Software: university courses versus workplace practice

Janet Liebenberg, Magda Huisman and Elsa Mentz

Abstract: *There is a shortage of software developers with the right skills and knowledge, not only in South Africa but worldwide. Despite reports of a gap between industry needs and software education, the gap has mostly been explored in developed countries and in quantitative studies. This paper reports on a mixed methods study of the perceptions of professional software developers regarding what topics they learned from their formal education and the importance of these topics to their actual work. The analysis suggests that there is a gap between software development education and workplace practice and recommendations for software development education are made.*

Keywords: *computing curricula; computing education institutions; software development education; software industry; software professionals*

The authors are with the North-West University, Private Bag X6000, Potchefstroom, 2520, South Africa. Janet Liebenberg (corresponding author) is a Lecturer and Magda Huisman is a Professor with the Department of Computer Science and Information Systems. Elsa Mentz is a Professor with the Faculty of Education Sciences. Corresponding author E-mail: janet.liebenberg@nwu.ac.za.

South Africa has a shortage of software developers; and across the African continent the overall levels of technical and soft skills needed are not sufficient to meet demand (Biztech Africa, 2013; Harris, 2012). This skills shortage is actually a worldwide phenomenon (Connolly, 2013). Not only are there shortages of software development (SD) workers, but students in the computing fields are graduating with a lack of the skills that companies require (Bateman, 2013). Regarding ICT skills in South Africa, according to Mawson (2010), 'not all graduates are prepared for the working environment, and don't always fit in. This is a gap that needs addressing.'

The SD industry expects students to be educated in courses and projects that are professionally relevant and that prepare them well for the workplace (Moreno *et al*, 2012); but, equally, the role of the university should also be considered. According to Reisman (2004), the mission of undergraduate and graduate programmes is educational, but that mission has steadily transformed into one of training. The curriculum focuses on a common body of knowledge, a set of basic topics and principles: the primary qualification of new employees is their foundational undergraduate education, and they could therefore easily be trained by their employer for entry-level project work using particular (vendor) products.

Information technology is described by some authors as the 'Great Globalizer' and, in these authors' view, computing education should meet global standards. However, when referring to developing countries, some authors argue that instead of tailoring and evaluating education to global standards, computing education should address local needs (Ezer, 2006; Wade, 2002; Mooketsi and Chigona, 2014).

Several researchers have studied the knowledge and skills requirements for IT professionals through quantitative analysis, but the gap has mostly been explored in developed countries. No mixed methods study in a developing country could be found that studied professional software developers' perceptions regarding the topics they learned in their formal education and the importance of these topics to their actual work. In view of the rapid pace at which technology is changing, and in the light of the shortage of skilled software developers, the present study investigated the possible current gap between software development education and software workplace practice.

The research questions were:

(1) What are the topics professional software developers learned in their formal education?;
(2) What topics are important to professional software developers in their actual workplace?; and
(3) Is there a gap between software development education and the workplace from the perspective of the software industry?

It is hoped that the answers might help SD educators, curriculum developers and corporate trainers, especially in developing countries, to form a picture of what knowledge and skills demanded by industry university courses do or do not cater for; and the software industry will be able to ascertain which tasks new recruits in the SD workplace might be well qualified to perform. University and industry should be able relate the results to the academic preparation of future software developers, as well as the continued education and training of software developers. It is argued that this could therefore lead to more relevant software development education with regard to the software industry and contribute to meeting the demand for skilled software developers.

Clarification of terminology

The dynamic nature and continual evolution of computing makes it difficult and potentially misleading to define the computing disciplines and related terminology (Guzdial *et al*, 2009; Gruner, 2014). For the purpose of this study, it is therefore useful and helpful to explain and clarify certain key terms, as follows.

Software development (SD)

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) define 'developer' as an individual or organization that performs development activities (including requirements analysis, design, testing through acceptance) during the system or software life cycle process (ISO/IEC 25000, 2014). The IEEE, ISO and IEC define the software development process as the process by which user needs are translated into a software product (ISO/IEC/IEEE 24765, 2010). For the purposes of this study 'software development' will refer to the process of developing software products through successive phases in an orderly way.

Software industry

The ISO and IEC define 'software manufacturer' as a group of people who or organizations that develops or develop software, typically for distribution and use by other people or organizations (ISO/IEC 19770–2, 2009). For the purposes of this study, 'software industry' will refer to software manufacturers, as well as organizations for which software is not the main product but is developed for use within the organization.

Conceptual framework

The student in the software development class

Students in current university classes are described by a great number of writers as the Net Generation. The Net Generation is characterized by people who may have never known life without the Internet (Jones *et al*, 2010). However, not all today's students can be described as the Net Generation, since not all students had and still have the benefit of state-of-the-art, ubiquitous technology. They may have information literacy characteristics and IT skills quite different from those typical of the Net Generation. Higher education comprises a highly diverse and growing student body with a wide variety of information literacy skills and abilities (Lorenzo *et al*, 2007; Jones *et al*, 2010).

Students in developing countries do not fit the description of the Net Generation since Internet penetration for households in 2013 is a mere 31.2%. South Africa was ranked 37th amongst developing countries, with 39.4% (25.5% in 2012) of South African households using the Internet. The considerable rise in Internet use is explained by mobile broadband subscriptions experiencing an 80% year-on-year growth in Africa (UN Broadband Commission, 2013, 2014) and Calitz (Biztech Africa, 2013) believes that mobile applications can play a key role in delivering ICT learning and generating interest in careers in ICT in Africa.

Software developers in the workplace

Software and technical developments have made remarkable strides in the last few decades, and continue to do so seemingly unabated. The result has been

profound transformation of markets, industries and society in general (Biztech Africa, 2013; Shaw *et al*, 2005). The demands on software developers are changing because the character of software production itself is changing and with this the dependence on software is increasing (Saiedian, 2009; Shaw *et al*, 2005; Gupta, 2005). The diversity of software applications, clients and contexts requires adaptability in responding to client needs and the ability to discriminate between criteria for success (Shaw *et al*, 2005; Gupta, 2005). Professionals in the computing field require personal skills, technical expertise and lifelong learning abilities (Fernandez-Sanz, 2009; Calitz, 2010) and employers must provide opportunities for lifelong learning specifically making use of on-line courses and training materials (Krakovsky, 2010).

In view of the discriminatory apartheid past of South Africa, the Employment Equity Act (South Africa) (1998) requires companies to ensure through affirmative action that designated groups (black people, women and people with disabilities) have equal opportunities in the workplace. This is a double concern for ICT companies in South Africa because there is a shortage of black ICT professionals (Calitz, 2010) and, on top of that, the shortage of women in the computing disciplines is a fact of life in South Africa as much as it is a worldwide phenomenon (Harris, 2012).

