Available online at www.sciencedirect.com
SCIENCE DIRECT®
The Journal of Systems and Software 74 (2005) 325–335
The Journal of Systems and Software
www.elsevier.com/locate/jss

An investigation of software engineering curricula

Barbara Kitchenham *, David Budgen, Pearl Brereton, Philip Woodall

Department of Computer Science, Keele University, Staffordshire ST5 5BG, UK

Received 5 November 2003; received in revised form 7 March 2004; accepted 15 March 2004
Available online 12 May 2004

Abstract

We adapted a survey instrument developed by Timothy Lethbridge to assess the extent to which the education delivered by four UK universities matches the requirements of the software industry. We propose a survey methodology that we believe addresses the research question more appropriately than the one used by Lethbridge. In particular, we suggest that restricting the scope of the survey to address the question of whether the curricula for a specific university addressed the needs of its own students, allowed us to identify an appropriate target population. However, our own survey suffered from several problems. In particular the questions used in the survey are not ideal, and the response rate was poor.

Although the poor response rate reduces the value of our results, our survey appears to confirm several of Lethbridge's observations with respect to the over-emphasis of mathematical topics and the under-emphasis on business topics. We also have a close agreement with respect to the relative importance of different software engineering topics. However the set of topics, that we found were taught far less than their importance would suggest, were quite different from the topics identified by Lethbridge.

Keywords: Software engineering curricula; Survey methods

1. Introduction

Lethbridge (1998, 2000) conducted two surveys to identify topics where software practitioners felt they needed more or better education. His initial goals (Lethbridge, 1998) were to provide

- information to educational institutions and companies as they plan curricula and training programs,
- data that will assist educators and practitioners in evaluating existing and proposed curricula.

To determine the effects of formal education, Lethbridge (1998) presented the respondents with a list of 57 topics related to software (31 topics), mathematics (9 topics), engineering (4 topics) and other concerns (13 topics). For each topic, the respondent was asked:

1. How much did you learn about this at University or College?
2. What is your current knowledge about this, considering what you have learned on the job as well as forgotten?
3. How useful has this specific material been to you in your career?
4. How useful would it be (or have been) to learn more about this (e.g. additional courses)?

The subjects were asked to reply to each question on a fully specified six-point ordinal scale. The topics were selected by examining university curricula and the initial IEEE/ACM software engineering body of knowledge. As a result of his survey Lethbridge concluded that software engineering curricula were not delivering appropriate education to practitioners and that changes to curricula were necessary. The survey was repeated later (with slightly amended questions and more topics) and similar results were obtained (Lethbridge, 2000).

We have a number of concerns about the validity of Lethbridge's surveys (see Section 2), but we believe the

* Corresponding author. Tel.: +44-178-258-3075; fax: +44-178-271-3082.
E-mail addresses: barbara@cs.keele.ac.uk, ap_kitchenham@onetel.net.uk (B. Kitchenham).

0164-1212 © 2004 Elsevier Inc. Open access under CC BY-NC-ND license.
doi:10.1016/j.jss.2004.03.016

topic he raised is very important. We have therefore attempted to replicate Lethbridge's surveys using an improved survey technique.

In Section 2, we discuss the problems with Lethbridge's survey method and the procedure we adopted to address those problems. In Section 3, we present the results of our survey. In Section 4, we contrast our results with Lethbridge's results. In Section 5, we discuss the problems and limitations of our own survey. We present our conclusions in Section 6.

2. The survey method

2.1. Population and sampling

There are a number of problems with Lethbridge's survey method but most arise because Lethbridge did not define the population to which his results would apply. He recruited participants for the surveys in two ways by approaching companies directly and asking them to participate, and by advertising for participants on the Web. Thus, Lethbridge's survey had no defined target population, nor was there any concept of sampling from a target population. He obtained a set of responses from the group of people motivated to respond. So, although he described the demographic properties of his respondents (age, highest education qualification, nationality etc.), he had no way of assessing response rate and, more importantly *formally*, no generalisation of his results is possible (Fink, 1995).

Inspection of the demographic properties of Lethbridge's respondents raises the issue of whether they were able to throw any light on the main concerns of the survey. In both surveys, some of the respondents graduated a very long time ago, and some graduated in non-computer science-related disciplines, migrating post graduation to software engineering. Thus, it seems unlikely that such respondents could offer useful information about current computer science-related curricula or training programs.

One reason it was difficult to define an appropriate subject population was because the persons expected to benefit from the survey were very general. In his first paper, Lethbridge (1998) referred to educational institutions and companies planning curricula and training programs. His later paper, Lethbridge (2000) referred to software engineering licensing bodies, companies focussing on better training for their staff, and the IEEE in its SWEBOK project (Bourque and Dupuis, 2001).

