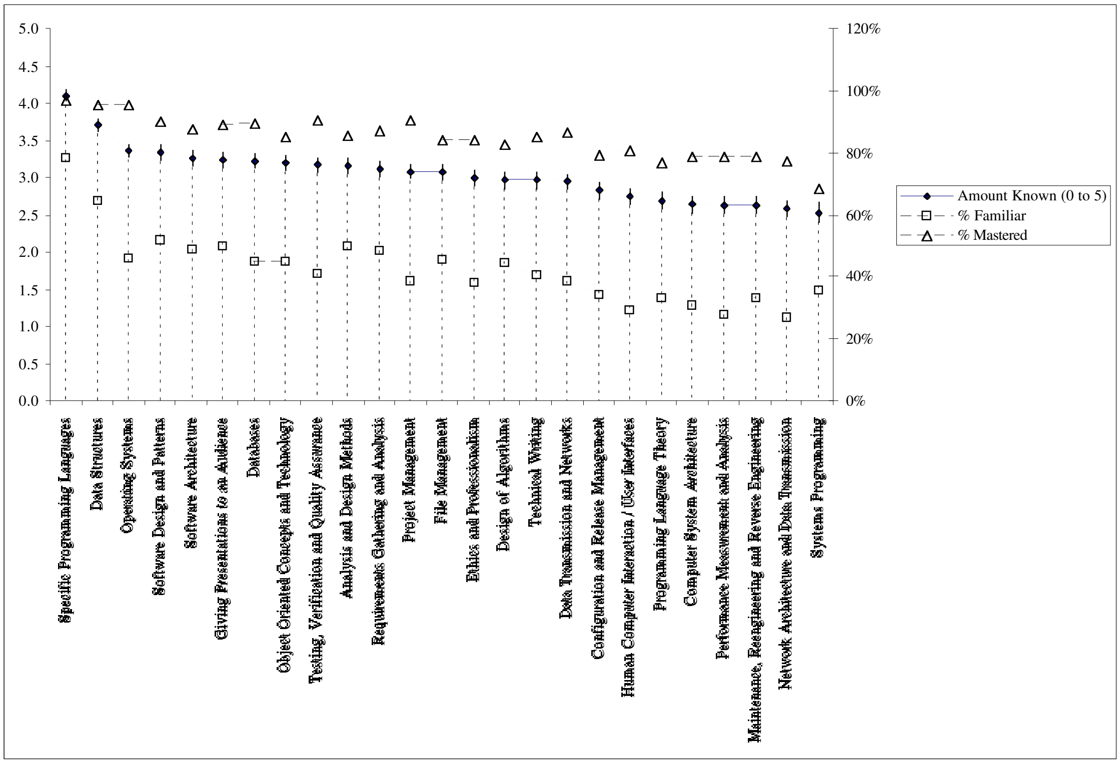

Figure 4: Graph of the three knowledge metrics for the top 25 topics.

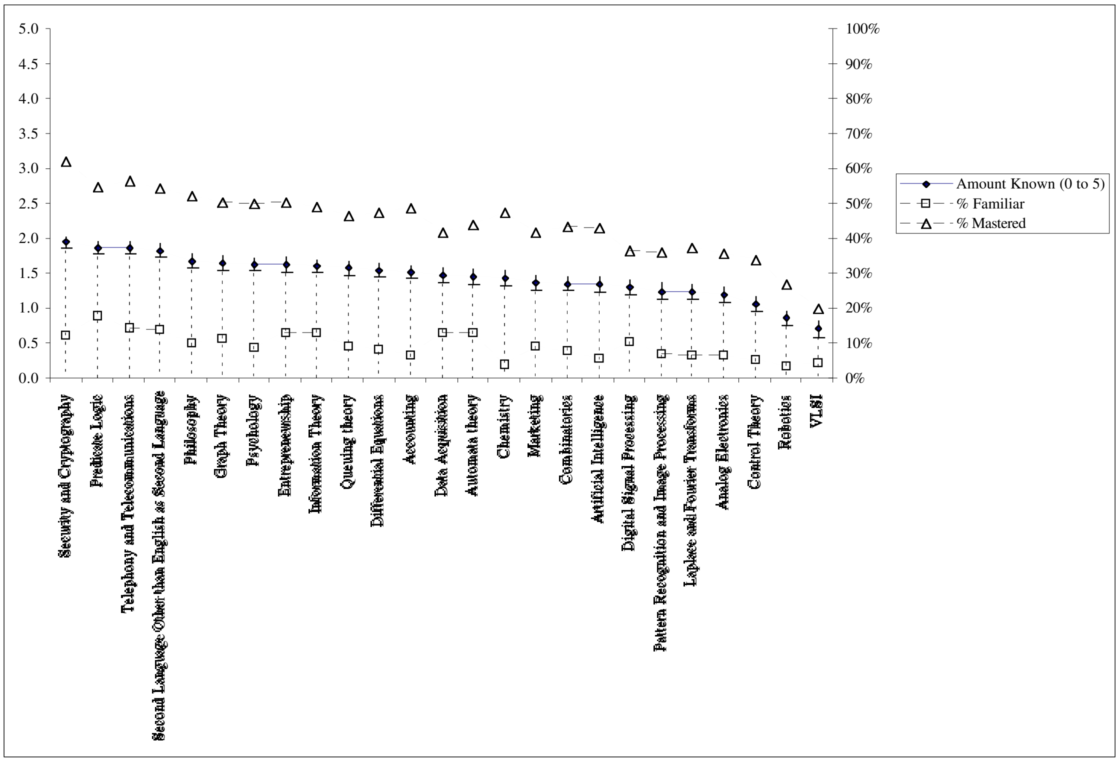

Figure 5: Graph of the knowledge metrics for the bottom 25 topics.

Table 16 shows the mean answers to question 2 for the categories of topics presented in Table 1.

Category	**Amount known**
All 75 topics	2.2
General mathematics (14 topics)	1.7
Computer science theory (10 software & math)	1.8
Software (all 36 topics)	2.6
Software design (10 topics)	3.2
Software engineering process (12 topics)	2.7
Specialized software techniques (10 topics)	2.1
Computer engineering (10 topics)	2.5
Hardware (10 topics)	1.7
Non-computing, non-math (15 topics)	2.1
Business (5 topics)	1.8

Table 16: Amount known about specific categories of topic.

Tables 17 and 18 highlight data from columns 4 and 5 of Table 14. Many of the same topics are in these top-ten lists; the most interesting observation is the appearance of project management near the top of Table 18: Although few people are expert in project management, almost everybody knows the basics.

Rank	Topic	% Expert
1	Specific Programming Languages	78%
2	Data Structures	65%
3	Software Design and Patterns	52%
4	Giving Presentations to an Audience	50%
5	Analysis and Design Methods	50%
6	Software Architecture	49%
7	Requirements Gathering & Analysis	48%
8	Operating Systems	46%
9	File Management	45%
10	Databases	45%

Table 17: Topics with the highest percentage of people who are experts.

Rank	Topic	% Familiar
1	Specific Programming Languages	97%
2	Operating Systems	96%
3	Data Structures	96%
4	Project Management	91%
5	Testing, Verif. & Quality Assurance	91%
6	Software Design and Patterns	90%
7	Databases	89%
8	Giving Presentations to an Audience	89%
9	Software Architecture	87%
10	Requirements Gathering & Analysis	87%

Table 18: Topics with the highest percentage of people who have become familiar with the topic.

Variability of Responses to Question 2

Table 19 presents those topics where there is the greatest bipolarity in the distribution of responses to question 2. A definition of bipolarity was given in equation 1 of section 3.

The data suggest that some organizations use metrics while others do not – with few intermediate cases. Similarly there are sharp differences between those who do cost estimation, lead projects or perform negotiation, and those who do not. The bipolarity in real-time knowledge makes sense because of the natural split between real-time software developers and MIS developers: Differences between these groups will be explored in the next subsection.

Some of these same differences appear in wide standard deviations of current knowledge, presented in table 20. Also notable in Table 20 is the wide variation in the amount of knowledge that participants have about process standards.

Rank	Topic	Bipolarity	Peak 1	Peak 2
1	Software Metrics	11.8	0	3
2	Software Cost Estimation	7.9	0	3
3	Leadership	7.4	0	3
4	Negotiation	6.9	0	3
5	Real-Time System Design	6.8	0	3
6	Queuing Theory	6.1	0	3
7	Data Acquisition	5.8	0	3
8	Parsing and Compiler Design	5.7	0	3
9	Telephony and Telecom.	5.2	0	3
10	Predicate Logic	4.9	0	3
11	Information Retrieval	4.7	0	3
12	Formal Languages	4.5	0	3

Table 19: Topics with the most pronounced bipolar distributions – indicating systematic differences in the kinds of knowledge needed for work.

Rank	Topic	Standard Deviation
1	Systems Programming	1.67
2	Process Stds. CMM / ISO 9000	1.65
3	Real-Time System Design	1.63
4	Formal Languages	1.62
5	Data Acquisition	1.60
6	Software Metrics	1.59
7	Parsing and Compiler Design	1.59
8	Information Retrieval	1.58

Table 20: Topics with the highest standard deviations of knowledge – indicating wide differences in amounts known.

Differences Among Subsets of Participants in Answers to Question 2

This section looks at differences in responses to question 2 from various demographic subsets. See section 9 for details of the demographic groups themselves.

Table 21 and 22 compare the knowledge of Real-Time and MIS developers. There are relatively few surprises: Real-time developers know more about real-time design, as well as related topics such as control theory, digital signal processing, systems programming and electronics. Interestingly, they also know more about process standards and software metrics; this is likely because organizations developing real-time software have greater requirements for reliability, and the software often is more complex.

As expected, MIS participants know more about business topics, information retrieval and databases. However, the number of topics where they beat real-time developers is relatively few, suggesting that real-time developers are more knowledgeable overall.

Rank	Topic	% Increase	Know-ledge of real time particip-ants	Know-ledge of whole sample	Absolute difference
1	Control Theory	37%	1.5	1.1	0.4
2	Real-Time System Design	35%	2.8	2.1	0.7
3	Digital Signal Processing	34%	1.7	1.3	0.4
4	Data Acquisition	29%	1.9	1.5	0.4
5	Robotics	29%	1.1	0.9	0.3
6	Process Stds. CMM / ISO 9000	29%	2.7	2.1	0.6
7	VLSI	21%	0.9	0.7	0.1
8	Analog Electronics	21%	1.5	1.2	0.3
9	Simulation	20%	2.3	2.0	0.4
10	Laplace and Fourier Transforms	20%	1.5	1.2	0.2
11	Microprocessor Architecture	19%	2.7	2.2	0.4
12	Digital Electronics & Digital Logic	19%	2.5	2.1	0.4
13	Software Metrics	17%	2.5	2.1	0.4
14	SW Reliability & Fault Tolerance	15%	2.9	2.5	0.4
15	Systems Programming	13%	2.9	2.5	0.3
16	Computer System Architecture	13%	3.0	2.7	0.4
17	Queuing Theory	13%	1.8	1.6	0.2
18	Software Cost Estimation	13%	2.4	2.2	0.3
19	Automata theory	12%	1.6	1.5	0.2
20	Technical Writing	12%	3.3	3.0	0.3

Table 21: Topics with the greatest relative difference in amount known between real-time developers and the whole sample.