University courses vs workplace practice

The seminal study by Lethbridge (2000) can serve as a framework for understanding relationships between the needs of the SD industry and the education of software developers. In the survey carried out by Lethbridge (*ibid*) over 200 software developers and managers from around the world were asked what they thought about 75 educational topics. For each topic they were asked how much they had learned about it in their formal education, how much they knew about it at the time of the survey and how important the topic had been in their career. The Lethbridge study and several others, mainly in the USA, suggested the existence of a gap between the knowledge and skills demanded by the industry and the knowledge and skills gained by graduates of university computing courses: Table 1 provides a summary of the relevant studies.

Courses with a primary emphasis on current technology, in which most of the knowledge will become obsolete as the technology does, are a major challenge in the education of software developers. Pressures arising from the changing character of software and from external pressures on educational institutions will require changes in what software developers are taught and how they are taught (Gupta, 2005).

Curricula for software development

To fill, effectively, this gap between the knowledge and skills demanded by the industry and the knowledge and skills gained by graduates of university computing courses, it would be necessary on the one hand to guarantee that the educational programmes provide the knowledge required for the job profiles suggested by industry; and on the other hand to ensure that this knowledge is taught in a manner enabling future professionals to tackle correctly the problems they will face during their professional career (Gupta, 2005; Loftus *et al*, 2011). Plice and Reinig (2007) found that emphasizing technical topics at the expense of business content in university courses may provide short-term benefits in making the transition into the workforce, but it might inhibit career advancement as graduates assume greater managerial responsibilities. Emphasizing communication and teamwork skills, while maintaining the existing curriculum balance between business and technical content, is indicated as an appropriate strategy for aligning the computing curricula with the needs of industry.

The Joint Task Force on Computing Curricula (Joint Task Force, 2013) emphasized that the education students receive must adequately prepare them for the workforce in a more holistic way than simply conveying technical facts. Students will, through the general university experience, acquire some soft skills and personal attributes (for example, patience, time management, work ethic, and an appreciation for diversity), but for the rest of the skills provision must be made through specific curricula.

The adoption of international curricula offers a ready means for updating university computing curricula, but universities in developing countries face challenges of implementing these curricula which are typically designed for Western realities and which therefore do not address local needs (Bass and Heeks, 2011; Ezer, 2006).

Lethbridge *et al* (2007) argue that the majority of quality and budgetary problems with software have their root cause in human error or lack of skill. These in turn arise in large part from inadequate education; thus improving education should go a long way towards improving software and software practice.

Methodology and data collection

In this section the research design, the demographics of the participants, as well as the data collection and analysis are discussed.

Research design and participants

A mixed methods approach was used to conduct the research. Tashakkori and Creswell (2007) describe mixed methods as: 'Research in which the investigator

Table 1. Studies on the knowledge and skills gap.

Author(s)	Study	Results
Lethbridge (2000)	Survey of software practitioners on what they thought about 75 educational topics.	Gaps in HCI/user interfaces, real-time system design, software cost estimation, software metrics, software reliability and fault tolerance, and requirements gathering and analysis. Mathematical topics over-emphasized.
Kitchenham *et al* (2005)	Surveyed SE graduates to assess the extent to which the education delivered by four UK universities matches the requirements of the software industry.	Gaps in Web-based programming, project management, configuration and release management, multimedia, security and cryptography, computer graphics, and business topics. Mathematical topics over-emphasized.
Kim *et al* (2006)	Examines IS/IT skills gaps from three perspectives: end-users, academia, and IS/IT employers.	Gaps in project management; most basic and widely used technologies (personal productivity and desktop operating systems), security, ERP, end-user computing, and the integration of soft skills.
Surakka (2007)	A small survey of software developers, academia and Master's students to evaluate the importance of subjects in demanding programming tasks.	Gap in Web-related subjects and skills. Mathematical topics over-emphasized.
Benamati and Mahaney (2007)	Thirteen IS executives were interviewed to learn their views on the state of the entry-level IS job market and what skills today's IS graduates lack most.	Lack in programming skills, project management skills, communications skills, business knowledge, and leadership skills.
Lee and Han (2008)	Investigated the skill requirements for a programmer/analyst by analysing 837 job ads posted on Fortune 500 corporate websites.	Require skills related to development, software, social skills, and business.
Aasheim *et al* (2009)	Compared the perceptions academics have of the importance of various skills for entry-level IT workers with the view of IT managers.	IT managers place more importance on hardware concepts, operating systems, leadership skills or entrepreneurial traits than academia. Both groups ranked interpersonal skills, personal skills, technical skills, organizational skills and work experience – in the same order of importance.
Bullen *et al* (2009)	Examined workforce trends in IT companies.	Companies seek client-facing capabilities, project management skills and business domain knowledge.
Gallagher *et al* (2010)	Interviewed senior IT managers in non-IT companies to investigate the premise that IT professionals should possess a varied set of skills.	Skills most critical are non-technical skills, such as project management, business-domain knowledge and relationship skills.
Moreno *et al* (2012)	Investigated the relationship between the competences of recent SE graduates and the tasks that these professionals are to perform as part of their jobs in industry.	The biggest gaps found concern tasks associated with IT business consultancy, knowledge related to leadership, negotiation or giving presentations.
Keil *et al* (2013)	Investigated the skill requirements for IT project managers in (IT) projects.	The top ?ve skills identi?ed were leadership, verbal communication skills, scope management, listening skills, and project planning.

collects and analyses data, integrates the findings, and draws inferences using qualitative and quantitative approaches or methods in a single study or programme of inquiry'. Mixed methods research can help develop rich insights into various phenomena of interest that cannot be fully understood using only a quantitative or a qualitative method. Often, mixed methods research will provide the most informative, complete, balanced and

useful research results (Venkatesh *et al*, 2013; Johnson *et al*, 2007). Creswell and Clark (2007) suggested four major types of mixed methods design: (1) triangulation (merge qualitative and quantitative data to understand a research problem); (2) embedded (use either qualitative or quantitative data to answer a research question within a largely quantitative or qualitative study); (3) explanatory (use qualitative data to help explain or elaborate quantitative results); and (4) exploratory (collect quantitative data to test and explain a relationship found in qualitative data). In this study the type of mixed methods research was explanatory, because the objective of the qualitative investigation was to supplement the quantitative investigation and to better understand and explain the observations of the quantitative investigation. This mixed methods study was conducted in South Africa: for the quantitative part of the study a survey was used and the qualitative data were acquired through the comments made by survey respondents.