We believe that it is easier to define an appropriate subject population, if the group intended to benefit from the survey is well-defined. From our point of view as lecturers in a specific university, we want to know that our curriculum is appropriate to the needs of *our* undergraduate students. To address this question, we need to assess whether our curriculum was appropriate to the needs of our recent graduates. Thus, restricting the scope of the survey goal helped us to define the appropriate population to survey.

We used the following method to define and sample our population. We selected as our target population people who had graduated in computer science or software engineering from four English universities including, Keele University, in 1998 and 1995.[1] This population was chosen because

- it included graduates who had been in industry long enough to have experienced a reasonably wide range of software engineering jobs and tasks;
- it excluded graduates who had not studied computer science or software engineering;
- it excluded graduates who had left university so long ago that the curriculum they experienced would have changed anyway.

We were also able to make our population more homogeneous with respect to the addressing the goal of the survey, using the following exclusion criteria:

- Female students were, unfortunately, a very small proportion of graduates for three of the universities participating in the survey, so they were excluded from the population list for those three universities. This was necessary because the survey included a self-assessment section. There is evidence that males over-estimate their ability and females underestimate their ability (see for example, Beyer, 1990; Beyer and Bowden, 1997). With a small sample of female students, it would not be possible to adjust for any systematic bias introduced by gender, so we excluded them from the population list for three of the universities.
- At Keele University, most students take dual honours degrees (i.e. the equivalent of a double-major in a US university), so we excluded single honours graduates. At the other universities, most graduates took single honours (i.e. the equivalent of a single major in a US university), so we excluded joint honours graduates. Thus, we ensured that the population list for each university reflected the main student group for which the university's curriculum was designed.

From the resulting population list for each university, we selected a random sample of 60 graduates within the following blocks:

[1] We have not named the other participating universities because they wished to remain anonymous.

1. *Year of graduation i.e. 1995, or 1998*. The survey was undertaken in summer 2002, so our subjects graduated either seven or four years prior to the survey.
2. *Version of the questionnaire*. Following Lethbridge's approach we used two versions of the questionnaire with the questions in different orders. This was done to reduce question order bias.

We gave each graduate a uniquely coded questionnaire so that (theoretically) we could follow-up subjects who did not respond. Because responses were not completely anonymous, we provided a secure filing cabinet to hold the information that linked the questionnaire identifier to a particular graduate. Each of the selected graduates received a personalised letter explaining the goals of the survey and how they had been selected to take part. They were also offered the opportunity to complete a Web-based version of the questionnaire as an alternative to completing the questionnaire by hand.

2.2. Other survey method issues

2.2.1. Question format

Each of Lethbridge's two surveys was on the same topic, but Lethbridge made minor changes to the first three questions and made a major change to his last question about each topic. In the first survey (Lethbridge, 1998), question 4 was

> "How useful would it be (or have been) to learn more about this (e.g. additional courses)?"

In his second survey (Lethbridge, 2000), question 4 was

> "How much influence has learning the material had on your thinking (i.e. your approach to problems and your general 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."

In our opinion, the first version of the question was better than the second version, because the second version is more complex and thus more difficult to interpret and understand. In particular, the second version appears to be two-edged (referring both to approach to problems and to general maturity) and rather imprecise (since it may not be clear what "general maturity" really means). Thus we used the first version of the question.

Unfortunately, further reflection indicates that even the first version of the question is ambiguous. Is the respondent supposed to answer in terms of whether he would have benefited from more courses at university, or in terms of whether he would benefit from industrial courses at the present time? Even more unfortunately, we did not recognize this problem until we had reviewed some of the responses to our questionnaire.

In his first survey (Lethbridge, 1998), Lethbridge identified 57 topics in four groups. In his second survey, he included 75 topics (Lethbridge, 2000). We amended the 75 topics to 78 classified in terms of software (39 topics), hardware (10 topics), maths (14 topics) and non-computing (15 topics). In particular, we included Web-based programming, and multi-media, which Lethbridge did not include, and we split software design and patterns into two separate topics: software design practices and software design patterns.

2.2.2. Research ethics

When undertaking research involving human subjects it is necessary to consider whether participating in the research could adversely affect the subject (Singer and Vinson, 2001). In this case there are two possible adverse effects:

1. *Wasting the subject's time*. If our survey were so badly planned that we were unable to draw any valid conclusions, we should have known this in advance, and abandoned any attempt at a survey. Or, if we failed to analyse, report and act upon the results, we could certainly be accused of wasting the respondent's time.
2. *Breaching anonymity*. We did not want respondents who commented unfavourably on the courses they were taught at university to be identified. It is possible that such subjects might receive a biased report if they later requested a reference from their university. We did not want this to happen or for subjects to fear that it might happen.