Rank	Topic	% Increase	Know-ledge of MIS particip-ants	Know-ledge of whole sample	Absolute difference
1	Marketing	21%	1.6	1.4	0.3
2	Accounting	19%	1.8	1.5	0.3
3	Psychology	16%	1.9	1.6	0.3
4	Security and Cryptography	14%	2.2	2.0	0.3
5	Information Retrieval	11%	2.6	2.3	0.3
6	Entrepreneurship	10%	1.8	1.6	0.2
7	Databases	10%	3.5	3.2	0.3

Table 22: Topics with the greatest relative difference in amount known between MIS developers and the whole sample.

Tables 23 and 24 compare the current knowledge of junior and expert participants. As might be anticipated, the expert participants have a much longer list of topics in which their knowledge is superior – the list is also very diverse.

Junior participants continue to know more about object-oriented technology (see also Table 9) and a few other topics that are still relatively fresh in their minds from their education (graph theory, predicate logic etc.).

Rank	Topic	% Increase	Know-ledge of junior particip-ants	Know-ledge of whole sample	Absolute difference
1	Graph Theory	13%	1.9	1.7	0.2
2	Object Oriented Concepts & Tech.	12%	3.6	3.2	0.4
3	Combinatorics	8%	1.5	1.4	0.1
4	Predicate Logic	8%	2.0	1.9	0.1
5	Linear Algebra and Matrices	7%	2.4	2.3	0.2
6	Programming Language Theory	5%	2.8	2.7	0.1
7	Computer Graphics	5%	2.1	2.0	0.1

Table 23: Topics that junior participants (up to 4 years experience) know better than the whole sample.

Rank	Topic	% Increase	Know-ledge of expert particip-ants	Know-ledge of whole sample	Absolute difference
1	Data Acquisition	28%	1.9	1.5	0.4
2	Real-Time System Design	25%	2.6	2.1	0.5
3	Software Cost Estimation	24%	2.7	2.2	0.5
4	Robotics	24%	1.1	0.9	0.2
5	Analog Electronics	21%	1.4	1.2	0.2
6	Software Metrics	19%	2.5	2.1	0.4
7	Psychology	16%	1.9	1.6	0.3
8	Economics	16%	2.3	2.0	0.3
9	Simulation	15%	2.3	2.0	0.3
10	Accounting	15%	1.7	1.5	0.2
11	Maint., Reeng. and Reverse Engg.	15%	3.0	2.6	0.4
12	Process Stds. CMM / ISO 9000	13%	2.4	2.1	0.3
13	Management	13%	2.8	2.5	0.3
14	Negotiation	13%	2.3	2.0	0.3
15	Digital Signal Processing	12%	1.5	1.3	0.2
16	VLSI	12%	0.8	0.7	0.1
17	Philosophy	12%	1.9	1.7	0.2
18	Configuration and Release Mgmt.	12%	3.2	2.8	0.3
19	Marketing	11%	1.5	1.4	0.2
20	Systems Programming	11%	2.8	2.5	0.3
21	Laplace and Fourier Transforms	11%	1.4	1.2	0.1
22	Microprocessor Architecture	10%	2.5	2.2	0.2
23	SW Reliability & Fault Tolerance	10%	2.8	2.5	0.2
24	Telephony and Telecom.	10%	2.0	1.9	0.2

Table 24: Topics that expert participants (12 or more years experience) know better than the whole sample.

Table 25 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 topics.

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.

Rank	Topic	% Increase	Know-ledge of managers	Know-ledge of whole sample	Absolute difference
1	Negotiation	50%	3.0	2.0	1.0
2	Management	48%	3.6	2.5	1.2
3	Marketing	42%	1.9	1.4	0.6
4	Process Stds. CMM / ISO 9000	40%	2.9	2.1	0.8
5	Psychology	38%	2.2	1.6	0.6
6	Software Metrics	37%	2.9	2.1	0.8
7	Leadership	35%	3.4	2.5	0.9
8	Project Management	32%	4.1	3.1	1.0
9	Software Cost Estimation	31%	2.8	2.2	0.7
10	Accounting	29%	2.0	1.5	0.4
11	Entrepreneurship	27%	2.1	1.6	0.4
12	Ethics and Professionalism	21%	3.6	3.0	0.6
13	Queuing Theory	21%	1.9	1.6	0.3
14	Requirements Gath. & Analysis	20%	3.8	3.1	0.6
15	Giving Presentations to Audience	19%	3.9	3.3	0.6
16	Information Theory	19%	1.9	1.6	0.3
17	Configuration and Release Mgmt.	18%	3.3	2.8	0.5
18	Real-Time System Design	18%	2.5	2.1	0.4
19	Testing, Verif. & Qual. Assurance	18%	3.8	3.2	0.6
20	Artificial Intelligence	17%	1.6	1.4	0.2
21	Data Acquisition	15%	1.7	1.5	0.2
22	Formal Languages	14%	2.3	2.1	0.3
23	Formal Specification Methods	14%	2.6	2.3	0.3
24	Parallel and Distributed Proc.	13%	2.4	2.1	0.3
25	Network Arch. & Data Trans.	12%	2.9	2.6	0.3
26	Digital Signal Processing	12%	1.5	1.3	0.2
27	Security and Cryptography	12%	2.2	2.0	0.2
28	Pattern Recog. and Image Proc.	12%	1.4	1.3	0.1
29	Technical Writing	12%	3.3	3.0	0.3

Table 25: Topics that managers know better than the whole sample.

Tables 26 and 27 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.

Rank	Topic	% Increase	Knowledge of programmers	Knowledge of whole sample	Absolute difference
1	Design of Algorithms	10%	3.3	3.0	0.3
2	Object Oriented Concepts & Tech.	9%	3.5	3.2	0.3
3	Specific Programming Languages	8%	4.4	4.1	0.3
4	Parsing and Compiler Design	8%	2.1	2.0	0.2
5	Combinatorics	8%	1.5	1.4	0.1

Table 26: Topics that programmers know better than the whole sample.

Rank	Topic	% Increase	Knowledge of programmers	Knowledge of whole sample	Absolute difference
1	Process Stds. CMM / ISO 9000	-23%	1.6	2.1	-0.5
2	Management	-22%	1.9	2.5	-0.5
3	Robotics	-21%	0.7	0.9	-0.2
4	Leadership	-19%	2.1	2.5	-0.5
5	Negotiation	-17%	1.7	2.0	-0.3
6	Software Metrics	-16%	1.8	2.1	-0.3
7	Data Acquisition	-15%	1.2	1.5	-0.2
8	Control Theory	-15%	0.9	1.1	-0.2
9	Software Cost Estimation	-15%	1.8	2.2	-0.3
10	Project Management	-13%	2.7	3.1	-0.4
11	Giving Presentations to Audience	-12%	2.9	3.3	-0.4
12	Marketing	-12%	1.2	1.4	-0.2
13	Entrepreneurship	-12%	1.4	1.6	-0.2
14	Psychology	-11%	1.4	1.6	-0.2
15	Accounting	-10%	1.4	1.5	-0.2
16	Requirements Gath. & Analysis	-10%	2.8	3.1	-0.3
17	Telephony and Telecom.	-10%	1.7	1.9	-0.2
18	Information Theory	-10%	1.5	1.6	-0.2

Table 27: Topics that programmers know less about than the whole sample.

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, 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 analysis of the answers to these questions. Tables 28 to 31 provide rankings of topic importance that combine various analyses. Tables 32-38 look specifically at question 3; Tables 39-45 look at question 4; Table 46 looks at topics that novices with no education were forced to learn, and the remaining tables in the section look at how importance differs according to the participant's background and job function.

Aggregate Measures of Importance

Table 28, on the next page, provides an aggregate importance ranking. A topic is listed on Table 28 if it appears in the top ten of any one of seven other importance tables. The seven tables considered were:

Rankings of the importance participants attributed to the topics' details:

- Table 32: Mean of question 3.
- Table 33: Percent of people who considered the topic's details very important (scored question 3 with 4 or 5).
- Table 34: Percentage of people who considered the topic's details at least minimally important (scored question 3 with at least 2).

Rankings of the influence that participants attributed to the topics:

- Table 39: Mean of question 4.
- Table 40: Percent of people who considered the topic to have influenced them very much (scored question 4 with 4 or 5).
- Table 41: Percent of people who considered the topic to have at least minimally influenced them (scored question 4 with at least 2).

Forced learning:

- Table 46: Topics that people were forced to learn about when they knew nothing to start with.

To arrive at the aggregate importance ranking for the topics, we applied the following formula to each of the top-ten rankings.

$$\text{TopicAggrImportance} = \sum_{table} 11 - topicRanking_{table} \qquad \text{(eq. 5)}$$

The most interesting observations about table 28 are: a) no math topics appear; b) 14 out of 17 topics (including the top 5) are software topics; and c) professional topics such as ethics, technical writing and giving presentations rank very high.

Rank	Topic *Table:*	# of top-10s	Aggregate importance (eq. 5)	***32*** Details mean (rank)	***33*** Details % important (rank)	***34*** Details minimally important (rank)	***39*** Influence mean (rank)	***40*** Influenced % important (rank)	***41*** Influenced minimally (rank)	***46*** Novice forced to learn (rank)
1	Data Structures	7	**61**	2	2	1	4	4	1	2
2	Specific Programming Languages	7	**59**	1	1	2	1	3	9	1
3	Software Design and Patterns	6	**44**	3	3	6	2	1	7	
4	Requirements Gath. & Analysis	7	**40**	5	7	3	6	5	2	9
5	Software Architecture	6	**33**	4	6	5	8	6	4	
6	Ethics and Professionalism	4	**21**				3	2	8	10
7	HCI / User Interfaces	4	**21**			7	5	8	3	
8	Object Oriented Concepts & Tech.	6	**20**	10	10		7	7	6	6
9	Project Management	4	**17**	8		4	10			5
10	Analysis and Design Methods	5	**13**	9	9		9	10	5	
11	Technical Writing	3	**12**	7	5	9				
12	Giving Presentations to Audience	2	**12**	6	4					
13	Testing, Verif. & Qual. Assurance	3	**11**			8			10	4
14	Operating Systems	1	**8**							3
15	Configuration and Release Mgmt.	2	**6**		8					8
16	Databases	1	**4**							7
17	Design of Algorithms	2	**3**			10		9		

Table 28: The most important topics, determined by combining the top-10 rankings from various other tables in this section. Each of the 17 topics appeared in at least one top-10 list. Topics nearest the top of this list appeared near the top of the other lists.