In the last quarter of 2013 a convenience sample of 995 professional software developers in South Africa was taken. The respondents were members of the following groups of the professional networks LinkedIn and MyBroadband: Software and Web Developers in South Africa, SA Developer.NET and C# Developers/Architects. They were contacted via e-mail and asked to complete the anonymous online survey. Some of the respondents indicated that they had sent the link of the survey to their colleagues for completion. In addition, five managers at software houses were contacted and they sent the link of the survey to the software developers in their company. The number of usable responses received was 214, a response rate of around 21%.

Table 2 provides a summary of the biographical data of the respondents. The gender profile, with only 8% of the respondents being female, is a matter for concern – but not surprising. The age profile shows that 42% of the respondents were 'young' (aged under 30) software developers. A greater concern is the ethnic background of the software developers, with only 19% of the respondents being Black. In terms of the respondents' education 49% of them were in possession of a CS/IS degree or degrees, with another 22.5% having related degrees.

It is not uncommon in software development to find people with few formal qualifications (4.5% of the respondents): they often teach themselves to programme and then prove themselves to employers in the software industry through their knowledge, skills and experience. The work experience of respondents indicates that 70.5% of them had more than five years' work experience. A common trend is for software developers not to stay long in one position or workplace: this became apparent in this study, with only 43% of the respondents having worked for more than two years with their current employer. About a quarter of the respondents (27%) were part of management and 46% of them were involved in the hiring of new graduates.

Table 2. Profile of respondents (*n*=214).

		Number (%) of respondents
Gender	Male	196 (92%)
	Female	18 (8%)
Age category	18–24	25 (12%)
	25–29	64 (30%)
	30–39	94 (44%)
	40–49	28 (13%)
	50–59	3 (1%)
	60+	0 (0%)
Ethnic background	African/Black	40 (19%)
	White	145 (68%)
	Coloured[a]	11 (5%)
	Indian/Asian	14 (6%)
	Other	4 (2%)
Education	Matric	10 (4.5%)
	Certification	22 (10%)
	National Diploma	30 (14%)
	CS/IS degree(s)	104 (49%)
	BSc/BCom	38 (18%)
	Engineering degree	10 (4.5%)
Work experience (in years)	0–4	63 (29.5%)
	5–9	62 (29%)
	10–14	51 (24%)
	15–19	22 (10%)
	20–29	13 (6%)
	30–39	3 (1.5%)
	40+	0 (0%)
Years at current employer	0–2	123 (57%)
	3–4	47 (22%)
	5–9	26 (12%)
	10–19	13 (6%)
	20–29	4 (2%)
	30–39	0 (0%)
	40+	1 (0.5%)
Involved in hiring of new graduates?	Yes	99 (46%)
	No	115 (54%)
Part of management?	Yes	58 (27%)
	No	156 (73%)

Note: [a] The term 'Coloured' is used by government organizations in South Africa, among others, as an ethnic label for people of mixed ethnic origin who possess ancestry from Europe, Asia and various Khoisan and Bantu tribes of Southern Africa and is not considered derogatory.

Table 3. Factors (with reliability coefficients) and items.

Factors	Set 1: Cronbach's α	Set 2: Cronbach's α
Information systems	0.800	0.812
Computer hardware and electrical and computer engineering	0.937	0.907
Software testing and maintenance	0.900	0.833
Computer science theory	0.912	0.908
Real-time and systems programming	0.888	0.867
Mathematics and statistics	0.933	0.910
Mobile technologies	0.946	0.967
Software development methodologies	0.915	0.803
Software management	0.882	0.831
General software design	0.853	0.834
Specialized application techniques	0.865	0.884
Web design and development	0.914	0.911
Hardware: data transmission	0.851	0.820
Software engineering methods	0.901	0.867
Items		
Data warehousing		
Security and cryptography		
Game development		
HCI/user interfaces		
Essential subsystem design: databases		

Note: See Appendix for the items in each factor.

Data collection, instrument and analysis

A survey with two sets of 63 items, and an open-ended question at the end of the questionnaire asking for further comments on the education of software developers, was developed. The first section of the questionnaire gathered information on the biographical data of the respondents as shown in Table 2. The second and third sections both listed the same 63 core software development topics. Similar to Lethbridge (2000), the first set of 63 topics asked in respect of each topic: 'How much did you learn about this in your formal education?' and was accompanied by a five-point Likert response scale with 1 ('Learned nothing at all'); 2=('Became vaguely familiar'); 3=('Moderate working knowledge'); 4=('Learned a lot'); 5=('Learned in depth; became expert'). The second set asked in respect of each of the 63 topics: 'How important have the details of this specific material been to you in your career as a software developer?' and was accompanied by a five-point Likert response scale with 1 ('No importance') 2=('Occasionally important'); 3=('Moderately important'); 4=('Very important'); 5 ('Essential').

Factor analysis was used to investigate the two sets of 63 items in more detail, to reduce the variables into a smaller number of factors but taking into account that in order to answer the third research question, the two sets needed to be comparable. The 214 responses were examined using principal components factor analysis and the two sets of attitude items each yielded 14 interpretable factors and five items were being handled as single research variables for each set. Factors were named according to their main context. A Cronbach's Alpha coefficient was calculated for each of the factors and was found, as Table 3 shows, to be reliable ($\alpha \geq 0.60$).

Kitchenham *et al* (2005) raised concerns over the population in the study by Lethbridge (2000), namely that some of the respondents graduated a very long time prior to the study, and some graduated in non-computer science-related disciplines or did not graduate at all. For this present study the concerns of Kitchenham *et al* (2005) were therefore addressed. Cross tabulation (see Table 4) was used to establish the core group of respondents with, in the first place, a pure CS/IS degree; and, in the second place, respondents who had completed their degree in the past fifteen years. These criteria were met by 93 respondents – from hereon referred to as the 'Core Group' – because their education can be considered as relatively recent and these respondents could offer useful information about current computer science-related courses. For the first and second research questions, the Core Group's views are reported together with those of the whole group, but for the third research question, when establishing the gap, only the Core Group's views are reported because only they can give a clear indication of the possible gap.

Basic analysis of quantitative data was done by calculating the mean values and standard deviation of each of the 19 variables. The statistical tests used in the

Table 4. Cross tabulation of experience vs education.

		Work experience (years)						
		0–4	5–9	10–14	15–19	20–29	30–39	Total
Education	Matric	1	4	2	2	0	1	10
	CS/IS degree(s)	37	35	21	7	4	0	104
	BSc/BCom degree	8	10	11	5	4	0	38
	Certification	9	2	5	2	2	2	22
	National diploma	7	8	8	5	2	0	30
	Engineering degree	1	3	4	1	1	0	10
Total		63	62	51	22	13	3	214

analysis varied as necessary to match the metric being analysed. When the results of the interaction analysis are reported, only the significant interactions or primary effects will typically be discussed. A convenience sample instead of a random sample was used therefore the *p*-values will be reported for the sake of completeness but will not be interpreted.