In the first case, we tried to identify as many of the problems with Lethbridge's survey as we could and to put in place procedures to address those problems. In the second case we prepared an ethical statement and published it on the web. The ethical statement is shown in Appendix A to this paper. Subjects were informed of the ethical statement's web address in the covering letter accompanying the survey questionnaire, so they could check that we had employed appropriate methods for preserving their anonymity. (Note this is an additional reason for excluding female graduates from a target population. If they represent only a very small proportion of a target population, it is possible that an analysis of responses from females might allow individuals to be identified.)

2.2.3. Analysis method

Each of Lethbridge's four main questions has its own associated ordinal scale with responses defined in the

context of the question. For instance, question 1 "How much did you learn about this at your university or college" had the following fully-defined ordinal scale:

Score	Definition
0	Learned nothing at all
1	Became vaguely familiar
2	Learned the basics
3	Became functional (moderate working knowledge)
4	Learned a lot
5	Learned in depth, became expert (learned almost everything)

Instead of using Lethbridge's six-point scale from 0 to 5, we used a scale from 1 to 6. Since the numerical values represent an ordinal scale, both scales are equivalent. We prefer to use 1–6 to emphasise that the underlying ordinal scale has six-scale points.

Lethbridge (2000) constructed two compound measures from the responses for each topic area:

$$\text{Overall importance}^2 = (Q3_i + Q4_i)/2, \quad (1)$$

$$\text{learned (forgotten) since education} = Q2_i - Q1_i, \quad (2)$$

where Qj_i is the numerical value of subject i's ordinal scale response to question j. In his earlier paper, Lethbridge (1998), used the second measure (i.e. $Q2 - Q1$) but did not take the average of questions 3 and 4. He noted that question 3 was considered a better indicator of usefulness than question 4.

Lethbridge (2000) also defined the current knowledge gap as the difference between the importance of the topic and the amount currently known:

$$\text{Knowledge gap} = Q2_i - \text{Overall importance}. \quad (3)$$

There are two concerns with this method of analysis:

1. The construction of the *overall importance*, *learned since education* and *knowledge gap* variables violates the mathematical restrictions implied by an ordinal scale (Fenton and Pfleeger, 1996). Although researchers frequently treat ordinal scale measures as if they are interval or ratio scale measures, it is preferable to avoid violating scale type restrictions if possible.
2. The results are based on the extent to which education suited the individual rather than whether the education received was reasonable for the cohort as a whole. Now, it is not possible for education to prepare each individual perfectly for his future employment because each person's job will be different. Thus, it we believe that it is preferable to assess whether the education was suitable for the group as a whole rather than the individual. Thus, we analysed our data from the viewpoint of the cohort's response to each topic rather than the viewpoint of the individual.

As an alternative method of assessing importance, we based our assessment on the proportion of subjects scoring three (indicating moderate usefulness) or more for question 3 (recall that our scale went from 1 to 6):

$$\text{Importance} = \text{proportion of subjects scoring three or more on } Q3/\text{number of subjects}. \quad (4)$$

Using the same approach for $Q1$, we measured the proportion of the cohort given a moderate or better education in each of the topic areas:

$$\text{Educational provision} = \text{proportion of subjects scoring three or more on } Q1/\text{number of subjects}. \quad (5)$$

Comparison of the proportions for each question let us see the extent to which the different topics were learned at university corresponds with their relative importance.

We defined the knowledge gap to be the difference between importance and educational provision:

$$\text{Knowledge gap} = \text{importance} - \text{educational provision}. \quad (6)$$

Thus, our view of a knowledge gap refers to a gap between formal education and topic importance, whereas Lethbridge's view of a knowledge gap refers to a gap between current knowledge and importance.

3. Results

3.1. Response rates

As shown in Table 1 the response rates were generally poor. Although follow-up actions were planned, they did not take place because the research interns assigned to the task were only available for a limited time during the summer vacation.

Table 1
Survey response rate

University	Responses	Sample	Response rate (%)
A	8	60	13.33
Keele	12	60	20
C	6	60	10.00
D	4	60	6.67
Total	30	240	12.50

University D included female students in its target population list, and three of the four replies were from females.

[2] There is a discrepancy between the description of the importance measure, as the average of Q3 and Q4 and the headings given in Tables 2–4 which suggest than the sum not the average was used (Lethbridge 2000). Inspection of the values for importance shown in Fig. 2 of Lethbridge (2000), confirms that the average not the sum was used.

It is important to stress, that we had originally planned to analyse the replies for each university separately. In Great Britain, each university department controls its own curricula, so it is likely that curricula from different universities will differ. However, with the extremely limited response, we decided to test whether the results could be analysed together.