Another approach to computing overall importance of topics is to simply take the average of questions 3 and 4. Results of this computation for all 75 topics are presented in tables 29 through 31.

This approach gives similar results to the approach taken for Table 28; topics are only shifted up and down a very few points in the rankings.

The most interesting data is found near the top of table 29, the most important topics that every software engineer should presumably know; and near the bottom of table 31, topics that most software engineers clearly do not need to know.

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	HCI / User Interfaces	3.3	3.3	3.3
7	Object Oriented Concepts & Tech.	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, Verif. & 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 Trans.	2.8	2.8	2.7
25	Negotiation	2.8	2.9	2.6

Table 29: The most important 25 topics – based on the average of both importance of details and influence.

Rank	Topic	Overall Importance	Details (Q3)	Influence (Q4)
26	Performance Measurement & Analysis	2.7	2.8	2.6
27	Maintenance, Reeng. 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	Comput. 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 30: The middle 25 topics in terms of importance – based on the average of both importance of details and influence. Continuation of Table 29.

Rank	Topic	Overall Importance	Details (Q3)	Influence (Q4)
51	Marketing	2.1	2.0	2.3
52	Comput. Methods for Numeric Probs.	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 Proc.	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 31: The least important 25 topics – based on the average of both importance of details and influence. Continuation of Table 30.

Importance Compared with Amount Learned and Current Knowledge

Figures 6, 7 and 8 present the topics ordered by overall importance, as in Tables 29 through 31, with the addition of data about the amount learned in formal education (Q1; discussed in Section 3) and the amount currently known (Q2; discussed in Section 4).

The data from question 1 (amount learned) is shown using boxes, and data from question 2 (amount currently known) is shown by the ends of the vertical bars. It can be seen that the amount currently known is quite similar to the importance, while there is a considerable difference between the amount learned and the importance. In particular, Table 6 shows many topics where the amount learned is considerably less than the importance; on the other hand Table 8 shows 10 topics where the amount learned exceeds the importance. This relationship is discussed further in Tables 64 and 65 in Section 7.

The vertical lines indicating amount currently known in most cases extend upwards from the boxes. This indicates how much the participants have learned since their formal education. In a few cases (in Figure 7 and 8), however, the vertical lines extend downward into the boxes, indicating that material has been forgotten since education. This phenomenon will be discussed further in Section 6.

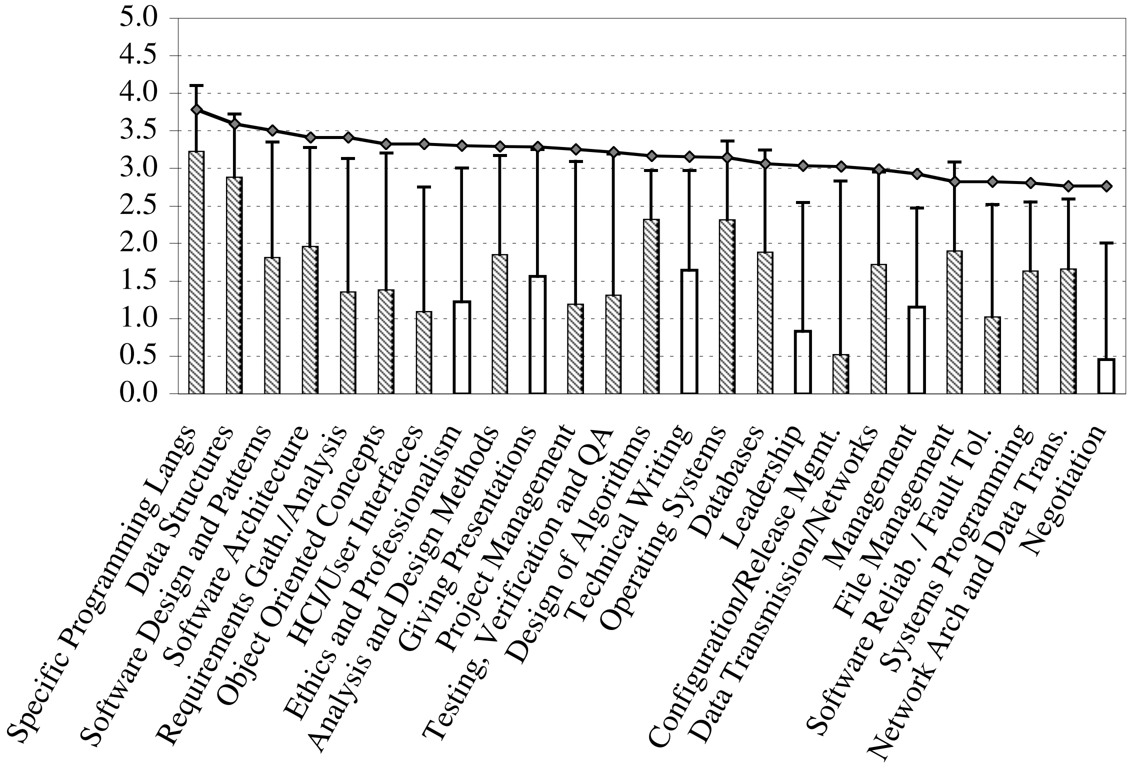

Figure 6. The 25 topics considered most important by participants. The connected line shows overall importance; boxes represent amount learned during education; vertical lines represent learning since education. (Shaded boxes represent computing or engineering topics, unshaded boxes highlight other topics).

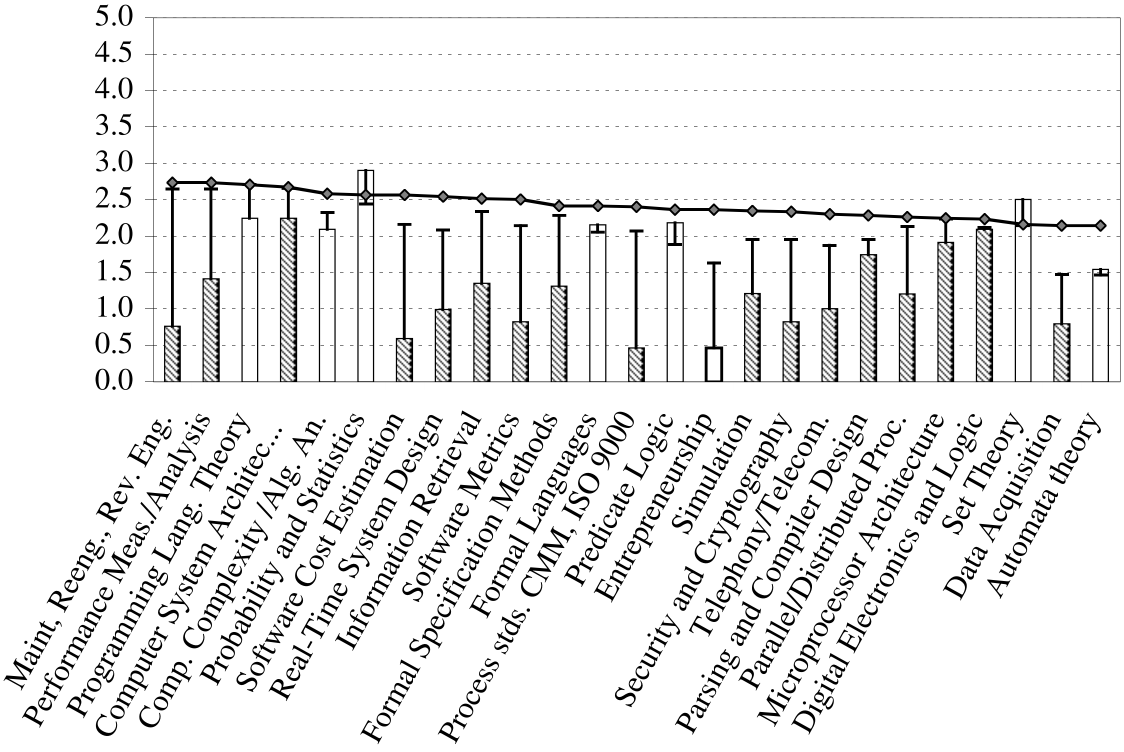

Figure 7. The 25 intermediate topics in terms of importance (continuation of Figure 6 using the same notation). Vertical lines extending downward represent material forgotten following education. Unshaded boxes without a bold outline highlight theory and mathematics topics.

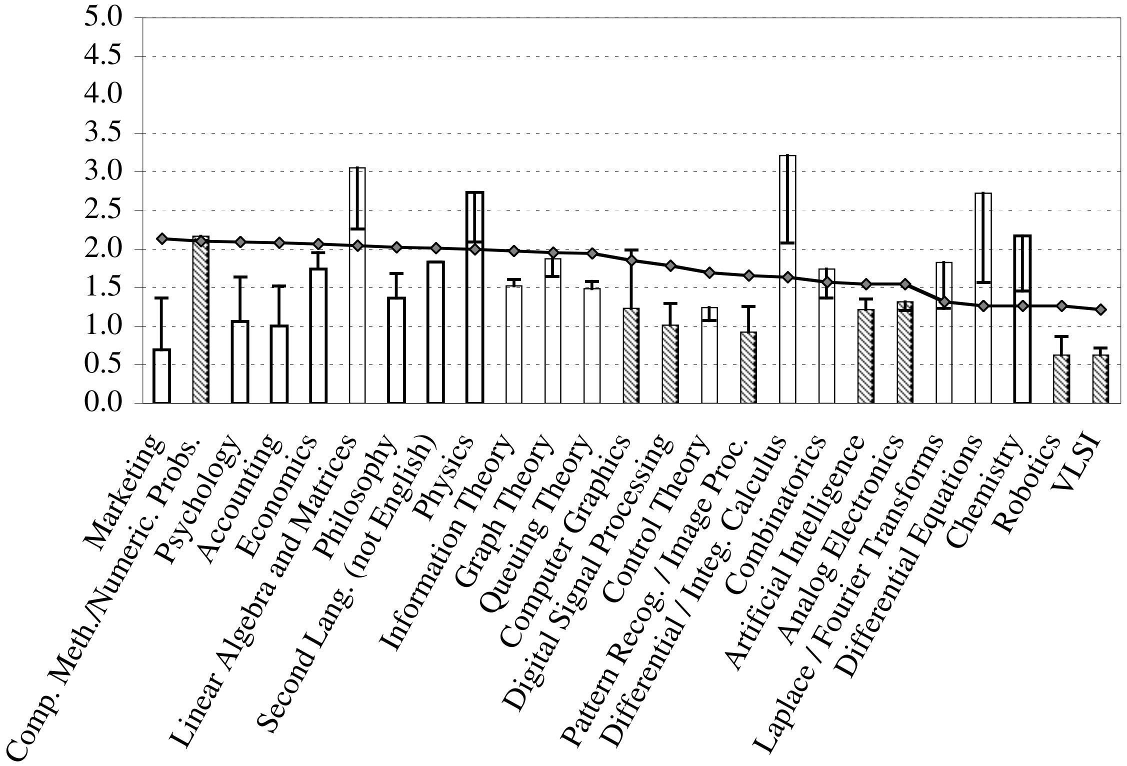

Figure 8. The 25 topics considered least important (Continuation of Figures 6 and 7 using the same notation).