The qualitative data gathered in the open-ended question at the end of the questionnaire came from 77 of the respondents. In addition, there were 21 respondents who felt so strongly about the topic that they gave up their anonymity and e-mailed the researcher with more comments and suggestions.

The ATLAS.ti 7.1.4 computer program was used for the analysis of the qualitative data. The data were stored as a hermeneutic unit and coded into themes and subthemes and analysed for dominant themes. Given that the objective of the qualitative investigation was to supplement the quantitative investigation, the question central to this analysis was, 'What knowledge is important to professional software developers in their actual workplace?' From the data analysis, some patterns emerged and the themes identified were:

- The knowledge needed by industry;
- The knowledge not needed by industry;
- The state of SD education; and
- Suggestions for SD education.

From the data analysis an overall description of the respondents' experiences and feelings about software development education was created. Since the product of qualitative research is richly descriptive (Merriam, 2009), the results of this part of the study are presented in the form of quotations taken from the participants' comments.

Threats to validity

As previously stated, the dynamic nature of computing makes it difficult to define the computing disciplines and related terminology. In this study the focus was on software developers, and all of the respondents fitted the label 'software developer'; but what a software tester, web developer, software architect, project manager, or any of a host of other software-related professionals might need in preparation might vary. Furthermore, software developers are dispersed across many different industries (banking, service industries, etc) and there would be many different needs to address. The researchers had taken care to select the software development groups from LinkedIn and MyBroadband: the views and opinions of these experienced software developers do not necessarily only represent their specific profession or sector, because a project manager in the SD department of a bank will, for instance, notice what knowledge and skills are lacking in the software development team and knowledge regarding software testing is not bound to a specific sector.

Software engineering (SE) is not yet offered as a separate degree programme at universities in South Africa, and software developers in South Africa therefore originate from CS/IS/IT degree programmes. The results of this study could not therefore be used as a remedy to fix a single degree programme; but, rather, as a guideline for role players ranging from lecturers, developers of degree programmes/curricula to corporate trainers.

Results and discussion

In this section, important data for each of the concepts are considered, as well as the qualitative data that provide a rich description of the information obtained. The qualitative data also helped the researchers to discover and gain understanding of the perspectives of the professional software developers regarding the topics they learned from their formal education and the importance of these topics to their actual work.

University courses

Table 5 shows the topics the Core Group (n=93) and the whole group (n=214) of professional software developers learned in their formal education. It is

Table 5. Topics learned in formal education.

Topics	Core Group Mean[a] (*n*=93)	Core Group SD	Whole group Mean[a] (*n*=214)	Whole group SD
Essential subsystem design: databases	3.613	0.847	3.308	0.992
General software design	3.240	0.758	3.071	0.806
Computer science theory	3.223	0.902	3.043	0.969
Mathematics and statistics	3.132	0.972	2.908	1.137
HCI/user interfaces	2.978	1.251	2.724	1.148
Software engineering methods	2.875	0.941	2.718	0.950
Real-time and systems programming	2.824	0.850	2.650	0.893
Security and cryptography	2.699	1.130	2.341	1.171
Hardware: data transmission	2.694	1.045	2.509	1.066
Web design and development	2.621	1.030	2.296	1.068
Information systems	2.541	0.891	2.321	0.903
Data warehousing	2.538	1.138	2.145	1.080
Specialized application techniques	2.490	0.950	2.265	0.937
Software testing and maintenance	2.336	0.951	2.158	0.902
Software management	2.263	0.869	2.026	0.877
Software development methodologies	2.161	1.033	1.827	0.982
Computer hardware and electrical and computer engineering	2.147	0.839	2.170	0.997
Mobile technologies	1.987	1.151	1.734	1.013
Game development	1.667	1.004	1.472	0.848

Note: [a] Likert-style responses were ranked from 1 to 5 respectively.

noteworthy that there are no significant differences between the Core Group and the whole group. Most of the differences might be explained by the 38 respondents who graduated more than 14 years prior to the study.

The topic that they learned the most was 'Essential subsystem design: databases' with the mean value indicating that their knowledge ranged from 'Learned a lot' to 'Moderate working knowledge'. They also rated 'Moderate working knowledge' on the topics: General software design; Computer science theory; and Mathematics and statistics. The topics they learned the least ('Became vaguely familiar') were 'Game development' and 'Mobile technologies'. It is a matter for concern that little was taught regarding 'Mobile technologies', because mobile broadband subscriptions were, and are, showing such a considerable growth in Africa.

The number of years' experience of the respondents was taken into consideration and was tested for significant differences between means of the respondents with less than 15 years' experience against those with 15 and more years' experience, using a *T*-test. There were three factors showing medium practically significant differences in terms of what they learned in their formal education, namely 'Mobile technologies' (d=0.50), 'Software development methodologies' (d=0.60) and 'Web design and development' (d=0.69). It is not surprising that these newer technologies and methods were not taught to the 'older/more experienced' respondents because these technologies might not even have existed when they received their education. These results also confirm that in order to establish the gap between current SD education and the workplace, these older respondents had to be left out of the equation.

The *T*-test also showed that there were no significant differences in the views of the 'older/more experienced' respondents and the rest in terms of the important topics in the workplace. This fact confirms that the whole group of respondents' views are important in answering the second research question.

Knowledge needed

Table 6 shows that these software developers viewed the topics in the workplace from 'Essential', 'Very important' through 'Moderately important' to 'Occasionally important'. These software developers view 'Essential subsystem design: databases' as the most important topic in the workplace, but it is encouraging to see that the same topic ranked first in the topics they learned in their formal education. 'General software design', followed by 'Web design and development' were viewed as very important topics in the workplace. The result for 'Web design and development' is in agreement with that of Surakka (2007) and Kitchenham (2005), that the industry views Web-related subjects and skills as important.

The topics viewed as the least important ('Occasionally important') were 'Game development'

Table 6. Important topics in the workplace.