We did this by comparing Keele University results with the combined results from the other universities. We did this because a priori there would seem to be a case for assuming that Keele, which concentrates on dual honours, would be somewhat different from the other universities.

As described in Section 2.2.3, we calculated the proportion of favourable responses for each question (i.e. proportion of subjects scoring three or better compared with the total number of responses). We then compared the rank order for each question (for the software engineering topics only) obtained from the Keele responses with the rank order for the other universities (see Table 2). Since all the rank order correlations were high (>0.7), we concluded that it was reasonable to analyse all the data together. [3]

Table 2
Correlation between the rank order of responses for Keele university compared with the other universities

$Q1$ correlation	$Q2$ correlation	$Q3$ correlation	$Q4$ correlation
0.83	0.78	0.86	0.81

3.2. Implication for curriculum design

Given the low response rate the following analysis must be treated with caution. However, it does illustrate how to analyse survey data from a cohort viewpoint with minimal violation of measurement scales.

The percentage of replies of three or more for each topic are shown in Table 3 for software engineering topics, Table 5 for hardware engineering topics, Table 6 for mathematical topics and Table 7 for other topics. The topics are ordered on $Q3$: "How useful has this specific material been to you in your career?".

We assume that the response to $Q1$: "How much did you learn about this topic in your formal education?" is related to the extent to which the topic is represented in the curriculum. Making this assumption, we can see from Table 3 that some topics seem to be under-represented on the curriculum relative to their importance, some are over-represented and some are represented in line with their importance.

This is easier to appreciate if we subtract the $Q3$ proportion from the $Q1$ proportion to construct a measure of the knowledge gap and order by the difference. This analysis is shown in Table 4, which also indicates the rank order of importance. This shows that the topics that are most seriously under-represented are Web-based programming, project management, and configuration and release management. Of these three topics, we might consider project management the most important topic for curriculum changes because it is also a topic that subjects judged to be of major importance in their jobs. Web-based programming and configuration management were judged to be of average importance, suggesting they are relevant to specific jobs rather than more generally. These topics might be candidates for industry-based courses. Topics that are substantially over-taught are parsing and compiler design, artificial intelligence, and formal specification methods. All these topics are also considered relatively unimportant in industry, so are candidates for curriculum changes.

Table 5 shows the results for hardware engineering topics. It suggests that networks, architecture, and telecommunications are quite important topics for software engineers. Furthermore, telecommunications is rather under-represented in the curriculum. Digital electronics and digital logic, however, is over-represented in the curriculum.

Table 6 shows the results for mathematical topics. It suggests that mathematical topics are not of much importance in the software industry and are, therefore, considerably over-represented in the curriculum.

Table 7 shows the results for non-computing topics. It indicates that general business skills such as giving presentations, management, leadership, ethics, and negotiation are important in industry and are all under-represented in the curriculum. Other academic topics are generally not considered very important, nor is much time allocated to them.

4. Comparison of results

In spite of the methodological difference, the results of our survey show some similarity to the results of Lethbridge's surveys. Like Lethbridge we found that

- mathematical topics were not very important to software engineers and appear to be taught more extensively than is required;
- general business topics are quite important, but are not taught in proportion to their importance, in particular, management, giving presentations, leadership, and negotiating.

We do not mean to suggest that mathematical topics are irrelevant to software engineers. Recently, in September 2003, Communications of the ACM published

[3] This is a purely heuristic procedure. A poor correlation would certainly confirm that the data should not be pooled but there is no minimum correlation coefficient level that confirms that data can be pooled.

Table 3
Percentage of responses of three or more for each software engineering topic ordered by importance (*Q*3)