Importance of The Details: Question 3

Tables 32 to 38 present the data for the importance of the details, one of the two components of importance investigated by the survey. In the 1997 survey (see section 8) we did not separate importance into the two components, however participants informed us that some topics might have influenced their thinking even though they had little use for the details and vice-versa.

Table 32 has most of the same topics in the same order as in Table 29 (combined overall importance), including the top five. It is interesting to note, however, the topics nearest 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 the details.

Rank	Topic	Importance of details	Standard Deviation
1	Specific Programming Languages	4.1	1.19
2	Data Structures	3.7	1.22
3	Software Design and Patterns	3.6	1.34
4	Software Architecture	3.5	1.29
5	Requirements Gathering & Analysis	3.5	1.22
6	Giving Presentations to an Audience	3.5	1.45
7	Technical Writing	3.4	1.41
8	Project Management	3.4	1.32
9	Analysis and Design Methods	3.3	1.38
10	Object Oriented Concepts & Tech.	3.3	1.45
11	Operating Systems	3.3	1.31
12	HCI / User Interfaces	3.3	1.23
13	Testing, Verif. & Quality Assurance	3.3	1.30
14	Databases	3.3	1.51
15	Configuration and Release Mgmt.	3.3	1.35
16	Design of Algorithms	3.3	1.30
17	File Management	3.2	1.34
18	Ethics and Professionalism	3.2	1.50
19	Data Transmission and Networks	3.1	1.30
20	Leadership	3.1	1.46

Table 32: Topics whose details are perceived to be most important. Mean responses to question 3 (see Table 35 for the bottom scores).

Tables 33 and 34 show complementary data to Table 32, the percentages of people who believe the topics to be very important, or minimally important, respectively.

Many of the topics in these lists are also in Table 32, indicating widespread agreement about the most important topics. A significant point to note in Table 33 is that details of configuration and release management are considered of moderate importance (ranked 15 in Table 32), but have a particularly large number of people who consider them to be very important (the ranking has risen to 8 in Table 33).

In Table 9, some topics are higher in the rankings than in Table 32, including requirements gathering, project management and user interfaces: This indicates a widespread acknowledgement that the details are useful.

Rank	Topic	% who believe important
1	Specific Programming Languages	75%
2	Data Structures	60%
3	Software Design and Patterns	58%
4	Giving Presentations to an Audience	57%
5	Technical Writing	53%
6	Software Architecture	53%
7	Requirements Gathering & Analysis	53%
8	Configuration and Release Management	48%
9	Analysis and Design Methods	48%
10	Object Oriented Concepts & Tech.	48%
11	Databases	48%
12	Design of Algorithms	47%

Table 33: Topics by the percentage who perceive the details to be very important (those who rated question 3 with 4 or 5).

Rank	Topic	% believe at least minimally important
1	Data Structures	96%
2	Specific Programming Languages	95%
3	Requirements Gathering & Analysis	93%
4	Project Management	92%
5	Software Architecture	92%
6	Software Design and Patterns	91%
7	HCI / User Interfaces	91%
8	Testing, Verification & Qual. Assurance	90%
9	Technical Writing	90%
10	Design of Algorithms	90%

Table 34: Topics by the percentage who believe the details to be at least of minimal importance (those who rated question 3 with at least 2).

Tables 35 and 36 show 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 these tables.

Rank	Topic	Importance
1	Chemistry	0.9
2	Differential Equations	1.1
3	VLSI	1.1
4	Robotics	1.2
5	Laplace and Fourier Transforms	1.3
6	Artificial Intelligence	1.3
7	Differential and Integral Calculus	1.3
8	Analog Electronics	1.4
9	Philosophy	1.5
10	Combinatorics	1.5

Table 35: Topics whose details are perceived to be least important: Mean responses to question 3 (see Table 32 for the top scores).

Rank	Topic	% who believe not important
1	Differential Equations	71%
2	Chemistry	71%
3	VLSI	70%
4	Robotics	68%
5	Differential and Integral Calculus	66%
6	Laplace and Fourier Transforms	65%
7	Artificial Intelligence	61%
8	Pattern Recognition and Image Processing	59%
9	Analog Electronics	55%
10	Control Theory	53%
11	Philosophy	53%
12	Digital Signal Processing	48%
13	Physics	48%
14	Combinatorics	47%
15	Second Language Other than English	46%

Table 36: Topics by the percentage who believe the details to be not at all important (those who rated question 3 with 0 or 1).

Variability of Responses to Question 3

Tables 37 and 38 show topics where there were differences of opinion about the importance of details.

Table 37 shows two topics where there is a bipolarity (defined in section 3) in the opinions, suggesting two schools of thought about the importance of the details of these topics.

Table 20 shows a variety of topics where there is a variety of opinion about the details. It can be seen that many of these are topics that are used only in particular job functions, or particular types of organizations.

Rank	Topic	Bipolarity	peak 1	peak 2	Details importance
1	Economics	5.5	0	3	1.8
2	Digital Signal Processing	3.7	0	3	1.7

Table 37: Topics with the most pronounced bipolar distribution in terms of the importance of details, indicating the presence of specific subpopulations with differing needs

Rank	Topic	Std. Dev.
1	Second Language Other than English	1.73
2	Entrepreneurship	1.70
3	Formal Languages	1.64
4	Real-Time System Design	1.62
5	Telephony and Telecommunications	1.60
6	Marketing	1.60
7	Digital Electronics & Digital Logic	1.56
8	Process Standards CMM / ISO 9000	1.56
9	Software Cost Estimation	1.56
10	Negotiation	1.55
11	Parsing and Compiler Design	1.53
12	Microprocessor Architecture	1.53

Table 38: Topics with the widest standard deviations of importance of details - indicating widely varying needs

Influence of Topics: Question 4

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

Tables 39, 40 and 41 present the answers to question 4 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 in each of the tables: Ethics and professionalism, and human computer interaction / user interfaces.

It is interesting to note that 'specific programming languages' is at the top of Table 39: Not only are the details of programming languages important, but learning them has been more influential, on average, than anything else. However in Table 41, this topic drops down to 9th rank behind other topics that have had more widespread at-least-minimal influence.

Rank	Topic	Influence
1	Specific Programming Languages	3.5
2	Software Design and Patterns	3.5
3	Ethics and Professionalism	3.4
4	Data Structures	3.4
5	HCI / User Interfaces	3.3
6	Requirements Gathering & Analysis	3.3
7	Object Oriented Concepts & Tech.	3.3
8	Software Architecture	3.3
9	Analysis and Design Methods	3.3
10	Project Management	3.2
11	Testing, Verif. & Quality Assurance	3.1

Table 39: Topics that have had the most influence. Mean responses to question 4.

Rank	Topic	% influenced
1	Software Design and Patterns	57%
2	Ethics and Professionalism	56%
3	Specific Programming Languages	55%
4	Data Structures	52%
5	Requirements Gathering & Analysis	51%
6	Software Architecture	49%
7	Object Oriented Concepts & Tech.	49%
8	HCI / User Interfaces	49%
9	Design of Algorithms	47%
10	Analysis and Design Methods	45%

Table 40: Percentage of participants who believe they were very influenced by each topic (those who rated question 4 with 4 or 5).

Rank	Topic	% minimally influenced
1	Data Structures	91%
2	Requirements Gathering & Analysis	90%
3	HCI / User Interfaces	88%
4	Software Architecture	87%
5	Analysis and Design Methods	87%
6	Object Oriented Concepts & Tech.	86%
7	Software Design and Patterns	86%
8	Ethics and Professionalism	86%
9	Specific Programming Languages	86%
10	Testing, Verif. & Quality Assurance	85%

Table 41: Percentage of participants that were at least slightly influenced by the topic (those who rated question 4 with at least 2).

Variability of Responses to Question 4

Tables 42 and 43 show topics where there was considerable variability of opinion about their influence.

Participants were somewhat divided about the influence obtained from learning about process standards, second languages and physics. On the other hand, there was a wide range of opinion, but not a division of opinion, about the influence obtained by learning about other topics such as entrepreneurship, formal languages, marketing, logic, etc.

Rank	**Topic**	**Bipol-arity**	**Peak 1**	**Peak 2**	**Peak 3**	**Infl-uence**
1	Process Standards CMM / ISO 9000	9.0	0	4		2.4
2	Second Language Other than English	8.8	0	3		2.1
3	Physics	6.9	0	3		2.3
4	Digital Electronics & Digital Logic	4.9	0	3		2.3
5	Parsing and Compiler Design	3.3	0	3		2.3
6	Formal Languages	2.9	0	3		2.4
7	Differential and Integral Calculus	2.8	0	3		1.9

Table 42: Topics with the most pronounced bipolar distribution in terms of influence, indicating the presence of specific subpopulations with differing perceptions.

Rank	**Topic**	**St. Dev.**
1	Process Standards CMM / ISO 9000	1.73
2	Entrepreneurship	1.71
3	Second Language Other than English	1.69
4	Physics	1.69
5	Formal Languages	1.68
6	Marketing	1.66
7	Predicate Logic	1.64
8	Parsing and Compiler Design	1.63
9	Digital Electronics & Digital Logic	1.62
10	Real-Time System Design	1.62

Table 43: Topics with the widest standard deviations of influence - indicating widely varying perceptions.

Comparison of Importance vs. Influence

Tables 44 and 45 compare the means of questions 3 and 4: Importance of details vs. general influence.

Table 44 shows those topics (some of them with low importance scores) where participants felt whatever importance they had came from their general influence. It is interesting to note that chemistry and calculus, two widely-taught topics are considered unimportant by participants primarily because their details are unimportant: At least some people think that learning these topics has been of some influence.