Topics	Core Group Mean[a] (*n*=93)	Core Group SD	Whole group Mean[a] (*n*=214)	Whole group SD
Essential subsystem design: databases	4.419	0.864	4.336	0.924
General software design	4.029	0.702	3.977	0.783
Web design and development	3.903	1.073	3.770	1.117
Software testing and maintenance	3.796	0.836	3.738	0.831
HCI/user interfaces	3.634	1.101	3.617	1.131
Software engineering methods	3.584	1.015	3.564	0.955
Mobile technologies	3.476	1.298	3.386	1.346
Software development methodologies	3.427	1.065	3.159	1.041
Security and cryptography	3.409	1.236	3.486	1.141
Software management	3.296	0.963	3.254	0.958
Data warehousing	3.215	1.214	3.037	1.214
Real-time and systems programming	3.138	1.079	3.169	1.000
Computer science theory	3.099	1.156	3.040	1.112
Information systems	2.792	1.104	2.776	1.082
Mathematics and statistics	2.621	1.025	2.584	1.068
Hardware: data transmission	2.559	1.091	2.680	1.094
Specialized application techniques	2.249	0.977	2.228	0.971
Computer hardware and electrical and computer engineering	1.880	0.843	1.985	0.882
Game development	1.591	1.024	1.636	0.982

Note: [a] Likert-style responses were ranked from 1 to 5 respectively.

and 'Computer hardware and electrical and computer engineering'. The qualitative data revealed the following regarding the knowledge needed by software developers. Respondents felt that students lacked certain knowledge and skills especially in relation to the way in which software development takes place in the real world:

> 'The common problem is that they have no concept of how real projects are managed, how projects are planned and timings are estimated, and various other things relevant to real-world development.'
>
> 'I think future developers need some introduction in the SDLC of the workplace along with something like SCRUM.'
>
> 'More practical exposure, become language and os agnostic use best tool for the job.'
>
> 'I think the education should focus on what is used in the industry like Agile and Extreme Programming and software practices (TDD).'
>
> 'Practical SDLC experience in a team setting (systems development projects) is a crucial part of the learning process.'
>
> 'General problem solving should be more of a focus.'
>
> 'They should learn the basic fundamentals and not just the methods to provide quick solutions.'
>
> 'The concepts are far more important (design patterns, algorithms).'
>
> 'Met too many honours degree students that don't grasp object-oriented design and design patterns.'

One of the respondents echoed the findings of other researchers (Mawson, 2010; Kitchenham *et al*, 2005; Kim *et al*, 2006; Plice and Reinig, 2007) regarding the lack of business knowledge: for example, 'I wasn't prepared from a business knowledge perspective'; and another respondent commented, 'When I first started I had very little idea of how to begin working with an existing code base and team structures'.

Respondents commented on recent developments and trends and what should be taught to students in order to stay up to date:

> 'The Functional Program (immutable) paradigm is taking over. Rather teach interfaces, generics, functions, delegates, lambdas and drum in the basics of stacks, heaps, queues and thread safety, etc.'
>
> 'Distributed Systems and Parallel computing are becoming more important now.'
>
> 'SOLID (Uncle Bob), Domain Driven Design (Evans) and Brock – Test-Driven Design are really useful.'
>
> 'SOLID is extremely important for writing good, maintainable, testable, extendable code.'

Table 7. The gap between university courses and workplace practice.

	Mean (n=93)			
Topics	**Learned in formal education**	**Importance in workplace**	**Effect size**	***p***
Software testing and maintenance	2.336	3.796	1.54**	<0.001
Software development methodologies	2.161	3.427	1.19**	<0.001
Web design and development	2.621	3.903	1.19**	<0.001
Mobile technologies	1.987	3.476	1.15**	<0.001
Software management	2.263	3.296	1.07**	<0.001
General software design	3.240	4.029	1.04**	<0.001
Essential subsystem design: databases	3.613	4.419	0.93**	<0.001
Software engineering methods	2.875	3.584	0.70*	<0.001
Security and cryptography	2.699	3.409	0.57*	<0.001
Data warehousing	2.538	3.215	0.56*	<0.001
Human–computer interaction/user interfaces	2.978	3.634	0.52*	<0.001
Real-time and systems programming	2.824	3.138	0.29	
Information systems	2.541	2.792	0.23	
Mathematics and statistics	3.132	2.621	0.50*	<0.001
Computer hardware and electrical and computer engineering	2.147	1.880	0.32	
Specialized application techniques	2.490	2.249	0.25	
Hardware: data transmission	2.694	2.559	0.12	
Computer science theory	3.223	3.099	0.11	
Game development	1.667	1.591	0.07	

Note: * Medium practically significant difference (d>=0.5); ** Large practically significant difference (d>=0.8).

> ‘Database design and development is severely under taught. Strong Knowledge of SQL is paramount and it’s not present.’
>
> ‘Ignore too many Mobile technology specific courses (i.e. no iOS, Android, WindowsPhone, etc) – stick with Web skills. UI design for small devices (and innovative UI design) is valuable.’
>
> ‘Get students to become familiar with version control of some kind through assigned projects.’
>
> ‘They are learning the wrong technologies for the industry; they should be focusing only on the .NET stack and LAMP stack and mob.’
>
> ‘They need to be taught industry relevant languages, i.e. C# and proper platforms if they come from JAVA.’

The knowledge not needed

Respondents had strong opinions on knowledge and topics that they felt were not important and which topics were over-emphasized.

> ‘Languages and syntax aren’t really important.’
>
> ‘OO is *dying* and Inheritance and Polymorphism is over emphasized at Varsity!’
>
> ‘Mutation of state of objects is no longer a desirable paradigm as it limits parallelism.’
>
> ‘Ignore too many mobile technology specific courses (i.e. no iOS, Android, WindowsPhone etc).’
>
> ‘Web dev is a fad. Need more hard core real-time programmers.’
>
> ‘They should learn the basic fundamentals and not just the methods to provide quick solutions.’

University courses versus workplace practice

The results of the Core Group were analysed to determine if there was a gap between software development education and the workplace from the perspective of the software industry. Differences were analysed with a *T*-test and Table 7 shows significant differences in means between 19 factors.

Thirteen factors revealed that their mean values for importance in the workplace were higher than the mean values of what they learned in their formal education. Six factors (Mathematics and statistics; Computer hardware and electrical and computer engineering; Specialized application techniques; Hardware: Data transmission; Computer science theory; Game development) showed lower mean values for importance in the workplace than the mean values of what they learned in their formal education.

Seven factors (Software testing and maintenance; Software development methodologies; Web design and development; Mobile technologies; Software management; General software design; Essential

subsystem design: Databases) showed large practically significant differences between what they learned in their formal education and what they view as important in the workplace. All seven factors indicated that these highly important topics were not extensively taught. Software testing and maintenance showed a very large difference and clearly needed a lot more coverage in their education, since it also ranked fourth in the most important topics in the workplace. The study by Lethbridge (2000) also found a gap in the education of software management (software cost estimation; software metrics) and software testing and maintenance (software reliability and fault tolerance).

There were also five factors showing a medium practically significant difference between what they learned in their formal education and what they view as important in the workplace. Four of the five factors (Software engineering methods; Security and cryptography; Data warehousing; Human–computer interaction/user interfaces) indicated not only that their education in these topics was lacking but also that the factor Mathematics and statistics is overemphasized at university. This finding is in agreement with the results of Lethbridge (2000), Kitchenham *et al* (2005) and Surakka (2007) in respect of the excessive importance attached to mathematics-related topics at university.