Topic	*Q*3 (importance), %	*Q*1 (educational provision), %	*Q*2 (current knowledge), %	*Q*4 (usefulness of extra training), %
Human computer interaction/user interfaces	93.33	70.00	96.67	76.67
Project management	83.33	40.00	73.33	86.67
Databases	76.67	76.67	76.67	80.00
Operating systems	75.86	56.67	83.33	75.86
Requirements gathering and analysis	73.33	63.33	80.00	76.67
Specific programming languages	73.33	73.33	83.33	60.00
Data structures	73.33	70.00	80.00	50.00
Software architecture	70.00	70.00	80.00	66.67
Data transmission and networks	70.00	53.33	66.67	63.33
Analysis and design methods	66.67	76.67	73.33	66.67
Testing, verification and quality assurance	66.67	60.00	86.67	66.67
Software design practices	65.52	65.52	72.41	72.41
Web-based programming	60.71	10.00	43.33	83.33
Object oriented concepts and terminology	60.00	80.00	66.67	73.33
Systems programming	58.62	36.67	56.67	48.28
Information retrieval	57.14	44.83	58.62	57.14
Software design patterns	51.72	48.28	51.72	70.00
Configuration and release management	48.28	6.67	56.67	66.67
File management	46.67	36.67	80.00	36.67
Security and cryptography	46.43	20.00	36.67	56.67
Design of algorithms	44.83	56.67	50.00	37.93
Performance measurement and analysis	43.33	23.33	36.67	37.93
Computer graphics	39.29	13.33	33.33	34.48
Programming language theory	37.93	56.67	46.67	31.03
Multi-media	34.48	6.67	36.67	48.28
Maintenance, reengineering and reverse engineering	33.33	20.00	53.33	56.67
Formal specification methods	33.33	73.33	43.33	36.67
Software cost estimation	28.57	10.00	30.00	56.67
Software reliability and fault tolerance	27.59	10.00	33.33	41.38
Parallel and distributed processing	27.59	26.67	26.67	30.00
Real-time system design	23.33	16.67	30.00	26.67
Computational complexity and algorithm analysis	20.69	33.33	23.33	24.14
Parsing and compiler design	20.69	56.67	23.33	20.69
Process standards (CMM/ISO 9000 etc.)	17.24	10.00	20.00	35.71
Simulation	11.11	6.67	10.00	14.29
Computational methods for numerical problems	10.71	23.33	23.33	14.29
Artificial intelligence	10.00	46.67	16.67	16.67
Software metrics	6.90	10.00	13.33	13.33
Pattern recognition and image processing	6.90	13.33	6.67	10.00

several papers with a covering editorial (Devlin, 2000) by computer science academics arguing for the inclusion of mathematics in computer science curricula. Commenting on Lethbridge's results (Lethbridge, 1998), Hendersen (2000) noted that "surveys of current practice reflect reality; many software engineers have not been taught to use discrete mathematics and logic as effective tools." He suggested education was the means of ensuring "future software engineers are able to use mathematics and logic as powerful tools for reasoning and thinking." If he is correct, we can only conclude that education has failed in the past to make the link between mathematics and software engineering practice clear to students. This implies that educationalists need to rethink the ways in which mathematics is taught to software engineering undergraduates.

Table 8 compares the top 10 software engineering topics we found, with the top 10 Lethbridge found (2000). Seven topics appear in both top 10 lists although in all but one case the relative importance has changed. Three topics appeared in our top 10 but not in Lethbridge's: databases, operating systems and data transmissions and networks. Three topics appeared in Lethbridge's top 10 but not ours: OO concepts (which our survey suggested was being taught slightly more than required), testing, verification and QA, and software design and patterns (which, in our survey, both appeared to be taught at a level commensurate with their importance).

Table 4
Appropriateness of curriculum to industry

Topic	Importance rank order	Knowledge gap ($Q1 - Q3$)
Web-based programming	13	−50.71
Project management	2	−43.33
Configuration and release management	18	−41.61
Multi-media	25	−27.82
Security and cryptography	20	−26.43
Computer graphics	23	−25.95
Human computer interaction/user interfaces	1	−23.33
Systems programming	15	−21.95
Performance measurement and analysis	22	−20.00
Operating systems	4	−19.20
Software cost estimation	28	−18.57
Software reliability and fault tolerance	29	−17.59
Data transmission and networks	8	−16.67
Maintenance, reengineering and reverse engineering	26	−13.33
Information retrieval	16	−12.32
Requirements gathering and analysis	5	−10.00
File management	19	−10.00
Process standards (CMM/ISO 9000 etc.)	34	−7.24
Real-time system design	31	−6.67
Testing, verification and quality assurance	10	−6.67
Simulation	35	−4.44
Software design patterns	17	−3.45
Data structures	6	−3.33
Parallel and distributed processing	30	−0.92
Databases	3	0.00
Specific programming languages	7	0.00
Software architecture	9	0.00
Software design practices	12	0.00
Software metrics	38	3.10
Pattern recognition and image processing	39	6.44
Analysis and design methods	11	10.00
Design of algorithms	21	11.84
Computational methods for numerical problems	36	12.62
Computational complexity and algorithm analysis	32	12.64
Programming language theory	24	18.74
Object oriented concepts and terminology	14	20.00
Parsing and compiler design	33	35.98
Artificial intelligence	37	36.67
Formal specification methods	27	40.00

Table 5
Percentage of responses of three or more for each hardware engineering topic ordered by importance

Topic	$Q3$ (%)	$Q1$ (%)	$Q2$ (%)	$Q4$ (%)
Network architecture and data transmission	55.56	46.67	63.33	60.00
Computer system architecture	40.74	43.33	66.67	41.38
Telephony and telecommunications	40.00	20.00	40.00	37.93
Microprocessor architecture	12.50	40.00	30.00	25.00
Data acquisition	11.54	3.33	10.00	24.14
Digital signal processing	8.70	10.00	3.33	17.86
Digital electronics and digital logic	8.00	40.00	33.33	17.86
Analogue electronics	8.00	16.67	6.67	10.71
Robotics	4.55	3.33	6.67	10.34
VLSI	0.00	3.33	3.33	4.00

It is not possible to determine the 10 least useful software engineering topics from Lethbridge's reports but the low rating his respondents gave to AI and pattern recognition/image processing is consistent with our results.