The topics listed in table 45 are mostly technical topics that presumably do not inspire participants: Their importance, if high, stems from a need to know the details.

Rank	Topic	% by which influence was greater	Influence	Details
1	Chemistry	70%	1.6	0.9
2	Philosophy	63%	2.5	1.5
3	Differential and Integral Calculus	47%	1.9	1.3
4	Physics	43%	2.3	1.6
5	Artificial Intelligence	41%	1.8	1.3
6	Differential Equations	30%	1.4	1.1
7	Economics	28%	2.3	1.8
8	Analog Electronics	23%	1.7	1.4
9	Robotics	16%	1.4	1.2
10	Entrepreneurship	15%	2.5	2.2

Table 44: Topics where influence was highest relative to details

Rank	Topic	% by which influence was less	Influence	Details
1	File Management	-24%	2.4	3.2
2	Technical Writing	-16%	2.9	3.4
3	Specific Programming Languages	-15%	3.5	4.1
4	Configuration and Release Mgmt.	-15%	2.8	3.3
5	Software Cost Estimation	-14%	2.4	2.7
6	Databases	-13%	2.8	3.3
7	Information Retrieval	-13%	2.3	2.7
8	Queuing Theory	-13%	1.8	2.1
9	Giving Presentations to an Audience	-11%	3.1	3.5
10	Data Transmission and Networks	-11%	2.8	3.1

Table 45: Topics where influence was lowest relative to details, suggesting that the material didn't make participants think in new ways

Forced Learning: An Alternative Measure of Importance

Table 46 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 very substantially in order to meet the requirements of work, then the topic must be important.

Table 46 lists topics with the highest increase in knowledge since completion of formal education, i.e. the difference between question 2 and question 1. 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 ways 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 46 might be the most suitable material for such retraining.

Note that increase in knowledge, irrespective of the amount learned in education, is discussed further in section 6.

Rank	**Topic**	**Forced Learning**	**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 46: Topics which people where forced to learn most about on the job when they knew almost nothing to start with – indirectly indicating importance.

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. The tables only include topics where there was a significance in overall importance from developers at large and where overall importance was greater than 2.5. The tables are ordered by overall importance so as not to give false impressions about which topics are important.

Tables 47 and 48 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. It is interesting to compare these tables with Tables 21 and 22.

Rank	Topic	% increased importance	Importance for real time developers	Importance for developers 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 Tol.	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
11	Data Acquisition	19%	2.5	2.1
12	Parallel and Distributed Processing	11%	2.5	2.3
13	Microprocessor Architecture	11%	2.5	2.2
14	Telephony and Telecommunications	8%	2.5	2.3

Table 47: Important topics (>= 2.5) that real-time developers considered more important than other developers.

Rank	Topic	% increased importance	Importance for MIS	Importance for all
1	Databases	10%	3.4	3.1
2	Information Retrieval	6%	2.6	2.5
3	Entrepreneurship	12%	2.6	2.4
4	Security and Cryptography	9%	2.5	2.3

Table 48: Important topics (>=2.5) that MIS developers consider more important than developers at large.

Tables 49 and 50 compare the importance scores given by participants with different levels of experience.

In Table 49, it can be seen that those topics that junior developers consider more important than developers at large have to do with programming and low-level design. On the other hand, it can be seen in Table 50 that experts ascribe relatively greater importance to topics that have to do mostly with management, communication and professionalism.

These tables can be compared with Tables 23 and 24 that look at the current knowledge of the same subsets of the participants.

Rank	Topic	% increase for juniors	Importance for juniors
1	Data Structures	5%	3.8
2	Software Design and Patterns	5%	3.7
3	Object Oriented Concepts & Technology	8%	3.6
4	Systems Programming	6%	3.0
5	Programming Language Theory	9%	2.9
6	Parsing and Compiler Design	9%	2.5

Table 49: Important topics (>=2.5) that junior developers consider more important than developers at large.

Rank	Topic	% increase for experts	Importance for experts	Importance for developers at large
1	Ethics and Professionalism	10%	3.6	3.3
2	Giving Presentations to an Audience	5%	3.5	3.3
3	Project Management	5%	3.4	3.3
4	Technical Writing	7%	3.4	3.1
5	Databases	5%	3.2	3.1
6	Management	6%	3.1	2.9
7	Software Cost Estimation	11%	2.8	2.6
8	Real-Time System Design	7%	2.7	2.5
9	Probability and Statistics	6%	2.7	2.6
10	Formal Languages	8%	2.6	2.4
11	Process Standards CMM / ISO 9000	5%	2.5	2.4

Table 50: Important topics (>=2.5) that expert developers consider more important than developers at large.

Table 51 gives a long list of topics that managers consider more important than developers at large. There are few surprises in this table: 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 25, the topics which managers know most about.

Rank	Topic	% increase for managers	Importance for managers	Importance for developers 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 Trans.	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 Stds. CMM / ISO 9000	32%	3.2	2.4
16	Software Metrics	24%	3.1	2.5
17	File Management	6%	3.0	2.8
18	Performance Meas. & Analysis	8%	3.0	2.7
19	Maintenance, Reeng. and Rev. Engg.	5%	2.9	2.7
20	Probability and Statistics	10%	2.8	2.6
21	Formal Specification Methods	17%	2.8	2.4
22	Simulation	20%	2.8	2.3
23	Marketing	31%	2.8	2.1
24	Real-Time System Design	10%	2.8	2.5
25	Information Retrieval	8%	2.7	2.5
26	Entrepreneurship	14%	2.7	2.4
27	Parallel and Distributed Processing	14%	2.6	2.3
28	Telephony and Telecommunications	11%	2.6	2.3
29	Psychology	21%	2.5	2.1
30	Security and Cryptography	8%	2.5	2.3
31	Economics	19%	2.5	2.1

Table 51: Important topics (>=2.5) that managers consider more important than developers at large.

The topics that programmers consider more important than developers at large, as listed in Table 52, also hold few surprises.

Rank	Topic	% increase for programmers	Importance for programmers	Importance for developers at large
1	Specific Programming Languages	8%	4.1	3.8
2	Object Oriented Concepts & Tech.	5%	3.5	3.3
3	Maintenance, Reeng. and Rev. Engg.	5%	2.9	2.7
4	Comput. Complexity & Algor. Analysis	7%	2.8	2.6
5	Parsing and Compiler Design	10%	2.5	2.3

Table 52: Important topics (>=2.5) that programmers consider more important than developers at large.

Table 53 presents a list of topics that those most knowledgeable consider to be more important than do developers at large. The most knowledgeable participants are those whose weighted average knowledge on question 2 was in the top 40%. The weighted average was calculated by counting the most important topics with a greater weight.

The objective of doing this separate analysis was to eliminate any bias that might have been introduced by the presence of participants with relatively little knowledge about what is actually important to a software developer.

It is not surprising that somebody who learns more about a topic should find it more important: Such a person may well have been motivated to learn a topic precisely because it was important to them. Also, in learning a topic they become both personally involved in the topic and see its potential.

The topics that rank highest in Table 53 are thus those topics that have the greatest synergy between learning and appreciation of importance. Interestingly, the highest ranking topics are high-level computer topics that have to do with architecture (software architecture, computer system architecture and microprocessor architecture), design (real-time design and compiler design), parallel processing and computational complexity.

One very significant piece of information does not appear on the table: There was exactly one topic where the more knowledgeable people were, the less they found this topic important: Differential and Integral Calculus.

One might have expected that the set of 83 expert software developers, discussed in Table 50, would be very similar to the set of knowledgeable developers. In fact, the two sets only share about 35 participants.

Rank	Topic	% increase	Importance for the most knowledge-able participants	Importance for all participants
1	Specific Programming Languages	7%	4.1	3.8
2	Data Structures	13%	4.0	3.6
3	Software Design and Patterns	14%	4.0	3.5
4	Software Architecture	16%	4.0	3.4
5	Analysis and Design Methods	13%	3.7	3.3
6	Requirements Gath. & Analysis	6%	3.6	3.4
7	Object Oriented Concepts & Tech.	9%	3.6	3.3
8	Testing, Verif. & Qual. Assurance	11%	3.6	3.2
9	Operating Systems	14%	3.6	3.1
10	Design of Algorithms	12%	3.6	3.2
11	HCI / User Interfaces	6%	3.5	3.3
12	Databases	7%	3.3	3.1
13	Computer System Architecture	18%	3.1	2.7
14	SW Reliability & Fault Tolerance	10%	3.1	2.8
15	File Management	8%	3.1	2.8
16	Systems Programming	9%	3.0	2.8
17	Comput. Complexity & Algor. Analysis	16%	3.0	2.6
18	Maint., Reeng. and Rev. Engg.	9%	3.0	2.7
19	Real-Time System Design	16%	3.0	2.5
20	Negotiation	7%	2.9	2.8
21	Performance Meas. & Analysis	7%	2.9	2.7
22	Programming Lang. Theory	6%	2.9	2.7
23	Software Cost Estimation	9%	2.8	2.6
24	Formal Specification Methods	14%	2.7	2.4
25	Entrepreneurship	15%	2.7	2.4
26	Information Retrieval	7%	2.7	2.5
27	Parsing and Compiler Design	18%	2.7	2.3
28	Parallel and Distributed Proc.	18%	2.7	2.3
29	Predicate Logic	11%	2.6	2.4
30	Microprocessor Architecture	17%	2.6	2.2
31	Simulation	10%	2.6	2.3
32	Formal Languages	7%	2.6	2.4
33	Security and Cryptography	7%	2.5	2.3

Table 53: Important topics (>=2.5) that the most knowledgeable people consider more important than developers at large.

Table 54 provides a similar analysis to Table 53, but this time the participants selected were those who scored above the 50th percentile in terms of mathematics knowledge. We wanted to see if these people would rank mathematics substantially higher in importance than it was ranked by others.

Indeed we do see two mathematics topics with significant increases in importance: Set theory and linear algebra. Notably absent is calculus – even those knowledgeable in mathematics don't see it as important to computing.

Physics and parsing also show significant increases in importance among those knowledgeable in mathematics.