However, one respondent contradicted the quantitative results that Mathematics and statistics are not very important in the workplace:

> 'Math – calculus/algebra/matrices, etc. – data structures – algorithms – machine learning!!! totally beneficial.'

Venkatesh *et al* (2013) stated that when conducting mixed methods research a researcher may find contradictory conclusions from the quantitative and qualitative strands, but these contradicting findings are valuable in that they not only enrich our understanding of a phenomenon but also open new avenues for future inquiries. It is noteworthy that the above-mentioned studies were conducted between 2000 and 2007. It would seem that universities might have taken note of the findings and decreased their coverage of mathematics to such an extent that some people in the SD industry are beginning to feel that students lack the necessary education in mathematics. This contradictory finding regarding mathematics is a clear indication that further research might be necessary.

The state of SD education

Some of the respondents were quite negative and felt there was a gap between what students learn at university and what is important in the workplace:

> 'I do think there is distinct disconnect between what students are being taught at varsity and what they actually need to know to work in a proper software development house and be useful and productive.'

> 'I really feel that the tertiary education system is failing them horribly.'

> 'I have been attempting to convince XYZ University to produce more able software developers, but nothing is changing.'

Not all the respondents were negative about SD education, however:

> 'On the plus side, I don't think that their degrees can be considered easy or of little value by any means. They clearly worked very hard to earn those degrees and they do seem to be taught a lot of solid foundational principles that can be built on very easily.'

> 'Computer science degrees often don't prepare an individual to go out there and start developing systems from scratch, but it gives them a long-term advantage, a broad knowledge and understanding of how they should learn development and what they should try to avoid.'

Suggestions for software development education

Respondents offered suggestions to improve SD education, which included practical experience for students and keeping the curriculum up to date – although in this latter regard the fact that the industry changes at such a rapid pace was acknowledged:

> 'Universities should bridge this gap by integrating more experience into curriculum.'

> 'The most useful thing that varsities could probably do would be to try and make the material they are teaching the students more current and more in-line with what is actually going on out there. I realize that is incredibly difficult when you are trying to plan a syllabus ahead of time and the industry changes at the pace that ours does, but if you want to improve the marketability of students straight out of varsity, that's what will do it.'

> 'Lecturers should be able to teach more up to date skills to students, and open their eyes to possible exposures to methodologies.'

Some respondents felt that since the industry changes so rapidly students should instead be taught the foundational and theoretical knowledge of SD:

> 'Education should be about important principles and knowledge, not what businesses may require at any particular time.'

> 'The market has a severe lack of solid theoretical computer science training.'

> 'Practical software development skills change rapidly so tertiary institutions should focus on general theory.'

One of the respondents acknowledged the fact that the majorities in South Africa are the minorities in SD classes and the workplace (also reflected in Table 2, with only 18% females and 19% African/Black respondents) and called on universities to prioritize diversity: 'Increased diversity within the student body should be a priority for universities.'

Software development education should not only include technical knowledge:

> 'The education of software developers should emphasize soft skills as much as technical skills.'

Respondents spoke of the gap that opens up after SD students have left university:

> 'No problem with the education. What falls flat is what happens with IT companies after graduation.'

> 'A high and consistent standard of education in South Africa is exceptionally difficult to find, especially continuing education.'

It is not uncommon in software development to find people without a computing degree but with a lot of experience – computing classes must be delivered with a variety of methods and to a variety of students, as these responses illustrate:

> 'There are opportunities for education institutions to pull in ''DIY'' students like me into the courses they offer. The Internet has a lot of free information making it easy to build your skills. I think a lot of people see this as an option for training and they ignore the traditional education institutions.'

> 'Look at for example Udacity (https://www.udacity.com/) and similar MOOCs. They draw the masses and I think it forces universities to change their approach towards education.'

Respondents felt strongly that the university and the IT industry should work together in creating up-to-date curricula.

> 'I think it's of vital importance in the software industry, more important than any other industry, that academic institutions consult with the private sector to learn how best to equip students for their first real-world position as a software engineer.'

> 'Industry professionals NEED to be brought in for consultation on the tool chain they make daily use of in their development role.'

Conclusions and recommendations

There is a shortage of skilled software developers, but there is a gap between the education of software developers and what they actually need to know to work in a software development house. The rapid pace at which technology is changing often causes the knowledge graduates acquire at university to be outdated. Graduates often lack business knowledge and they generally lack experience of teamwork, and general practical experience with real-life projects.

The objective of this study was specifically *not* to determine either the needs of a specific sector or the knowledge needed for a specific development role. Rather, the study provides information regarding the topics viewed as important in the workplace and the extent to which these topics are taught in university courses from the perspective of software development professionals. The results are further supplemented by the qualitative findings providing rich insights into the perspective of the SD industry regarding SD education.

We would argue that the purpose of a university is not that of a vocational training institution, but rather of a higher education institution. The university cannot be expected to deliver software developers who can contribute productively to the development of software from the first day they enter the workplace. However, universities must consider increasing their coverage of the above-mentioned topics and take cognisance of the specific knowledge and skills the software industry seeks.

Equally, the software industry must anticipate that graduates will be underprepared in the above-mentioned topics and, as employers, they can put mechanisms in place, such as in-house training on these topics, to fill the gaps effectively.

The following are recommended for the provision of relevant software development education from the perspective of the industry.

- *More coverage.* The topics that need significantly more coverage are: software testing and maintenance; software development methodologies; web design and development; mobile technologies; software management; general software design; and essential subsystem design: databases.
- *Real-life projects*. Real-life and practical experience must be included in students' education.
- *Soft skills and business skills*. Universities must examine their curricula to ensure that not only technical but also soft skills and business skills are included.
- *Up to date*. Universities must attempt to keep pace with the rapid changes in technology.
- *Diversity*. SD education must be made accessible to a diverse range of students, including minority groups.
- *Continuing education.* Universities as well as industry must put mechanisms in place in order for SD workers wanting to continue and expand their education to stay at the forefront of the latest developments in the SD field.
- *Teamwork.* The university and the IT industry should work together in creating up-to-date curricula. Individuals from industry can be brought into software development classes: lecturers can acquire industry experience.

References

Aasheim, C., Li, L., and Williams, S. (2009), 'Knowledge and skill requirements for entry-level information technology workers: a comparison of industry and academia', *Journal of Information Systems Education*, Vol 20, No 3, pp 349–356.

Bass, J., and Heeks, R. (2011), 'Changing computing curricula in African universities: Evaluating progress and challenges via design–reality gap analysis', *The Electronic Journal of Information Systems in Developing Countries*, Vol 48, No 5, pp 1–39.