The major difference between our survey and Lethbridge's survey is the set of software engineering topics for which a knowledge gap appears to exist. This is shown in Table 9. There is no commonality at all between the two lists, although it should be noted that

Table 6
Percentage of responses of three or more for each mathematics topic ordered by importance

Topic	Q3 (%)	Q1 (%)	Q2 (%)	Q4 (%)
Predicate logic	28.57	66.67	43.33	10.71
Set theory	28.57	60.00	40.00	32.14
Probability and statistics	20.69	66.67	56.67	21.43
Linear algebra and matrices	17.86	40.00	40.00	17.86
Graph theory	14.29	30.00	26.67	14.29
Information theory	11.11	20.00	13.33	14.81
Differential equations	10.71	30.00	30.00	14.29
Automata theory	7.14	16.67	10.00	14.29
Formal languages	7.14	53.33	26.67	14.29
Laplace and Fourier transforms	7.14	16.67	10.00	10.71
Queuing theory	3.70	23.33	13.33	14.81
Combinatorics	3.70	10.00	0.00	10.71
Control theory	3.70	3.33	0.00	14.81
Differential and integral calculus	3.57	26.67	30.00	10.71

Table 7
Percentage of responses of three or more for each non-computing topics ordered by importance

Topic	Q3 (%)	Q1 (%)	Q2 (%)	Q4 (%)
Giving presentations to an audience	80.00	53.33	93.33	73.33
Management	62.96	27.59	60.00	75.86
Leadership	62.96	16.67	66.67	72.41
Ethics and professionalism	55.56	25.00	63.33	55.56
Negotiation	51.85	17.86	50.00	74.07
Technical writing	43.33	27.59	56.67	62.07
Accounting	33.33	24.14	33.33	28.57
Marketing	29.63	13.79	33.33	42.86
Entrepreneurship	16.00	0.00	16.67	25.93
Second language	16.00	11.54	28.57	37.04
Physics	11.54	7.14	33.33	10.71
Chemistry	11.54	3.57	20.00	10.71
Economics	11.54	20.69	30.00	17.86
Psychology	7.69	17.24	16.67	14.29
Philosophy	7.69	7.14	16.67	11.11

Table 8
Top 10 software engineering and computer science topics

Lethbridge survey	Our survey
Specific programming languages	HCI/user interfaces
Data structures	Project management
Software design and patterns	Databases
Software architecture	Operating systems
Requirements gathering/analysis	Requirements gathering/analysis
OO concepts	Specific programming languages
HCI/user interface	Data structures
Analysis and design methods	Software architecture
Project management	Data transmission and networks
Testing, verification and QA	Analysis and design methods

Table 9
Software engineering topics with the greatest knowledge gap

Lethbridge's survey	Our survey
HCI/user interfaces	Web-based programming
Real-time system design	Project management
Software cost estimation	Configuration and release management
Software metrics	Multi-media
Software reliability and fault tolerance	Security and cryptography
Requirements gathering and analysis	Computer graphics

Web-based programming and multi-media were topics we included in our list of topics that were not in Lethbridge's list. This difference may be due to differences between our view of a knowledge gap and Lethbridge's view. However, it is more likely that the respondents in Lethbridge's survey, who had been in industry longer than the graduates in our survey, had experienced more dated curricula than had our respondents. Whatever the reason, the difference suggests that the balance of topics need to be evaluated frequently to keep in line with industry needs.

Although the main focus of this paper is to compare our survey methodology and results with those of Lethbridge, it is nonetheless interesting to compare our results with the structure being proposed for an under-

Table 10
Comparison of top survey topics and SEEK ratings

Topic	SEEK category	SEEK level (essential, desirable, optional)
Analysis and design methods	c (a)	Essential
Databases	c	Essential
Data structures	a	Essential
Data transmission and networks	c	Essential
HCI/user interface	a	Essential
OO concepts	a	Essential
Operating systems	c	Essential
Project management	c (a)	Essential
Requirements gathering/analysis	c	Essential
Software architecture	k	Essential
Software design and patterns	a	Essential
Specific programming languages	c	Essential
Testing, verification and QA	a	Essential

graduate Software Engineering curriculum, by the ACM and IEEE. A key element of this curriculum is the description that it provides of the goals for a programme in terms of the software engineering education knowledge (SEEK). The SEEK identifies relevant curriculum topics, how essential they are to a programme, and the level of knowledge that a graduate should have about each topic (Diaz-Herrara and Hilburn, 2003).