Rank	Topic	% increase	Importance for those most knowledge-able in math	Importance for all
1	Software Design and Patterns	8%	3.8	3.5
2	Software Architecture	10%	3.8	3.4
3	Design of Algorithms	15%	3.6	3.2
4	Object Oriented Concepts & Tech.	8%	3.6	3.3
5	Operating Systems	7%	3.4	3.1
6	Programming Lang. Theory	11%	3.0	2.7
7	Comput. Complexity & Algor. Analysis	16%	3.0	2.6
8	SW Reliability & Fault Tolerance	6%	3.0	2.8
9	Network Arch. & Data Trans.	8%	3.0	2.8
10	Systems Programming	6%	3.0	2.8
11	Computer System Architecture	10%	2.9	2.7
12	Real-Time System Design	13%	2.9	2.5
13	Parsing and Compiler Design	26%	2.9	2.3
14	Formal Languages	19%	2.9	2.4
15	Probability and Statistics	11%	2.8	2.6
16	Set Theory	30%	2.8	2.2
17	Predicate Logic	19%	2.8	2.4
18	Simulation	19%	2.8	2.3
19	Dig. Electronics & Dig. Logic	24%	2.7	2.2
20	Linear Algebra and Matrices	34%	2.7	2.0
21	Formal Specification Methods	10%	2.7	2.4
22	Physics	33%	2.6	2.0
23	Microprocessor Architecture	14%	2.6	2.2
24	Parallel and Distributed Proc.	12%	2.5	2.3
25	Automata theory	17%	2.5	2.1

Table 54: Important topics (>=2.5) which the most knowledgeable people in mathematics consider more important than developers at large.

Table 55 gives a similar analysis to Tables 53 and 54, but this time considering those who know most about software process.

All but one of the process topics listed in Table 1 appear in Table 55; most notably cost estimation and formal specification. In addition, process experts find real-time design and parsing to be important – perhaps because experts in those areas tend to become process experts.

Topic	**% increase**	**Importance for those most knowledge-able about the software process**	**Importance for all participants**
Software Design and Patterns	18%	4.1	3.5
Software Architecture	21%	4.1	3.4
Requirements Gath. & Analysis	17%	4.0	3.4
Analysis and Design Methods	20%	3.9	3.3
Data Structures	9%	3.9	3.6
Project Management	16%	3.8	3.3
Testing, Verification & Quality Assurance	14%	3.7	3.2
Giving Presentations to an Audience	11%	3.6	3.3
Design of Algorithms	12%	3.5	3.2
Leadership	12%	3.4	3.0
Configuration and Release Mgmt.	12%	3.4	3.0
Negotiation	17%	3.2	2.8
Real-Time System Design	26%	3.2	2.5
Performance Measurement & Analysis	17%	3.2	2.7
Computer System Architecture	19%	3.2	2.7
Software Cost Estimation	24%	3.2	2.6
File Management	12%	3.2	2.8
Maintenance, Reeng. and Rev. Engg.	14%	3.1	2.7
Formal Specification Methods	29%	3.1	2.4
Comput. Complexity & Algor. Analysis	17%	3.0	2.6
Software Metrics	21%	3.0	2.5
Programming Lang. Theory	11%	3.0	2.7
Process Standards CMM / ISO 9000	19%	2.9	2.4
Parsing and Compiler Design	23%	2.8	2.3
Formal Languages	15%	2.8	2.4
Simulation	17%	2.7	2.3
Predicate Logic	15%	2.7	2.4

Table 55: Important topics (>=2.5) which the most knowledgeable people in software process consider more important than developers at large.

Tables 56, 57 and 58 list those topics that people with different educational backgrounds have found relatively more important in their careers than have others.

As Table 56 shows, those with a graduate degree in computing show an increased appreciation for process standards and software metrics.

Topic	**% increase**	**Importance for those with a CS/SE postgraduate degree**	**Importance for all participants**
Software Architecture	9%	3.7	3.4
Analysis and Design Methods	13%	3.7	3.3
Project Management	11%	3.6	3.3
Object Oriented Concepts & Tech.	8%	3.6	3.3
Giving Presentations to an Audience	8%	3.5	3.3
Config. and Release Mgmt.	11%	3.4	3.0
Leadership	8%	3.3	3.0
Systems Programming	17%	3.3	2.8
Management	8%	3.2	2.9
Process Standards CMM / ISO 9000	29%	3.1	2.4
Software Metrics	23%	3.1	2.5
Negotiation	10%	3.0	2.8
Performance Measurement & Analysis	8%	3.0	2.7
Software Cost Estimation	11%	2.8	2.6
Real-Time System Design	7%	2.7	2.5
Formal Specification Methods	11%	2.7	2.4
Marketing	18%	2.5	2.1
Entrepreneurship	6%	2.5	2.4

Table 56: Important topics (>=2.5) which those with a CS/SE postgraduate degree consider more important than do developers at large (ordered by importance)

For participants with an engineering degree (Table 57), the greatest increases in perceived importance are towards marketing, digital electronics and formal languages.

Finally, as Table 58 shows, for those with a computing degree (undergraduate or graduate) the topic with the greatest increase in importance is process standards.

Topic	**% increase**	**Importance for those with an engineering degree**	**Importance for developers at large**
Formal Languages	17%	2.8	2.4
Probability and Statistics	7%	2.7	2.6
Digital Electronics & Digital Logic	19%	2.7	2.2
Simulation	12%	2.6	2.3
Marketing	20%	2.6	2.1

Table 57: Important topics (>=2.5) which those with an engineering degree consider more important than developers at large (ordered by importance).

Topic	**% increase**	**Importance for those with a CS/SE degree**	**Importance for developers at large**
Data Structures	6%	3.8	3.6
Software Architecture	6%	3.6	3.4
Object Oriented Concepts & Tech.	6%	3.5	3.3
Analysis and Design Methods	6%	3.5	3.3
Operating Systems	6%	3.3	3.1
Systems Programming	9%	3.0	2.8
Programming Lang. Theory	7%	2.9	2.7
Software Metrics	11%	2.8	2.5
Process Stds. CMM / ISO 9000	15%	2.8	2.4
Comput. Complexity & Algor. Analysis	6%	2.7	2.6
Formal Specification Methods	7%	2.6	2.4

Table 58: Important topics (>=2.5) which those with a CS/SE degree consider more important than developers at large (ordered by importance).

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 learning, however, because if a topic has already been extensively taught then this will reduce its potential for 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.

Graphical views of the data provided in this section can be seen in Figures 6 through 8, where the length of the vertical lines extending from the bars is the on-the-job learning. Forgetting is indicated by vertical lines that extend downward into the boxes.

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 59, should be relatively easier to learn on the job that a complex topic like a branch of mathematics, or real-time design. Future researchers might consider studying this issue. Knowing 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.

Tables 59 and 60 provide two different ways of measuring high on-the-job learning: The mean difference between questions 2 and 1, as well as the percentage of people reporting a large difference. Both lists contain most of the same topics.

Rank	Topic	On-the-job learning (Q2-Q1)	Q1: Knowledge after education.	Q2: Knowledge now
1	Configuration and Release Mgmt.	2.3	0.5	2.8
2	Project Management	1.9	1.2	3.1
3	Testing, Verif. & Qual. Assurance	1.9	1.3	3.2
4	Maint., Reeng. and Rev. Engg.	1.9	0.8	2.6
5	Object Oriented Concepts & Tech.	1.8	1.4	3.2
6	Requirements Gath. & Analysis	1.8	1.4	3.1
7	Ethics and Professionalism	1.8	1.2	3.0
8	Leadership	1.7	0.8	2.5
9	HCI / User Interfaces	1.7	1.1	2.8
10	Giving Presentations to Audience	1.6	1.6	3.3
11	Process Stds. CMM / ISO 9000	1.6	0.5	2.1
12	Software Cost Estimation	1.6	0.6	2.2
13	Negotiation	1.5	0.5	2.0
14	Software Design and Patterns	1.5	1.8	3.4
15	SW Reliability & Fault Tolerance	1.5	1.0	2.5
16	Databases	1.3	1.9	3.2
17	Software Metrics	1.3	0.8	2.1
18	Analysis and Design Methods	1.3	1.9	3.2
19	Management	1.3	1.2	2.5
20	Technical Writing	1.3	1.6	3.0
21	Software Architecture	1.3	2.0	3.3
22	Performance Meas. & Analysis	1.2	1.4	2.6
23	Data Transmission and Networks	1.2	1.7	3.0
24	File Management	1.2	1.9	3.1
25	Entrepreneurship	1.1	0.5	1.6
26	Security and Cryptography	1.1	0.8	2.0
27	Real-Time System Design	1.1	1.0	2.1
28	Operating Systems	1.0	2.3	3.4

Table 59: Topics for which on-the-job learning was highest: Differences between questions 2 and 1.

Rank	Topic	Percentage reporting high on-the-job learning
1	Configuration and Release Mgmt.	54%
2	Project Management	41%
3	Testing, Verif. & Qual. Assurance	39%
4	Requirements Gath. & Analysis	36%
5	Object Oriented Concepts & Tech.	36%
6	Ethics and Professionalism	35%
7	HCI / User Interfaces	35%
8	Maintenance, Reeng. and Rev. Engg.	34%
9	Leadership	33%
10	Software Cost Estimation	32%
11	Process Stds. CMM / ISO 9000	32%
12	Giving Presentations to an Audience	31%
13	Software Design and Patterns	29%
14	SW Reliability & Fault Tolerance	29%
15	Software Metrics	29%
16	Databases	28%
17	Negotiation	27%
18	Software Architecture	24%
19	Technical Writing	24%
20	Management	23%
21	Data Transmission and Networks	23%
22	File Management	23%
23	Real-Time System Design	22%
24	Analysis and Design Methods	22%
25	Performance Measurement & Analysis	22%
26	Operating Systems	21%
27	Entrepreneurship	21%
28	Network Arch. & Data Trans.	20%

Table 60: Topics for which the most people reported very high on-the-job learning (A difference between question 2 and question 1 of 3 points or greater).

It may be considered a waste if students forget significant amounts of what they have learned following graduation. Table 61 and 62 show those topics where this has happened: Both tables show very similar sets of topics, starting with continuous mathematics, and also containing other mathematics as well as basic science and theoretical material.

It might be the case that mathematical and theoretical material might be 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 a lot of correlation.

Rank	Topic	Forgetting since education (Q2-Q1)	Q1: Knowledge after education	Q2: Knowledge Now
1	Differential Equations	-1.2	2.7	1.6
2	Differential and Integral Calculus	-1.1	3.2	2.1
3	Linear Algebra and Matrices	-0.8	3.1	2.3
4	Chemistry	-0.8	2.2	1.4
5	Physics	-0.7	2.7	2.1
6	Laplace and Fourier Transforms	-0.6	1.8	1.2
7	Probability and Statistics	-0.5	2.9	2.4
8	Combinatorics	-0.4	1.7	1.4
9	Set Theory	-0.4	2.5	2.1
10	Predicate Logic	-0.3	2.2	1.9
11	Graph Theory	-0.2	1.9	1.7
12	Control Theory	-0.2	1.2	1.1
13	Analog Electronics	-0.2	1.3	1.2
14	Formal Languages	-0.1	2.2	2.1
15	Automata theory	-0.1	1.5	1.5
16	Comput. Methods for Numeric Probs.	-0.1	2.2	2.1
17	2nd Lang. Other than English	-0.1	1.8	1.8

Table 61: Topics for which there was net forgetting of material following completion of education.