Bateman, K. (2013), 'The irony of an unemployment problem and an IT skills shortage within the IT industry', ComputerWeekly.com, October, http://www.computerweekly.com//itworks/2013/10/the-irony-of-an-unemployment-p.html (accessed 30 July 2014).

Benamati, J., and Mahaney, R.C. (2007), 'Current and future entry-level IT workforce needs in organizations', Proceedings of the 2007 ACM SIGMIS CPR Conference, *Computer Personnel Research: The Global Information Technology Workforce* (SIGMIS CPR '07), ACM, New York, pp 101–104).

Biztech Africa (2013), 'Africa's high end ICT skills shortfall grows', 25 November, http://www.biztechafrica.com//africas-high-end-ict-skills-shortfall-grows/7302/#.UqwYi_SnqSA (accessed 21 January 2014).

Bullen, C., Abraham, T., Gallagher, K., Simon, J.C., and Zwieg, P. (2009), 'IT workforce trends: implications for curriculum and hiring', *Communications of the Association for Information Systems*, Vol 24, pp 129–140.

Calitz, A.P. (2010), 'A model for the alignment of ICT education with business ICT skills requirements', DBA thesis, Nelson Mandela Metropolitan University, Port Elizabeth.

Connolly, B. (2013), 'IT worker shortage continues as jobs remain unfilled', CIO. http://www.cio.com.au/article/454650/_worker_shortage_continues_jobs_remain_unfilled/ (accessed 31 October 2013).

Creswell, J.W., and Clark, V.L.P. (2007), *Designing and Conducting Mixed Methods Research*, Sage, Thousand Oaks, CA.

Employment Equity Act (South Africa) (1998), Number 55 of 1998, Government Printer, Pretoria.

Ezer, J. (2006), 'India and the USA: a comparison through the lens of model IT curricula', *Journal of Information Technology Education*, Vol 5, pp 429–440.

Fernandez-Sanz, L. (2009), 'Personal skills for computing professionals', *Computer*, Vol 42, No 10, pp 110–112.

Gallagher, K.P., Kaiser, K.M., Simon, J.C., Beath, C.M., and Goles, T. (2010), 'The requisite variety of skills for IT professionals', *Communications of the ACM*, Vol 53, No 6, pp 144–148.

Gruner, S. (2014), 'On the historical semantics of the notion of software architecture', *TD: The Journal for Transdisciplinary Research in Southern Africa*, Vol 10, No 1, pp 37–66.

Gupta, A. (2005), 'Securing the future of the Indian IT industry: a case for educational innovation in higher technical education–challenges and the road ahead', *Industry and Higher Education*, Vol 19, No 6, pp 423–431.

Guzdial, M., Prey, J., Topi, H., Urban, J., Cassel. L., and Schneider, D. (2009), 'Future of computing education summit', 25–26 June 2009, http://www.acm.org/education/future-of-computing-education-summit/FoCES_web.pdf (accessed 30 July 2014).

Harris, L. (2012), 'Mind the ICT skills gap', *Brainstorm*, September,http://www.brainstormmag.co.za/index.php?option=_content&view=article&id=4699:mind-the-ict-skills-gap&catid=92:features&Itemid=125 (accessed 24 July 2013).

ISO/IEC 25000 (2014), *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE*. ISO/IEC, Geneva, Switzerland.

ISO/IEC 19770–2 (2009), *Information Technology – Software Asset Management – Part 2: Software Identification Tag*, ISO/IEC, Geneva, Switzerland.

ISO/IEC/IEEE 24765 (2010), *3.2758: Systems and Software Engineering – Vocabulary*, ISO/IEC/IEEE, Geneva, Switzerland.

Johnson, R.B., Onwuegbuzie, A.J., and Turner, L.A. (2007), 'Toward a definition of mixed methods research', *Journal of Mixed Methods Research*, Vol 1, No 2, pp 112–133.

Joint Task Force [on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society] (2013), 'Computer science curricula 2013: curriculum guidelines for undergraduate degree programs in computer science', ACM, New York.

Jones, C., Ramanau, R., Cross, S., and Healing, G. (2010), 'Net Generation or Digital Natives: is there a distinct new generation entering university?', *Computers and Education*, Vol 54, No 3, pp 722–732.

Keil, M., Lee, H., and Deng, T. (2013), 'Understanding the most critical skills for managing IT projects: a Delphi study of IT project managers', *Information and Management*, Vol 50, pp 398–414.

Kim, Y., Hsu, J., and Stern, M. (2006), 'An update on the IS/IT skills gap', *Journal of Information Systems Education*, Vol 17, No 4, pp 395–402.

Kitchenham, B., Budgen, D., Brereton, P., and Woodall, P. (2005), 'An investigation of software engineering curricula', *Journal of Systems and Software*, Vol 74, No 3, pp 325–335.

Krakovsky, M. (2010), 'Degrees, distance, and dollars', *Communications of the ACM*, Vol 53, No 9, pp 18–19.

Lee, C., and Han, H. (2008), 'Analysis of skills requirement for entry-level programmer/analysts in Fortune 500