Table 10 shows the ratings of the joint set of topics from Table 8, using Bloom's taxonomy to rank the level of capability expected of a graduate (Bloom, 1956):

- k *knowledge* at the level of remembering past material;
- c *comprehension* involving understanding information and grasp meaning of material presented;
- a *application* as the ability to use the learned material in new and concrete situations.

Where we have put an (a) in parenthesis, this is to indicate that the item appears in different elements of the SEEK, with the more specialized elements being at the applicable level. Overall, the survey results and the SEEK classifications show good agreement, with all the topics rated as important in industry being rated as essential by SEEK, and all but one rated as requiring a capability in excess of simple knowledge of the topic. Only Software Architecture is rated as (k) in the SEEK classification, and this is primarily because of the nature of the topic.

5. Survey weaknesses and limitations

The major weakness in this survey is the poor response rate. This means that there is a serious risk that the people who responded are not representative of the target population. In particular, we may have found similarities between our survey and Lethbridge's survey because volunteers have similarities, not because the phenomena investigated in the survey are stable (Rosnow and Rosenthal, 1997).

If the survey were to be repeated it is important that the response rate be improved. For example planned follow-up procedures must take place. In addition, the survey as a whole needs to be reviewed to assess whether there are any amendments that can be made to improve the response rate such as reducing its length.

We noted earlier that question 4 is ambiguous. If universities are surveying their graduates then the question should be phrased as

> "How useful would it have been to learn more about this at university/college (e.g. additional courses)?"

Alternatively, if a company is surveying employees to determine their training needs the question should be phrased as

> "How useful would it be to learn more about this (e.g. additional training courses)?"

Lethbridge intended his survey to address the needs of industry training courses and university curriculum development. However, we believe that the joint goal has made the survey unnecessarily complicated. For a university survey, question 2 "what is your current knowledge about this considering what you have learned on the job as well as forgotten?" does not seem to offer any useful information. Question 2 would be of more importance in a company survey, where in contrast question 1 "how much did you learn about this at University or College?" is much less important.

In addition, question 1 is not easy to interpret. Respondents are asked to judge "how much did you learn about this topic in your University or College?" For curriculum development we need to know how

much exposure a topic received in the curriculum, which is not the same thing as how much a student learned. The amount a student learns is a complicated mix of

- the extent to which the topic is taught,
- the ability of the lecturer(s),
- the ability and motivation of the individual student.

This implies question 1 needs to be reworded to focus on the extent to which the topic is taught. However, logically, a university ought to be able to assess the extent of exposure from its own timetable, so question 1 may be completely unnecessary for a survey undertaken by a single university.

We have suggested a survey procedure intended to allow individual universities to evaluate their own curricula. This is a very limited goal compared with Lethbridge's goal, but it does allow us to identify a population of subjects who are in a position to provide information related to this goal. We do not suggest that our procedure scales up to answer more general questions. If researchers need to assess the specific set of knowledge-levels proposed by the working group that developed the SEEK (Diaz-Herrara and Hilburn, 2003), then we believe that a different target population is required. In this case, a more appropriate target population would be members of the IEEE Computer Society and the ACM. [4] Furthermore, we would suggest placing a time frame on requested answers, i.e. asking people what topic were important in the previous 4/5 years, not over their entire career.

6. Conclusions

In this paper we have adapted a survey instrument developed by Timothy Lethbridge to assess the extent to which the education delivered by four UK universities matches the requirements of the software industry, as experienced by graduates of the respective universities. This paper proposes a survey methodology that we believe addresses the research question more appropriately than the one used by Lethbridge. However, we note that our own survey is not without flaws. In particular the questions used in the survey are not ideal, and the response rate was extremely poor, probably in part because planned follow-up activities did not take place.

An important distinction between our survey and Lethbridge's survey is that we were interested in assessing the appropriateness of our own curricula for the needs of our own students. Using our survey method (with better follow-up procedures and simpler questions), any university can assess its own curricula but the results cannot be easily generalised to generic curricula.

Our survey analysis is mainly intended to demonstrate how to analyse the survey data for cohorts while minimising any violation of mathematical scales. However, assuming we are not affected too much by the problem that it is dominated by volunteers, our results appear to confirm several of Lethbridge's observations with respect to the over-emphasis of mathematical topics and the under-emphasis on business topics. We also have a close agreement with respect to the relative importance of different software engineering topics. We diverge from Lethbridge with respect to topics that have a large knowledge gap. The divergence supports our view that for curricula to remain in step with the changing needs of industry, surveys of this kind need to take place on a regular basis to reflect the rapid changes we find in software technology.