Rank	Topic	Percentage reporting high forgetting since completing their education
1	Differential Equations	38%
2	Differential and Integral Calculus	38%
3	Linear Algebra and Matrices	24%
4	Chemistry	21%
5	Physics	21%
6	Laplace and Fourier Transforms	21%
7	Probability and Statistics	17%
8	Formal Languages	12%
9	Set Theory	12%
10	Combinatorics	12%
11	Predicate Logic	12%
12	Comput. Methods for Numeric Probs.	11%

Table 62: Topics for which the most people reported a high level of forgetting since their education.

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 curriculum improvement process on the most needed topics.

Table 63 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.

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	HCI / User Interfaces	17%	2.8	3.3
9	Configuration and Release Mgmt.	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
14	Systems Programming	14%	2.5	2.9
15	Software Reliability & Fault Tolerance	14%	2.5	2.9
16	Management	15%	2.5	2.9
17	Negotiation	31%	2.0	2.9
18	Performance Meas. & Analysis	7%	2.6	2.8
19	Maintenance, Reeng. and Rev. Engg.	6%	2.6	2.8
20	Network Arch. & Data Trans.	8%	2.6	2.8
21	Software Cost Estimation	21%	2.2	2.7
22	Information Retrieval	13%	2.3	2.7
23	Real-Time System Design	21%	2.1	2.6
24	Comput. Complexity & Algor. Analysis	10%	2.3	2.6
25	Software Metrics	17%	2.1	2.6

Table 63: Topics for which there may be a need for training. Important (>2.5) topics where current knowledge lags most behind the importance of details (ordered by importance).

For Table 63, we have selected only the topics that have an overall importance greater than 2.5 (the half-way point) because trainers probably would not care about a large lag for other 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. real-time design), and we thought that this would be misleading. Trainers should pay particular attention to topics that are both near the top of the table and have the highest lag: Requirements gathering, technical writing and human-computer interaction / user interfaces.

Table 64 is structured similarly to table 63. This time, however, the comparison is between overall importance of topics and the amount 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.

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	HCI / User Interfaces	67%	1.1	3.3
5	Object Oriented Concepts & Tech.	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, Verif. & Quality Assurance	59%	1.3	3.2
11	Technical Writing	48%	1.6	3.1
12	Leadership	73%	0.8	3.0
13	Configuration and Release Mgmt.	83%	0.5	3.0
14	Data Transmission and Networks	42%	1.7	3.0
15	Management	61%	1.2	2.9
16	SW Reliability & Fault Tolerance	64%	1.0	2.8
17	Systems Programming	42%	1.6	2.8
18	Network Architecture & Data Trans.	40%	1.7	2.8
19	Negotiation	84%	0.5	2.8
20	Maintenance, Reeng. and Rev. Engg.	72%	0.8	2.7
21	Performance Meas. & Analysis	48%	1.4	2.7
22	Software Cost Estimation	77%	0.6	2.6
23	Real-Time System Design	61%	1.0	2.5
24	Information Retrieval	46%	1.4	2.5
25	Software Metrics	67%	0.8	2.5

Table 64: Data showing the need for improvement in university courses. Important (>2.5) topics where learning in educational programs lagged most behind overall importance (ordered by importance).

Table 65 shows a similar comparison to table 64, but for topics 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.

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, in fact, to be useful to computing professionals in their work, even if, according to the survey, software practitioners do not recognize it. This is particularly true for set theory as well as probability and statistics. Both 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 increased 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 in other disciplines: For example, certain medical specialists like 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.
- The knowledge 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).

Rank	Topic	% Excess	Knowledge after education (Q1)	Overall importance
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	Comput. Methods for Numeric Probs.	3%	2.2	2.1

Table 65: Topics taught relatively more than their importance might warrant.

8. Comparisons with the 1997 Survey

In this section we present a comparison between the 1997 survey (Lethbridge 1998a, 1998b) and the 1998 survey.

The topics and questions were not exactly the same between the two surveys; we therefore have attempted a best-match comparison. We have only included the 53 topics that are present in both surveys. In general, the 1998 survey covered more topics; most notably missing from the 1997 survey was 'Specific Programming Languages' – the topic considered the most important in 1998.

The first two questions asked about each topic were roughly similar between the two surveys. In 1997, however, we asked a single question about importance; this was split into two separate questions in 1998 (details vs. influence). To form a comparison we are therefore using the average of the two 1998 questions; we also used this 'overall importance' data earlier in this report.

In all of the comparison tables (Tables 66 to 71), we present both the differences in mean score and the differences in topic rank. Add the given differences to arrive at the 1998 score or rank. Differences are more interesting if they occur at the top of the tables where the differences between successive mean scores are wider.

We have highlighted the most important differences in bold. Any differences should be caused by one of the following reasons:

- Differences in questions or topic names
- Differences in the sampling approach
- Normal statistical variation

We will attempt to explain each of the significant differences according to one of these reasons.

Tables 66 and 67 compare the two surveys' responses to question 1: The amount learned in education. We attribute most of the negative differences (lower scores in 1998) to be due to a bias in 1997 towards real-time developers. The negative value for 'General Software Design and Analysis' was probably because in 1998 we split this into two separate topics (the average is given for comparison purposes).

The positive change for second language learning is probably because of the increased international coverage in 1998; and the positive change for object orientation is probably due to this material being more widely taught in recent years. The positive change for chemistry is not easy to explain, but may have to do with differences in the types of education given to international audiences.

Topic	1997 mean score	Difference with 1998 mean score	Rank difference
Differential and Integral Calculus	3.43	-0.22	0
Linear Algebra and Matrices	3.38	-0.33	0
Data Structures	3.21	-0.33	-1
Probability & Statistics	3.10	-0.20	1
Differential Equations	2.96	-0.24	-1
General software Design and Analysis	**2.88**	**-1.00**	**-11**
Set Theory	2.83	-0.33	0
Physics	2.75	-0.02	3
Programming Language Theory	2.73	-0.49	-1
Operating Systems	2.65	-0.34	2
Computer Architecture	2.64	-0.40	2
Computational Methods for Numeric Problems	2.56	-0.40	-1
Digital Electronics & Digital Logic	2.52	-0.43	-2
File & Information Management	2.43	-0.53	-2
Predicate Logic	2.43	-0.25	4
Graph Theory	2.33	-0.46	-3
Computational Complexity & Algorithm Analysis	2.28	-0.19	3
Chemistry	**2.20**	**-0.03**	**6**
Databases	2.07	-0.19	1
Parsing and Compiler Design	2.06	-0.32	-1
Systems Programming	2.03	-0.40	-4
Data Transmission And Networks	1.93	-0.21	-1
Simulation	**1.91**	**-0.70**	**-13**
Information Theory	1.87	-0.35	-2
Analog Electronics	**1.86**	**-0.55**	**-8**

Table 66: Top 25 topics according to question 1, the amount learned in formal education, considering topics found in both 1997 and 1998 surveys. Interesting differences highlighted in bold are explained in the text.

Topic	**1997 mean score**	**Difference with 1998 mean score**	**Rank difference**
Economics	1.72	0.02	4
Requirements Gathering and Analysis	1.66	-0.31	-2
Artificial Intelligence	**1.58**	**-0.37**	**-9**
Technical Writing	1.55	0.09	5
Second Language Other Than English	**1.53**	**0.30**	**10**
Information Retrieval	1.46	-0.11	1
Real Time System Design	**1.43**	**-0.44**	**-13**
Formal Methods	1.41	-0.10	1
Parallel And Distributed Processing	1.39	-0.19	-4
Computer Graphics	1.33	-0.10	1
Project Management	1.30	-0.11	-3
OO Analysis And Design	**1.26**	**0.12**	**10**
Human Computer Interaction/User Interfaces	1.24	-0.15	-3
Ethics And Professionalism	1.24	-0.02	4
Philosophy	1.24	0.12	12
Testing And Quality Assurance	1.22	0.09	10
Pattern Recognition & Image Processing	1.19	-0.27	-4
Psychology	1.15	-0.09	1
Accounting	1.06	-0.06	0
Software Metrics	0.91	-0.09	-2
Software Reliability	0.91	0.11	3
Management	0.88	0.27	7
Robotics	0.88	-0.26	-2
Maintenance, Reengineering and Rev. Engg.	0.83	-0.07	1
Software Cost Estimation	0.76	-0.17	-1
Configuration Management	0.60	-0.08	-1
Marketing	0.52	0.17	3
Process Standards	0.44	0.02	0

Table 67: Continuation of table 66; topics about which less was learned, comparing 1997 with 1998 data.

Tables 68 and 69 compare the two surveys' responses to question 2: The amount currently known. The major negative differences (lower scores in 1998) for real-time system design and process standards are probably due to the real-time bias in 1997. It would appear from the data that organizations that develop software care more about process standards, perhaps because they have more stringent reliability needs. The positive differences for databases and information retrieval are also probably due to correction of the 1997 real-time bias: In 1998 there was a more appropriate representation from MIS and data processing practitioners who apparently need databases more.

The negative difference for second language is interesting and unexplained. Whereas 1998 participants had *learned* more second languages, they *currently know* less than their 1997 counterparts.

Topic	**1997 mean score**	**Difference with 1998 mean score**	**Rank difference**
General software Design and Analysis	3.87	-0.56	-2
Data structures	3.67	0.05	1
Testing and quality assurance	3.36	-0.18	-3
Requirements gathering and Analysis	3.28	-0.15	-3
Operating systems	3.24	0.12	3
Project management	3.18	-0.09	-2
Technical writing	3.17	-0.20	-4
File & information management	3.06	0.02	-1
Data transmission and networks	3.04	-0.09	-3
OO analysis and design	3.02	0.18	5
Configuration management	3.00	-0.17	-2
Databases	**2.95**	**0.28**	**8**
Human Computer Interaction/User interfaces	2.89	-0.14	-1
Real time system design	**2.88**	**-0.80**	**-18**
Ethics and professionalism	2.88	0.12	5
Computer architecture	2.87	-0.22	0
Programming language theory	2.84	-0.14	2
Maintenance, Reengineering and Rev. Engg.	2.84	-0.20	1
Systems programming	2.83	-0.29	1
Process standards	**2.57**	**-0.49**	**-13**
Management	2.46	0.01	1
Software reliability	2.44	0.07	3
Computational complexity & algorithm analysis	2.31	0.01	0
Software cost estimation	2.30	-0.14	-2
Probability & statistics	2.30	0.14	4

Table 68: Top 25 topics according to question 2, the amount currently known, considering topics found in both 1997 and 1998 surveys. Interesting differences highlighted in bold are explained in the text.