corporations', *Journal of Information Systems Education*, Vol 19, No 1, pp 17–27.
Lethbridge, T.C. (2000), 'Priorities for the education and training of software engineers', *Journal of Systems and Software*, Vol 53, No 1, pp 53–71.
Lethbridge, T., Diaz-Herrera, J., LeBlanc, R., and Thompson, J.B. (2007), 'Improving software practice through education: challenges and future trends', in *Future of Software Engineering*, IEEE Computer Society, Washington, DC, 2007, pp 12–28.
Loftus, C., Thomas, L., and Zander, C. (2011), 'Can graduating students design: revisited', in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*, ACM, New York, pp105–110.
Lorenzo, G., Oblinger, D., and Dziuban, C. (2007), 'How choice, co-creation, and culture are changing what it means to be net savvy', *Educause Quarterly*, Vol 30, No 1, pp 6–12.
Mawson, N. (2010), 'ICT skills shortage to cost SA', ITWeb, http://www.itweb.co.za/index.php?option=com_&view=article&id=29992 (accessed 30 April 2015).
Merriam, S.B. (2009), *Qualitative Research: A Guide to Design and Implementation*, Jossey-Bass, San Francisco, CA.
Moreno, A., Sanchez-Segura, M., Medina-Dominguez, F., and Carvajal, L. (2012), 'Balancing software engineering education and industrial needs', *Journal of Systems and Software*, Vol 85, No 7, pp 1607–1620.
Mooketsi, B. E., and Chigona, W. (2014). 'Different shades of success: educator perception of government strategy on e-education in South Africa', *Electronic Journal of Information Systems in Developing Countries*, Vol 64, No 8, pp 1–15.
Plice, R. K., and Reinig, B. A. (2007), 'Aligning the information systems curriculum with the needs of industry and graduates', *Journal of Computer Information Systems*, Vol 48, No 1, pp 22.
Reisman, S. (2004), 'Higher education's role in job training', *IT Professional*, Vol 6, No 1, pp 6–7.
Saiedian, H. (2009), 'Software engineering challenges of the "Net" generation', *Journal of Systems and Software*, Vol 82, No 4, pp 551–552.
Shaw, M., Herbsleb, J., and Ozkaya, I. (2005), 'Deciding what to design: closing a gap in software engineering education', in *Proceedings of the 27th International Conference on Software Engineering (ICSE '05)*, ACM, New York, pp 607–608.
Surakka, S. (2007), 'What subjects and skills are important for software developers?', *Communications of the ACM*, Vol 50, pp 73–78.
Tashakkorri, A., and Creswell J.W. (2007), 'The new era of mixed methods (editorial)', *Journal of Mixed Methods Research*, Vol 1, pp 3–7.
UN Broadband Commission (2013), 'The state of broadband 2013: universalizing broadband. A report by the Broadband Commission', September 2013, http://www.broadbandcommission.org/documents/bbannualreport2013.pdf (accessed 13 December 2013).
UN Broadband Commission (2014), 'The state of broadband 2014: broadband for all. A report by the Broadband Commission', September 2014, http://www.broadbandcommission.org/documents/bbannualreport2014.pdf (accessed 23 September 2014).
Venkatesh, V., Brown, S.A., and Bala, H. (2013), 'Bridging the qualitative–quantitative divide: guidelines for conducting mixed methods research in information systems', *MIS quarterly*, Vol 37, No 1, pp 21–54.
Wade, R. (2002), 'Bridging the digital divide: new route to development or new form of dependency?', *Global Governance*, Vol 8, No 4, pp 443–466.

Acknowledgment

The financial assistance of the National Research Foundation (NRF) towards the work done in this research is hereby acknowledged. Opinions expressed, and conclusions arrived at, are those of the authors and are not necessarily to be attributed to the NRF.

Appendix

Items and descriptive statistics (see Table 3)

		Learned in formal education (Set 1)		Important in the workplace (Set 2)	
Factors	**Items**	**Mean[a]**	**SD**	**Mean[a]**	**SD**
Information systems	Information retrieval	2.692	1.129	3.322	1.254
	Decision support systems	2.248	1.083	2.720	1.277
	Expert systems	2.023	0.986	2.285	1.277
Computer hardware and electrical and computer engineering	Digital electronics and digital logic	2.528	1.270	2.192	1.141
	Microprocessor architecture	2.519	1.247	2.107	1.176
	Computer system architecture	2.780	1.106	2.495	1.198
	Analog electronics	1.911	1.181	1.603	0.897
	Digital signal processing	2.014	1.250	1.864	1.145
	Data acquisition	1.874	1.104	2.136	1.243
	Robotics	1.561	0.999	1.500	0.838
Software testing and maintenance	Performance measurement and analysis	2.220	0.999	3.505	1.029
	Testing, verification, and quality assurance	2.215	1.035	4.079	0.929
	Software reliability and fault tolerance	2.210	1.069	3.949	0.965
	Maintenance, re-engineering, and reverse engineering	1.986	1.009	3.421	1.134
Computer science theory	Programming language theory	3.220	1.072	3.121	1.261

		Learned in formal education (Set 1)		Important in the workplace (Set 2)	
Factors	**Items**	**Mean[a]**	**SD**	**Mean[a]**	**SD**
	Formal languages	3.089	1.082	2.995	1.265
	Computational complexity and algorithm analysis	2.972	1.113	3.159	1.250
	Information theory	2.893	1.093	2.883	1.252
Real-time and systems programming	Operating systems	3.019	1.034	3.196	1.210
	Systems programming	2.846	1.021	3.196	1.248
	Data transmission and networks	2.846	1.070	3.379	1.155
	Parallel and distributed processing	2.308	1.117	3.107	1.268
	Real-time system design	2.229	1.130	2.967	1.298
Mathematics and statistics	Discrete mathematics	2.874	1.263	2.491	1.178
	Probability and statistics	3.037	1.174	2.827	1.184
	Linear algebra and matrices	3.070	1.282	2.654	1.268
	Continuous mathematics	2.650	1.261	2.364	1.182
Mobile technologies	User interface design	1.916	1.203	3.393	1.399
	Mobile application development	1.720	1.103	3.336	1.387
	Security and privacy	1.692	1.061	3.477	1.436
	Compatibility	1.607	0.991	3.336	1.424
Software development methodologies	Agile methods	1.897	1.100	3.500	1.190
	Extreme programming	1.888	1.078	2.696	1.262
	Scrum	1.696	1.005	3.280	1.236
Software management	Project management	2.444	1.144	3.322	1.144
	Software metrics	2.098	0.957	2.991	1.092
	Software cost estimation	1.846	1.007	3.154	1.289
	Configuration and release management	1.715	0.963	3.547	1.169
General software design	Data structures	3.187	0.985	3.981	1.105
	Algorithm design	3.079	1.025	3.668	1.149
	Software design and patterns	2.762	1.169	4.112	0.982
	Software architecture	2.612	1.111	4.089	0.958
	Object-oriented concepts and technology	3.425	1.080	4.364	0.923
	Specific programming languages	3.360	0.982	3.645	1.201
Specialized application techniques	Simulation	2.360	1.161	2.495	1.174
	Artificial intelligence	2.252	1.114	1.935	1.086
	Pattern recognition and image processing	2.243	1.232	2.248	1.233
	Computer graphics	2.093	1.118	2.187	1.160
	Parsing and compiler design	2.379	1.183	2.276	1.219
Web design and development	Web-based methods	2.047	1.069	3.477	1.281
	Interface design	2.411	1.263	3.720	1.228
	Security and privacy	2.257	1.177	3.935	1.247
	Web application development	2.467	1.269	3.949	1.275
Hardware: data transmission	Network architecture and data transmission	2.696	1.116	2.916	1.148
	Telecommunications	2.322	1.168	2.444	1.227
Software engineering methods	Requirements gathering and analysis	2.766	0.998	3.874	1.006
	Formal specification methods	2.631	1.092	3.215	1.159
	Analysis and design methods	2.757	1.029	3.603	1.051
Data warehousing	Data-warehousing	2.145	1.080	3.037	1.214
Security and cryptography	Security and cryptography	2.341	1.171	3.486	1.141
Game development	Game development	1.472	0.848	1.636	0.982
Human–computer interaction/user interfaces	Human-computer interaction/user interfaces	2.724	1.148	3.617	1.131
Essential subsystem design: databases	Essential subsystem design: databases	3.308	0.992	4.336	0.924