With respect to curriculum changes, our results suggest either that the balance between practical software engineering topics and mathematically-based computer science topics needs to be revised, or that teaching methods need to be rethought. With respect to business topics however, we do not believe that it is the role of computer science departments to provide training in general business skills. Such skills are required by students in all disciplines, not just computer science students. A better option is to provide general training in communication skills to all students. This approach is one that is currently being adopted at Keele University.

Acknowledgements

We would like to thank Ben Burch and Ray Seys for organising the questionnaires and creating the Web-based questionnaire.

Appendix A. Statement about ethical issues concerning the project

In this project we are planning to collect responses to a survey asking questions about how useful the topics taught in their degrees had been in their job, from graduates of four computer science departments in England. This document summarises the ethical issues that we believe are relevant to the project and states the rules that will govern our handling of the data.

A.1. Rationale

While the provision of any responses will of course be purely voluntary, we believe that as researchers, it is our duty to ensure that there are no adverse consequences for those taking part. This means that we need to ensure that situations cannot occur which could potentially

[4] Joint IEEE Computer Society and ACM Steering Committee for the establishment of software engineering as a profession.

lead to any form of victimisation of subjects. Such situations might involve

- a person who has made adverse comments about their course subsequently asking their former department for a reference;
- a person who has made adverse comments about their employer being identifiable in any way.

A.2. The data to be gathered

The data to be gathered in the survey will be concerned solely with the applicability of their degree programme material to their work. In selecting sample candidates from each site, we will seek to obtain a mix of gender, ethnic origin and degree classification, based upon departmental records.

A.3. Subject confidentiality

Data will be collected on the basis of anonymous submission. Forms will be numbered (all data collection will be paper-based) and the key identifying subject and response will at all times be kept confidential by the members of the project team and used only for confirming the pattern of responses. We will also include a statement to that effect within the questionnaire itself. All data will be kept in a secure file at Keele University.

A.4. Publication restrictions

No results will be published in any form that would enable respondents to be identified. In some cases, that may mean that small groups may need to be omitted from the results if the size of the group makes it likely that individuals could be identified. Any information obtained will only be published with the agreement of the participating institutions.

A.5. Further access

Since the responses themselves will form a valuable source of data for further study should the opportunity to extend the sample arise, these will be retained at the end of the project, but will not be publicly available. Any information about the keys that would enable responses to be linked to individuals will be destroyed.

References

Beyer, S., 1990. Gender differences in the accuracy of self-evaluations of performance. J. Pers. Soc. Psychol. 59, 960–970.

Beyer, S., Bowden, E.M., 1997. Gender differences in self perceptions: convergent evidence from three measures of accuracy and boas. Pers. Soc. Psychol. B 23, 157–172.

Bloom, B.S. (Ed.), 1956. Taxonomy of Educational Objectives: the Classification of Educational Goals. Handbook 1, Cognitive Domain. Longmans, New York (See also www.officeport.com/edu/blooms.htm, viewed 18/02/03).

Bourque, P., Dupuis, R. (Eds.), 2001. Guide to the Software Engineering Body of Knowledge, SWEBOK. IEEE Computer Society Press, Silver Spring, MD.

Devlin, K., 2000. Why CS students need math. CACM 46 (9), 37–39.

Diaz-Herrara, J., Hilburn, T. (Eds.), 2003. Computing curriculum-software engineering (CCSE). Public Draft 3.1, February 6, 2004. Available from <http://sites.computer.org/ccse>.

Fenton, N.E., Pfleeger, S.L., 1996. Software Metrics. A Rigorous & Practical Approach, second ed. International Thomson Computer Press.

Fink, A., 1995. The Survey Handbook. Sage Publications, Beverley Hills, CA.

Hendersen, P.B., 2000. Mathematical reasoning in software engineering education. CACM 46 (9), 45–50.

Lethbridge, T., 1998. A survey of the relevance of computer science and software engineering education. In: Proc. 11th International Conference on Software Engineering. IEEE Computer Society Press, Silver Spring, MD, pp. 56–66.

Lethbridge, T., 2000. What knowledge is important to a software professional. IEEE Comput. (May), 44–50.

Rosnow, R.R., Rosenthal, R., 1997. People Studying People. Artefacts and Ethics in Behavioural Research. W.H. Freeman and Company, New York.

Singer, J., Vinson, N., 2001. Why and how research ethics matter to you. Yes You! Empirical Software Eng. 6 (4), 287–290.