Topic	1997 mean score	Difference with 1998 mean score	Rank difference
Second Language Other than English	**2.28**	**-0.45**	**-15**
Physics	2.26	-0.18	-7
Formal methods	2.25	0.03	4
Linear Algebra and Matrices	2.23	0.03	4
Parsing and compiler design	2.22	-0.27	-8
Parallel and Distributed Proc.	2.22	-0.08	4
Information retrieval	**2.21**	**0.12**	**10**
Software metrics	2.20	-0.06	5
Digital Electronics & Dig. Logic	2.18	-0.06	4
Simulation	2.12	-0.17	-4
Set Theory	2.08	0.06	7
Differential. and Integ. Calculus	2.03	0.05	2
Predicate logic	2.00	-0.13	-2
Computational Methods for Numeric Problems	1.90	0.21	8
Economics	1.84	0.12	3
Graph Theory	1.75	-0.10	-2
Computer Graphics	1.74	0.23	6
Information Theory	1.66	-0.05	-2
Differential Equations	1.63	-0.08	-2
Psychology	1.51	0.12	1
Accounting	1.48	0.04	-1
Philosophy	1.47	0.21	5
Artificial Intelligence	1.46	-0.11	-2
Chemistry	1.44	0.00	1
Analog Electronics	1.41	-0.21	-2
Pattern Recognition and Image Processing	1.35	-0.10	0
Marketing	1.33	0.03	3
Robotics	0.89	-0.03	0

Table 69: Continuation of table 68; topics about which less is known, comparing 1997 with 1998 data.

Tables 70 and 71 compare the two surveys' responses to the questions about importance of topics.

There are many more interesting differences between the surveys for importance than for amount learned or known. We have no clear explanation why this should be so.

The big negative difference in real-time design appears in the importance data, as it did in the current-knowledge data. As before, this is attributed to 1997 survey bias; the 1998 figures are seen as more representative. Negative changes for operating systems, data transmission, testing and possibly technical writing might also be related to a real-time bias. We also see a significant mean-score difference in 'General Software Design and Analysis' that was found in the 'amount learned' data, and attributed to a split in the topic for 1998.

Topic	**1997 Score**	**Difference with 1998 mean score**	**1998 Rank difference**
General software Design and Analysis	**4.33**	**-0.88**	**-1**
Data structures	4.05	-0.46	1
Testing and quality assurance	**3.71**	**-0.50**	**-5**
Requirements gathering and Analysis	3.69	-0.28	1
Technical writing	**3.63**	**-0.49**	**-4**
Operating systems	**3.51**	**-0.37**	**-4**
Project management	3.51	-0.26	0
Data transmission and networks	**3.50**	**-0.52**	**-5**
Real time system design	**3.35**	**-0.81**	**-15**
OO analysis and design	3.30	0.02	5
Configuration management	3.28	-0.26	-1
File & information management	3.23	-0.41	-3
HCI/User interfaces	**3.19**	**0.13**	**9**
Maint., Reeng. and Rev. Engg.	3.17	-0.44	-4
Systems programming	3.11	-0.32	-2
Databases	3.10	-0.04	5
Ethics and professionalism	**3.01**	**0.29**	**11**
Computer architecture	2.90	-0.24	-2
Management	2.84	0.07	5
Programming language theory	2.80	-0.10	1
Software reliability	2.69	0.13	5
Software cost estimation	2.62	-0.07	-1
Process standards	2.51	-0.11	-5
Computational complexity & algorithm analysis	2.41	0.17	4
Information retrieval	2.37	0.14	1

Table 70: The most important 25 topics, considering topics found in both 1997 and 1998 surveys. Interesting differences highlighted in bold are explained in the text.

Two interesting positive changes can be noted, although with little explanation: Human computer interaction /user interfaces, and ethics and professionalism were both found significantly more important in the 1998 data.

Other new positive differences appear in the importance data: These are found in business topics (marketing, economics and accounting) and mathematics topics (statistics and predicate logic). Since these differences are nearer to the bottom of the rankings, they are of less interest.

Topic	**1997 Score**	**Difference with 1998 mean score**	**1998 Rank difference**
Formal methods	2.31	0.10	0
Parallel and Distributed Proc.	2.31	-0.05	-4
Software metrics	2.22	0.27	3
Dig. Electronics & Dig. Logic	2.21	0.02	-3
Parsing and compiler design	2.19	0.09	0
Probability & statistics	**2.18**	**0.39**	**10**
Simulation	2.13	0.21	3
2nd Lang. Other than English	2.11	-0.10	-8
Predicate logic	**1.88**	**0.48**	**6**
Psychology	1.82	0.27	-1
Set Theory	1.76	0.40	3
Comput. Methods for Numeric Probs.	1.75	0.35	2
Linear Algebra and Matrices	1.62	0.42	-1
Information Theory	1.58	0.39	-4
Physics	1.56	0.43	-2
Graph Theory	1.55	0.40	-3
Marketing	**1.55**	**0.59**	**8**
Computer Graphics	1.47	0.38	-2
Economics	**1.47**	**0.59**	**6**
Accounting	**1.29**	**0.79**	**8**
Analog Electronics	1.27	0.27	-3
Differential. and Integ. Calculus	1.13	0.50	1
Philosophy	**1.09**	**0.93**	**9**
Pattern Recog. and Image Proc.	**1.04**	**0.61**	**4**
Artificial Intelligence	0.97	0.57	3
Differential Equations	0.94	0.32	2
Chemistry	0.60	0.66	4
Robotics	0.56	0.69	4

Table 71: Continuation of table 70; topics that are considered less important, comparing 1997 with 1998 data.

9. Demographics

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

Educational Backgrounds

Table 72 indicates the fields in which participants had obtained degrees, while Figure 9 describes the highest level of education they reached.

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

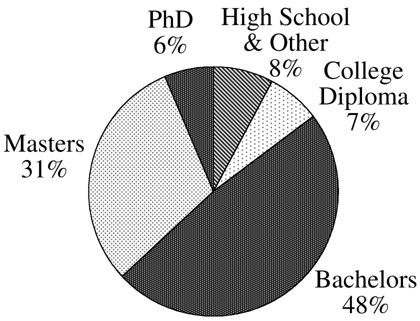

Figure 9. Highest educational level reached by the participants (170 responses).

Geographical Distribution

Table 73 and Figure 10 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.

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

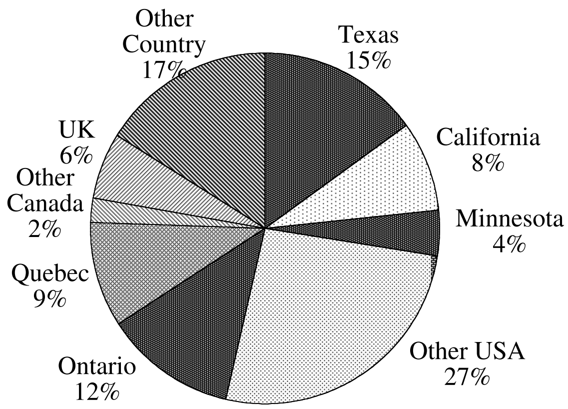

Figure 10. Work locations of the participants. Participants work in a total of 24 countries.

Type of Software Developed

Table 74 describes the types of software developed by the participants. Participants were given four categories from which they could choose one or more. Only 28% of participants just chose one of the categories.

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 74: 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.

Industry in Which Participants Work

Table 75 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.

Industry	# parti-cipants	Percent
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 75: 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.

Team Size

Table 76 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.

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 76: The sizes of teams in which participants worked.

Type of Work Performed

Figure 11 and Table 77 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 76 contains time that overlaps with some of the other categories (e.g. maintenance programming, or maintenance design).

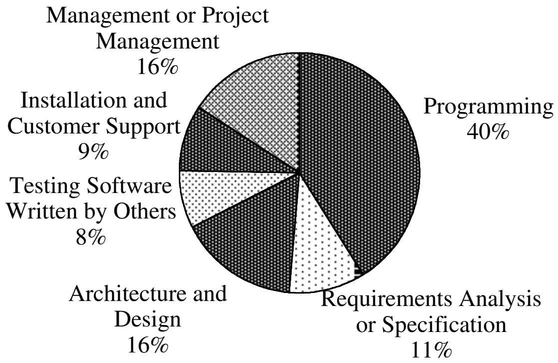

Figure 11. How the average participant distributes his or her time. These figures should be considered highly subjective, and account for no idle time, meetings, phone calls etc.

	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 77: Details of percentage of time spent by participants

Experience of Participants

Table 78 and Figure 12 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).

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%

Table 78: Graduation decades of participants in the survey.

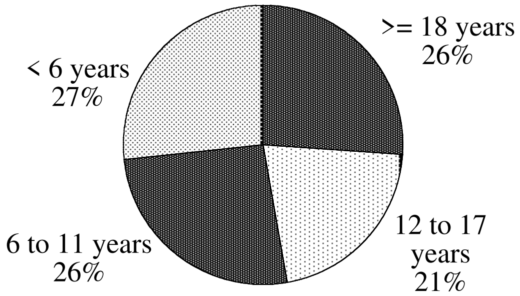

Figure 12. Experience of participants developing software: the number of years working in the software industry.

10. Conclusions

The data from the 1998 survey of software practitioners contain relatively few surprises. However, they provide concrete evidence that can be used for decision-making on the part of educators and trainers.

Among the important conclusions that can be drawn are:

• There is far more to software development than programming; however the participants in this survey put programming topics, specifically languages and data structures, at the very top when asked both 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 rank quite high.

• Mathematics, especially calculus, is extensively taught in computing programs. Participants feel 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: 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 math education is not wasted by it being subsequently forgotten.

• Many ‘skills’ or ‘soft’ topics were placed very high in this survey. Giving presentations, technical writing, as well as ethics and professionalism were given very prominent rankings. Educators should therefore increasing their emphasis to 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, information retrieval, databases and software process 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 computer science graduates in important areas such as digital signal processing, telecommunications, control systems, simulation and entrepreneurship.

• 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 data 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 report.

References

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, pp. 56-66.

Lethbridge, T.C. (1998b), "The Relevance of Software Education: A Survey and Some Recommendations", *Annals of Software Engineering*, 6, pp. 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), "Priorities for the Education and Training of Software Engineers", to be submitted, *Journal of Systems and Software.*

Lethbridge, T.C. and Singer, J. (1998), "From Work Patterns to Requirements", Submitted December 1998, *International Journal of Human-Computer Studies.*